UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Towards Reliable and Secure Physical Unclonable Functions

by

Mohd Syafiq Mispan

Thesis for the degree of Doctor of Philosophy

July 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Doctor of Philosophy

TOWARDS RELIABLE AND SECURE PHYSICAL UNCLONABLE FUNCTIONS

by Mohd Syafiq Mispan

Physical Unclonable Functions (PUFs) have emerged as a promising primitive that can be used to provide a hardware root of trust for integrated circuit (IC) applications. PUFs exploit the random intrinsic manufacturing process variations that map a set of challenges to a set of responses. The mapping of challenge-response pairs (CRPs) is unique and random to each PUF instance, which makes PUFs a very promising technology for robust security devices. PUFs have been proposed for lightweight IC identification and authentication, and cryptographic key generation. However, as CMOS technology scales down, device ageing becomes more pronounced and introduces reliability issues for PUF circuits. When PUFs undergo ageing, the response changes. As a consequence, the trustworthy identity of the ICs can be violated. The area overhead of an error correction code (ECC) in a PUF-based system needed to generate error-free cryptographic keys also increases. Furthermore, a PUF is physically unclonable but its function is susceptible to modelling attacks from machine learning (ML) techniques. Therefore, providing reliable and secure PUFs for lightweight applications is a major challenge. This thesis studies the reliability of PUFs for lightweight applications under ageing. It also considers the susceptibility of PUFs to ML-based attacks.

This thesis presents three major contributions. The context of the first and second contributions is within the lightweight IC identification and authentication, and the third contribution is within the cryptographic key generation. The first contribution presents an analysis of the impact of ageing on PUF-based differential architectures. The simulation results demonstrate that a differential design technique to build a PUF can be a mechanism to mitigate the first-order dependencies of ageing such as the duty cycle and supply voltage. The second contribution proposes a challenge permutation technique to increase the complexity of the CRP mapping. The technique has been implemented on an Arbiter-PUF using a TSMC 65-nm technology. The simulation results show that using a challenge permutation technique can alter the output transition probability of Arbiter-PUF, resulting in the reduction of its predictability from ≈99% to ≈65%. The challenge permutation technique introduces no extra overhead as it can be implemented by routing obfuscation. Finally, the third contribution proposes a bit selection technique in a dual use of SRAM as a memory and PUF to mitigate the ageing impact and reduce the area overhead of the ECC. The results show that the proposed technique can effectively reduce the bit errors due to ageing and the area overhead of the ECC is reduced by about 6 times compared to that without bit selection.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Mohd Syafiq Mispan , declare that the thesis entitled *Towards Reliable and Secure Physical Unclonable Functions* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: [1],[2],[3],[4],[5],[6],[7]

Signed:.....  .....

Date:.....17/07/2018.....

Acknowledgements

I owe my deepest gratitude to my supervisors, Professor Mark Zwolinski and Dr. Basel Halak, for their precious support and guidance through my research project. Their vision and wisdom in approaching a problem is the greatest thing I have learned. They have been a great mentor and a source of inspiration throughout my Ph.D. journey.

Also, I would like to thank Professor CH Kees De Groot and Dr. Jeff Reeve, for their revision of my work and their valuable advice for my first-year and second-year reports. I would also like to offer my thank to the Ministry of Education Malaysia and the Technical University of Malaysia Malacca, for the financial support. This Ph.D. project would not have been possible without this support.

Finally, my lovely wife, my daughters, my late parents, my parents-in-law and the rest of my immediate family which have always been my strongest supporters. I am greatly thankful for their love and faith in me, and I would like to dedicate this work to them.

Abbreviations

ADC	Analogue-to-digital Converter.
ALU	Arithmetic Logic Unit.
ANN	Artificial Neural Network.
ASIC	Application-specific Integrated Circuit.
BCH	Bose-Chaudhuri-Hocquenghem.
BTI	Bias Temperature Instability.
CPLD	Complex Programmable Logic Device.
CRP	Challenge-response Pair.
DAA	Direct Accelerated Ageing.
DIBL	Drain-Induced Barrier Lowering.
DOS	Denial Of Service.
ECC	Error Correction Code.
EEPROM	Electrically Erasable Programmable Read-only Memory.
ES	Evolution Strategies.
FIB	Focus Ion Beam.
FPGA	Field-Programmable Gate Array.
GE	Gate Equivalent.
HCI	Hot Carrier Injection.
HD	Hamming Distance.
HW	Hamming Weight.
IC	Integrated Circuit.
IOT	Internet-of-Things.
IP	Intellectual Property.
LFSR	Linear Feedback Shift Register.
LR	Logistic Regression.
LS	Laser Stimulation.
MAC	Message Authentication Code.
ML	Machine Learning.
MLP	Multilayer Perceptron.
MTP	Multiple-Time Programmable.
NBTI	Negative Bias Temperature Instability.

NVM	Non-volatile Memory.
OTP	One-Time Programmable.
PBTI	Positive Bias Temperature Instability.
PEA	Photonic Emission Analysis.
POWF	Physical One-way Function.
PUF	Physical Unclonable Function.
RDF	Random Dopant Fluctuations.
RFID	Radio Frequency Identification.
RNG	Random Number Generator.
RO	Ring Oscillator.
ROM	Read-Only Memory.
SAC	Strict Avalanche Criteria.
SEM	Scanning Electron Microscope.
SLP	Single-layer Perceptron.
SOC	System-on-a-chip.
SRAM	Static Random-access Memory.
SUV	Start-up Value.
SVM	Support Vector Machine.
T-D	Trapping/de-trapping.

Nomenclature

C_{ox}	Gate-oxide capacitance
C_d	Depletion layer capacitance
ϵ	Hamming Distance threshold
k_B	Boltzmann constant
L	Channel length
μ_o	Electron or hole mobility
μ	Mean
q	Electrical charge of electron
σ	Standard deviation
$SiON$	Silicon oxynitride
T	Absolute temperature
V_{th}	Threshold voltage
V_t	Thermal voltage
W	Channel width

Chapter 1

Introduction

Nowadays, electronic devices are becoming ubiquitous. The inception of a network of physical objects or the so-called Internet-of-Things (IoT) allows these electronic devices to be interconnected, operated, and controlled remotely through the internet infrastructure. Examples of these applications include secure access, mobile payment, electronic passports, smart meters, and smart homes. With this wide range of applications, they process sensitive, user-specific data, by which if disclosed, may lead to loss of privacy and other unwanted implications. Low-cost pervasive devices, such as Radio Frequency Identification (RFID) devices and wireless sensor nodes are the foundations for building the next generation of ubiquitous networks, IoT [8]. Such devices typically have limited area and energy resources which introduce a significant challenge in providing fundamental security services, such as device authentication and cryptographic key storage/generation.

1.1 Cryptographic Key Storage Technologies

Cryptography is the study and practice of securing communication in the presence of adversaries or third parties. Cryptography can ensure the confidentiality, integrity, authenticity, and acknowledgement of the user data. Cryptographic primitives are low-level algorithms which are used to build secure protocols. These include but not limited to the authentication, digital signatures, one-way functions, encryption, and decryption. Figure 1.1 shows a basic cryptographic algorithm which is an encryption of a plaintext XORed with a secret key.

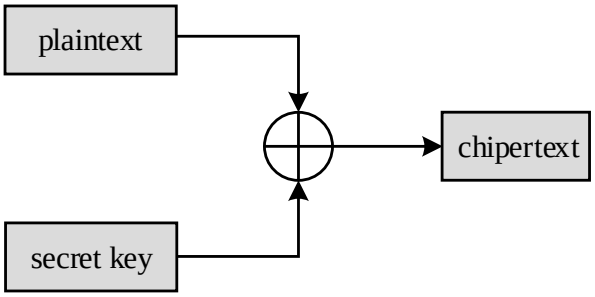


Figure 1.1: Plaintext encryption using a XOR function.

Implementation of current security solutions relies on the secret keys stored in the on-chip non-volatile memory (NVM) or battery-backed static random-access memory (SRAM) [9, 10]. These approaches introduce critical security-related issues. By storing the secret keys in NVM or battery-backed SRAM, the keys are always available. Therefore, they are susceptible to read-out or tampering through invasive or semi-invasive attacks based on techniques derived from integrated circuit (IC) failure analysis. The resilience against these physical attacks can be improved through a tamper-sensing environment which would further increase the cost of implementation. Moreover, the secret keys need to be programmed which often relies on the IC manufacturer or system owner. Hence, there is a possibility that the secret keys can be compromised by an untrusted third party within the product supply chain.

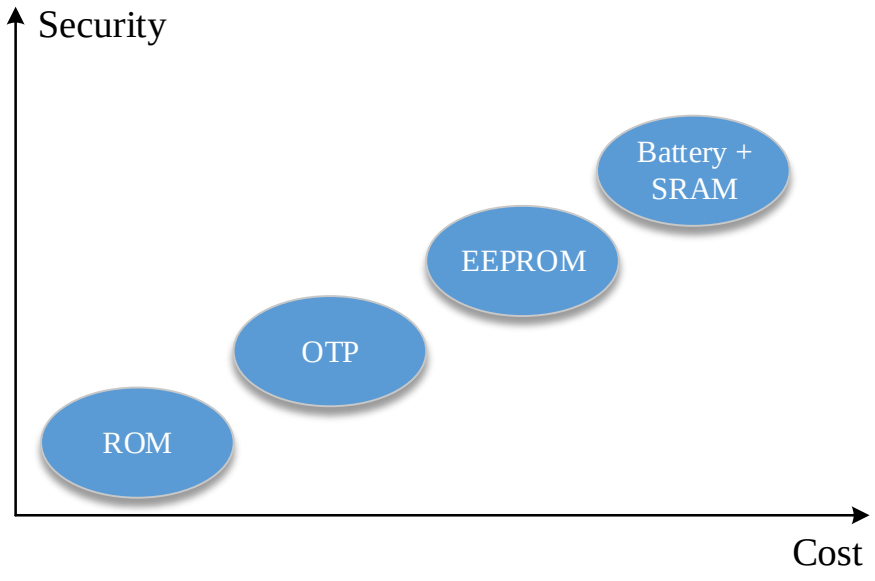


Figure 1.2: Security-cost positioning of permanent key storage technologies [9].

The relative cost of implementation of the aforementioned key storage solutions is depicted in Figure 1.2. Different technologies can be deployed for the key storage on NVM

such as Read-Only Memory (ROM), One-Time Programmable (OTP), and Multiple-Time Programmable (MTP). A lightweight method of storing keys in NVM is by using ROM when large-quantities and low-cost are required. However, this approach has low security and is very inflexible as the secret keys are mask defined and shared by each device [10]. The secret keys, however, can be programmed once through the OTP NVM using fuse or anti-fuse technologies. Reprogramming can be emulated via the partitioning of a large OTP NVM which further increases the cost. Instead, it is more efficient to use MTP NVM, such as electrically erasable programmable read-only memory (EEPROM) which offer more flexibility as the secret keys can be re-programmed many times but still, the cost is high. Battery-backed SRAM requires a standard CMOS SRAM, but the battery that is needed to retain the secret keys in a volatile memory is considered to be costly and bulky (i.e., space overhead), particularly for resource-constrained applications.

1.2 Physical Unclonable Function

A Physical Unclonable Function (PUF) is an emerging technology which offers a promising solution to the critical security-related issues as discussed above with a relatively low cost. PUF maps a set of challenges to a set of responses. Its challenge-response relationship is determined by the intrinsic process variations in the transistor and interconnects of a silicon chip. The intrinsic process variations are caused by uncontrollable deviations in the chip manufacturing process, which are unique and random from die to die and wafer to wafer. Therefore, a PUF can be used to generate a unique and random key or identifier. Furthermore, the complex and random nature of the manufacturing process variations makes a PUF practically and physically impossible to clone [11]. A PUF is considered as secure and low-cost technology since the identifier or the key can only be generated during power-on state and is wiped-out in the power-off state. A PUF can be implemented using a standard CMOS circuit design technique which requires no special fabrication process. Besides, a physically invasive attack to recover the identifier or key, which is only available during the power-on state by a micro-probing technique, is most likely to destroy the unique delay characteristics, effectively destroying the identifier or key [12]. This makes a PUF a tamper-resistance solution.

All of the above show that PUFs provide uniqueness, randomness, unclonability, security, low-cost, and tamper resistance which make them very suitable candidates for robust hardware-based intrinsic security devices. With these advantages, PUFs have been proposed for lightweight device identification and authentication, and cryptographic key generation. Now, PUFs are transforming from research to commercial products as the results of the huge potential of PUFs observed by the industry [13, 14].

1.3 Motivations For Research

In order for a PUF to be usable, the response of the PUF, which later will be used for an identifier or a key must be reproducible and reliable over multiple authentication or key generation processes. However, as the PUF is implemented using CMOS circuit design techniques, it is susceptible to environmental variations such as temperature and supply voltage, and also CMOS ageing effects. Although the environmental variation is a concern, it is a reversible effect. For example, the operating frequency of a CMOS circuit is slower than normal at an elevated temperature, but its normal operating frequency is restored once it is cooled down to its normal operating temperature. On the contrary, the impact of ageing is irreversible over an extended period, leading to a permanent shift in the circuit behaviour. Generally, the effect of CMOS device ageing mechanisms such as Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI) can be manifested as a degradation in the threshold voltage, V_{th} [15]. This reduces the drive currents and leads to a performance (e.g., speed) degradation of the CMOS circuit. When a PUF ages, the response of the PUF is not 100% reliable. As a consequence, there may be two possible errors in the device authentication, either a false negative or a false positive. A false negative means that the response of a PUF deviates significantly from its initial state, and it might be deemed as a different PUF and rejected during an authentication process. On the other hand, a false positive occurs when the response of a PUF deviates and become nearly same as another PUF's response, and it might be accepted during an authentication process as another device. For a cryptographic key generation, inherently noisy PUF response must be corrected with an error correction mechanism before it can be used as a secret key. Due to the impact of ageing, the number of errors in PUF response increases. As a consequence, the complexity (i.e., area) of an error correction code (ECC) to generate error-free cryptographic keys increases. An increase in the area can be seen as a disadvantage for a PUF to be used in resource-constraint security devices.

As discussed in Section 1.2, the mapping between the challenge-response pairs (CRPs) of a given PUF is determined by its process variations. The impact of process variations remains static. Therefore, although it is impossible to physically clone the PUF through the fabrication process, the relationship of its CRPs can be modelled or software-cloned using machine learning (ML) techniques. From all of the above, the two keywords that summarise the open problems are reliability (i.e., due to ageing) and security (i.e., susceptibility to ML-attack). The aim of this project is to study the reliability and security aspects of PUFs with regard to the ageing and susceptibility to ML-attack, respectively, in the context of resource-constrained security devices.

1.4 Objectives

The objectives of the research presented in this thesis are summarised as follows:

- To implement and characterise a PUF using a 65-nm CMOS fabrication technology.
- To investigate the impact of ageing on PUFs and proposed mitigation techniques, if necessary.
- To develop a lightweight technique to increase the robustness of PUFs against ML attacks and to validate the technique using a suitable machine learning algorithm.

1.5 Thesis Structure

Chapter 2 provides an overview of the literature and the essential background related to the work in this thesis. The types of PUFs, related PUFs construction, and PUF quality metrics are discussed. The CMOS ageing process is then described and its potential impact on the reliability of PUFs is discussed. The proposed use of PUFs in cryptographic key generation and low-cost identification and authentication are summarised. The known attacks on PUFs are discussed, including invasive, semi-invasive, and non-invasive attacks. Despite the potential issues in reliability and security, the advantages of PUFs as compared to a conventional method of storing a key in NVM are discussed.

Chapter 3 presents the first contribution of the thesis. We demonstrate that a differential design technique to build a PUF can be a mechanism to mitigate the first-order dependencies of ageing, such as signal probability/duty cycle and supply voltage. A "Two Chooses One" PUF or TCO-PUF is introduced and the circuit-level implementation is detailed using a TSMC 65-nm CMOS technology. The quality metrics of TCO-PUF such as the uniqueness, reliability, and uniformity are analysed. The susceptibility to an ML-attack is further analysed. An evaluation of the impact of ageing on PUF-based differential architectures which include TCO-PUF and Arbiter-PUF (both implemented using a 65-nm CMOS fabrication technology) is carried out. The results show that a differential design technique is desirable for PUF implementation to achieve high reliability under device ageing.

Chapter 4 describes the second contribution, which is a challenge permutation technique to increase the resiliency of an Arbiter-PUF against an ML-attack. An Arbiter-PUF has been implemented using a TSMC 65-nm CMOS technology. An artificial neural network (ANN) has been used to evaluate the vulnerability of the Arbiter-PUF to an ML-attack. The results show that a challenge permutation technique can alter the output transition probability of the Arbiter-PUF, resulting in an increase of the resilience to an ML-attack. Moreover, the results show that a random challenge permutation is required to maximise

the complexity of the output transition probability of the Arbiter-PUF. Hence, a high unpredictability for an Arbiter-PUF can be achieved. A challenge permutation technique can be implemented by routing obfuscation which introduces no extra overhead.

Chapter 5 presents the third contribution. A bit selection technique is proposed to reduce the bit error rates due to ageing in a dual function SRAM used as a memory and PUF, which results in a reduction in the area overhead of the ECC. The effect of ageing on SRAM used as a PUF is not uniform, but is dependent on the patterns of the data stored in the memory. The distribution pattern of a 32-bit ARM instruction cache (i-cache) is found to be predictable as in the impact of ageing. Based on these analyses, a bit selection technique is proposed to select only SRAM cells that have close to a 50% probability of storing a value of '1'. By using a bit selection technique, a reduction in the bit error rates is achieved and the area overhead of the ECC is reduced by 6 times compared to that without a bit selection technique. Meanwhile, the proposed technique requires a negligible area overhead with respect to the reduction in the area overhead of the ECC.

Chapter 6 concludes the findings and contributions of the work in this thesis. Suggestions for future work directions are also provided.

1.6 Publications

A list of publications related to this research is as follows:

1. B. Halak, Y. Hu, and M. S. Mispan, "Area efficient configurable physical unclonable functions for FPGAs identification," in IEEE International Symposium on Circuits and Systems, 2015, pp. 946-949.

2. M. S. Mispan, B. Halak, Z. Chen, and M. Zwolinski, "TCO-PUF : A subthreshold physical unclonable function," in 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), 2015, pp. 105-108.

3. M. S. Mispan, B. Halak, and M. Zwolinski, "NBTI aging evaluation of PUF-based differential architectures," in IEEE International Symposium on On-Line Testing and Robust System Design, 2016, pp. 103-108.

4. B. Halak, M. Zwolinski, and M. S. Mispan, "Overview of PUF-based hardware security solutions for the Internet of Things," in IEEE Midwest Symposium on Circuits and Systems, 2016, pp. 1-4.

5. M. S. Mispan, B. Halak, and M. Zwolinski, "Lightweight obfuscation techniques for modeling attacks resistant PUFs," in IEEE International Verification and Security Workshop, 2017, pp. 19-24.

6. M. S. Mispan, B. Halak, and M. Zwolinski, "Extended abstract: SRAM-PUF based on selective power-up and non-destructive scheme," in International Workshop on Microprocessor/SoC Test and Verification, 2016, pp. 1-2 (**unpublished**).

7. M. S. Mispan, H. Su, and M. Zwolinski, and B. Halak, "Cost-efficient designs for modeling attacks resistant PUFs," in Design, Automation & Test in Europe Conference & Exhibition, 2018, pp. 467-472.

8. M. S. Mispan, S. Duan, B. Halak, and M. Zwolinski, "A reliable PUF in a dual function SRAM," in International Symposium on Power and Timing Modeling, Optimization and Simulation, 2018, pp. 1-6.

Chapter 2

Physical Unclonable Function

This chapter provides a broad overview of the research related to the topic in this thesis. The literature relevant to Chapters 3, 4, and 5 is separately introduced, to describe the contributions in these chapters in the relevant research context. Section 2.1 describes the general definition of a PUF. Section 2.2 explains the process variations in ICs which are exploited by a PUF. The types of PUFs are discussed in Section 2.3. Section 2.4 summarises the existing proposed techniques of silicon PUFs, in which some of the proposed PUFs are used as a case study in Chapter 3, 4, and 5. The metrics to quantify the quality of PUFs are discussed in Section 2.5. Section 2.6 introduces the effect of ageing on PUFs, which are related to the ageing evaluation on differential PUF architecture discussed in Chapter 3 and the proposed ageing mitigation technique described in Chapter 5. In addition, two primary applications of PUFs are discussed in Section 2.7, while Section 2.8 describes the known attacks on PUFs, in which a countermeasure for one of the known attacks is proposed in Chapter 4. Despite the known attacks on PUFs, Section 2.9 describes the advantages of PUFs. Finally, Section 2.10 summarises the literature reviewed in this chapter.

2.1 Definition of a PUF

A PUF is defined as a function that maps challenges to responses and that function is embodied by the physical material of the device [16]. As opposed to a mathematical function which is deterministic in nature and that generates a fixed output for the same input, a function embodied in a physical device is non-deterministic and varies from one instance to another [17]. Based on this notion, in our study, we have focused on silicon PUFs, which exploit the intrinsic and random variations in CMOS devices due to the manufacturing process, as described briefly in Section 1.2. Nevertheless, generally a PUF can be constructed using a non-silicon material, as described later in Section 2.3. For a silicon PUF, the complex statistical variations of devices and interconnects can be

used to map a set of challenges to a set of responses in one instance of a PUF and the mapping changes from one instance to another, stochastically. The set of CRPs for a PUF can be defined as (C_i,R_i), $i = 1, ...N$. Depending on the types of PUFs, however, generally a challenge C can be described as a k-bit input. Challenges are used to control the behaviour of a PUF and based on the challenges applied, corresponding responses are generated. As shown in Figure 2.1, when a challenge is applied to two different PUFs (PUF A and PUF B), the respective responses were produced where Response A $\neq$ Response B.

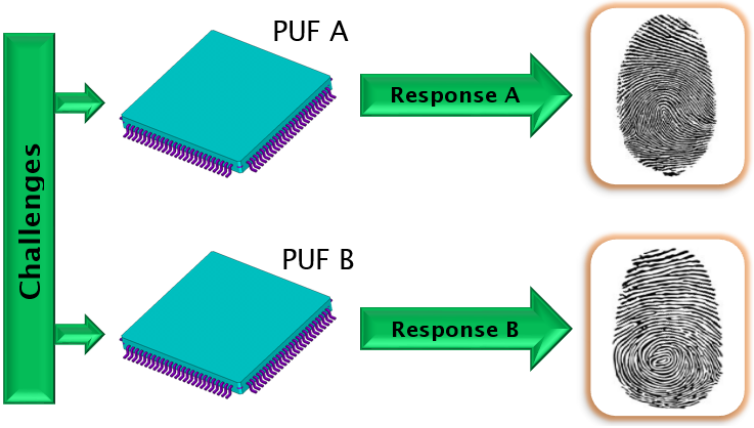


Figure 2.1: Basic functionality of PUF.

A unique response is like an electronic fingerprint which uniquely identifies each PUF. Therefore, there is no need to store a secret key in any memory devices, unlike the conventional method of IC security as discussed in Section 1.1. Instead, a secret key only needs to be generated when it is required, by applying a challenge. Thus, a PUF provides unclonable, random, and secure features, which make it a very promising technology as a replacement for current security solutions.

2.2 Variability in Integrated Circuits

Manufacturing process variation is a fundamental limitation of the control of the device's physical features and interconnects during fabrication [18]. Process variations can be divided into two categories [19]. The first category is inter-die variations where the same device on a die can have different characteristics across various dies. The second category is intra-die variations where similar transistors within a single die can have different characteristics. The aggressive scaling of CMOS technology has led to a drastic increase in process variations such as oxide thickness and random dopant fluctuations (RDF) which causes a direct impact on the electrical behaviour of MOSFETs. One of the fundamental challenges for CMOS device performance is RDF which is caused

by the randomness in the amount and position of dopants during dopant implantation, resulting in a fluctuation of the total number of dopants in the transistor channel [20].

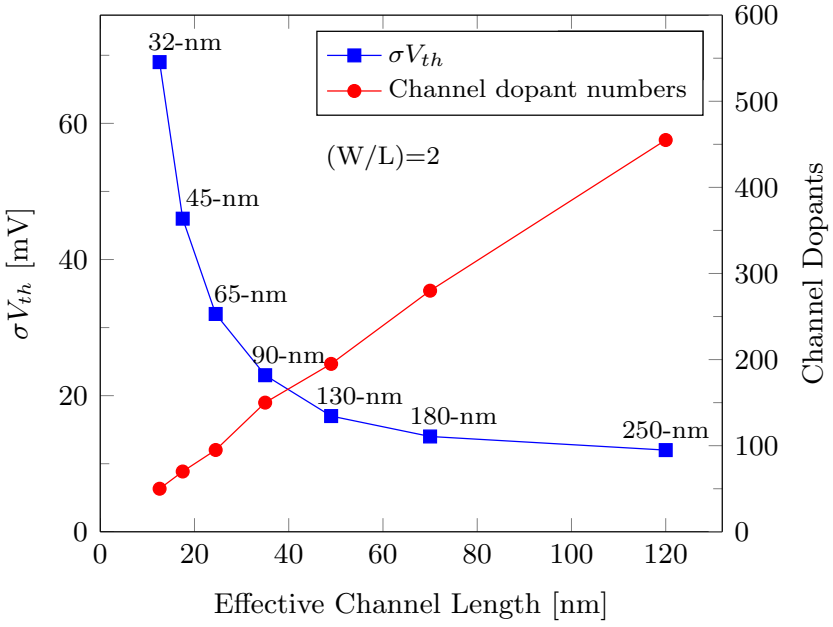


Figure 2.2: Scaling trend of V_{th} variations due to RDF [20].

The V_{th} of MOSFETs is significantly determined by the dopant density in the transistor channel. As devices scale down, the channel volume decreases and the total number of channel dopants decreases. As a result, the relative effect of a single change in dopant number increases and the variations in the V_{th} becomes significant [20]. The effect of RDF on device scaling from 250-nm to 32-nm technology node can be seen in Figure 2.2. Although process variation is an unwanted effect for CMOS circuitry, it is the desired effect for PUFs.

2.3 Types of PUFs

2.3.1 Non-Silicon and Silicon PUFs

The idea of a PUF was introduced when Pappu *et al.* proposed the concept of physical one-way functions (POWFs) which are based on an optical principle of operation [21]. The speckle patterns resulted from applying laser lights (at the different angle, distance, and wavelength) on a transparent optical medium, which contains scattering particles, are found to be unique and unpredictable. Following Pappu's works, the concept of a silicon PUF was first introduced [16]. In this study, Gassend *et al.* argued that a complex IC can be viewed as a silicon PUF and described a technique to identify

and authenticate individual ICs. Further, the PUF was realized in a real silicon and called an Arbiter-PUF [12]. The Arbiter-PUF exploits the delay mismatch between two nominally identical delay paths. Another delay-based PUF called the Ring Oscillator-PUF (RO-PUF) was proposed in [22], where the output response is generated based on the frequency mismatch between a pair of ring oscillators (ROs). In [23], a memory-based PUF, which is based on the random start-up values (SUVs) of SRAM cells was proposed. The power-up SRAM state as an identifying fingerprint was also concurrently proposed in [24]. A PUF based on cross-coupled latches was proposed in [25], namely the Butterfly-PUF, targeted at protecting intellectual property (IP) designs in Field-Programmable Gate Arrays (FPGAs). Another memory-based PUF was proposed in [26] which uses a known cell structure of data bus keeper or data bus holder (i.e., cross-coupled inverters) as a PUF, namely the Buskeeper-PUF.

2.3.2 Strong, Weak and Controlled PUFs

Silicon PUFs can be categorised into three sub-types according to the security properties of their challenge-response behaviours, each with their own preferred applications. Three established types are the Strong PUFs [23], the Weak PUFs [23], and the Controlled PUFs [27].

1. Strong PUFs: Strong PUFs are PUFs with a very large number of CRPs (C_i,R_i), $i = 1, ...N$ [23]. The number of CRPs of the considered PUFs grows exponentially as the number of bit challenges increases. The challenge-response interface is directly accessible without a protection mechanism in which the CRPs can be collected using a non-invasive CRPs measurement. Rührmair *et al.*, [28] refined the Strong PUF definition in which it must also be infeasible to be numerically modelled with a high prediction accuracy based on the observed CRPs (i.e., the PUF response is unpredictable).

2. Controlled PUFs: Controlled PUFs are improved Strong PUFs where the challenge-response interface is not directly accessible but it is protected by a logic processing unit using techniques such as random hash function, permutation, obfuscation and etc. Gassend *et al.*, [27] used random hash function technique for the pre-processing of challenges before being input to the Strong PUF. In a similar way, the responses of the Strong PUF are post-processed by the random hash function before being output by the Controlled PUF.

   A model-building attack is one of the plausible attacks on strong PUFs [28, 27], where the adversary builds a numerical model of the PUF by measuring a number of CRPs. The introduction of an extra pre and post-processing steps for the Controlled PUFs increases the level of difficulty to measure and collect the CRPs. Hence, this reduces the vulnerability to a model-building attack [27].

3. Weak PUFs: Weak PUFs are PUFs with a very small number of CRPs, fixed challenges, and in the extreme case with just only a single challenge [23, 28].

Based on the above definition, the main distinction between Strong and Weak PUFs is the number of generated CRPs. As Strong-PUFs can support a large number of CRPs, they can provide authentication capabilities, particularly using a challenge-response protocol without having to store a secret key as a unique identifier. For Weak-PUFs, a limited number of CRPs meant that these CRPs must be kept secret. For this reason, Weak PUFs are well suited for a secret key generation for any cryptographic process such as encryption/decryption and message authentication code (MAC). The preferred applications based on the categorisation above are further discussed later in Section 2.7.

2.4 Silicon PUF Constructions

Since the first idea of silicon PUF, as been described in [16], which was motivated by the Pappu's works [29], an enormous number of PUF techniques have been proposed in the past decade. In this section, silicon PUFs are classified based on their construction and operating principles:

1. The first class of construction consists of delay-based PUFs, which are often constructed from simple digital circuit structures and exploit the intrinsic variations in the logic gate and interconnect delays to produce device-specific random signatures. Delay-based PUFs include Arbiter-PUFs [12, 30, 22, 31] and RO-PUFs [16, 22].

2. The second class of construction is memory-based PUFs, which exploit intrinsic variations in bi-stable memory elements such as SRAM PUFs [23, 24], D Flip-flop PUF [32], Butterfly PUF [25], SR-NOR latch PUF [33], and Buskeeper PUF [26].

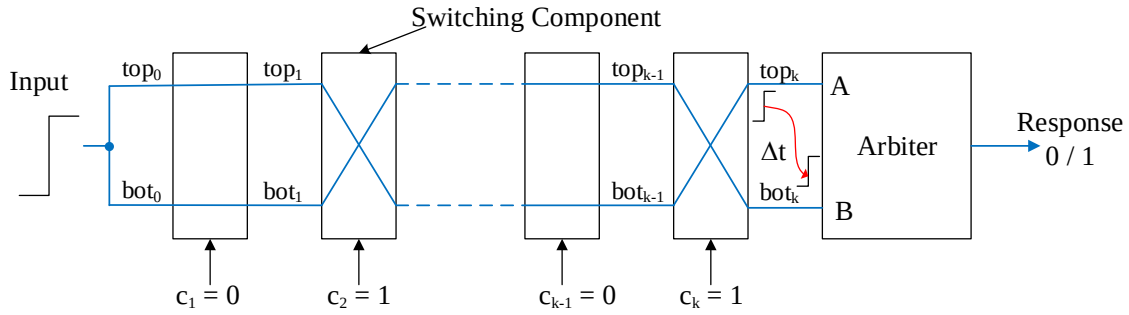
3. The third class of construction is mixed-signal PUFs in which the PUF behaviour is inherently of an analogue nature and require analogue-to-digital converter (ADC) to digitise the measured analogue responses. A mixed-signal PUF typically exploits the variability of minimum size MOSFETs to enhance the threshold voltage mismatches, such as in the Voltage Transfer Characteristic PUF (VTC-PUF) [34] and Current Mirror-PUF [35].

4. The final class of construction is PUF built from emerging nanotechnology devices in order to further achieve secure, robust, and lightweight PUF designs [11].

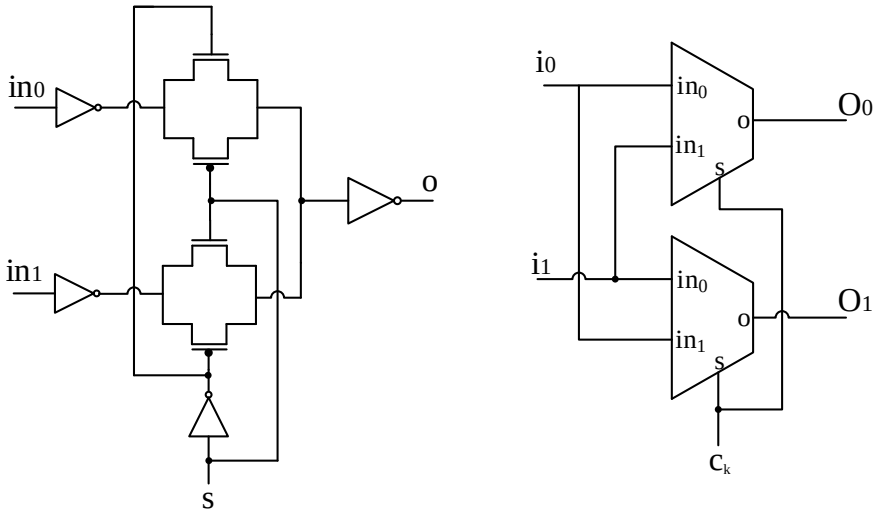
The following sections discuss in detail the construction of each PUF that are mentioned above. Though this is not a comprehensive discussion on different PUFs proposed so far, it gives an overview of PUF development in general.

2.4.1 Arbiter-PUF

Lee *et al.* presented the first PUF which was fabricated on silicon using a TSMC 180-nm CMOS technology, namely an Arbiter-PUF [12]. This PUF circuit exploits the logic gate delays and interconnect variations which are affected by the process variations. The Arbiter-PUF consists of k stages or switching component where each stage is composed of two 2-to-1 multiplexers as shown in Figure 2.3 [36]. A rising pulse at an input propagates through two nominally identical delay paths. The paths for the input pulse are controlled by the switching elements, which are set by the bits of the challenge, $C = \{c_1, c_0..., c_k\}$. For $c_k = 0$, the paths go straight through, while for $c_k = 1$, they are crossed. Because of manufacturing variations, there is a delay difference of $\triangle t$ between the paths. An arbiter at the end generates a random response, '0' or '1', depending on the difference in arrival times.



(a) k-bit Arbiter-PUF



(b) Switching component

Figure 2.3: The construction of Arbiter-PUF as proposed in [36].

Ideally, an Arbiter-PUF response is equally likely to be '0' or '1' in which it is solely dependent on the randomness in process variations. However, the response could be

biased to be either '0' or '1'. A large bias reduces the uniqueness and randomness of Arbiter-PUF responses. To avoid the bias, two important criteria must be fulfilled at the design stage:

1. The highest degree of symmetry in the layout is required to ensure the unbiased response of the Arbiter-PUF. The symmetry routing includes the routing before the first switching element, inside the switching element, in between stages, before and inside the arbiter circuit. Although it is non-trivial, it is possible to achieve a symmetrical routing in application-specific integrated circuits (ASICs) [12], while implementations on FPGAs seem to be difficult due to the placement and routing constraints of the general FPGA hardware architecture [37].

2. An unbiased arbiter circuit must be used for digitisation of the delay difference. According to [38], fair arbitration can be achieved by using an SR-latch which has a symmetric circuit topology.

For an Arbiter-PUF, although the effect of process variations on the logic gates and interconnect delays is random, there is a small yet non-negligible probability that metastability could happen if the rising edges at the inputs A and B of the arbiter circuit have a very small timing difference. A fair arbiter circuit such as an SR-latch which has a symmetric circuit topology is desired to minimise the arbiter bias and the metastability effects [39].

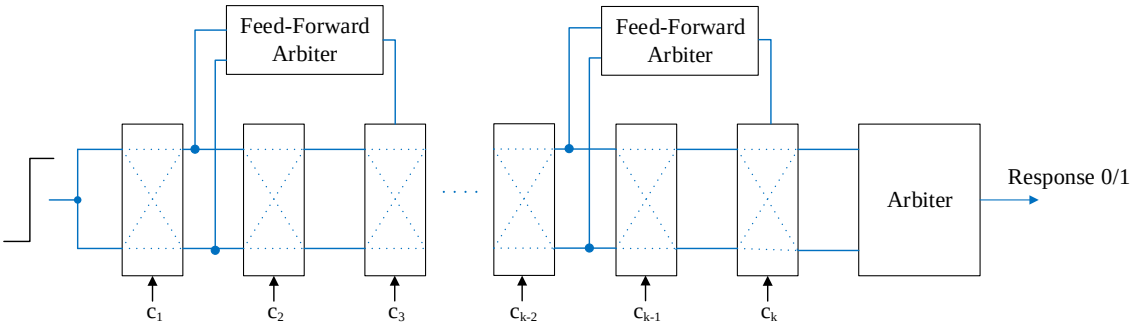


Figure 2.4: The construction of Feed-Forward Arbiter-PUF as proposed in [30].

The construction of an Arbiter-PUF, as in Figure 2.3, is based on additive delays caused by the individual switching components which have linear characteristics [36]. Therefore, the complexity of the CRPs mapping is minimal which enables the model building attacks using ML techniques. A few derivatives of the Arbiter-PUF have been proposed with the aim in introducing the non-linearity into it. Lim *et al.*, [30] presented the Feed-Forward Arbiter-PUF which uses the output of intermediate arbiters to configure the subsequent switching components as depicted in Figure 2.4. However, the intermediate arbiters increase the probability of metastability states, which would result in more errors in the PUF response.

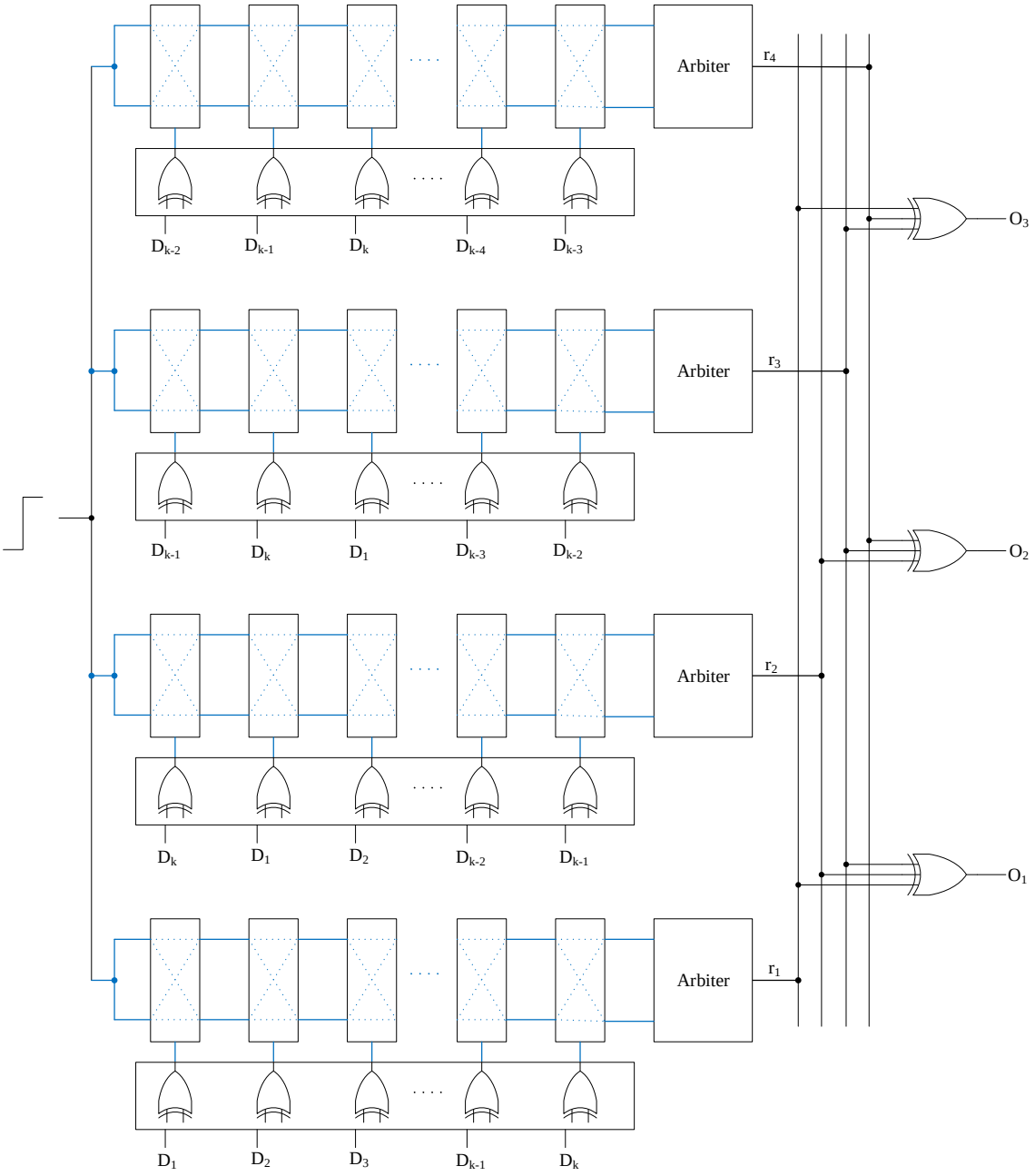


Figure 2.5: The construction of Lightweight-PUF as proposed in [31] for l=4.

Elsewhere, Majzoobi *et al.*, [31] proposed a Lightweight-PUF which consists of l parallel Arbiter-PUFs. The outputs of the Arbiter-PUFs are XORed to generate an overall PUF response O_1,O_2, and O_3 as shown in Figure 2.5. The input XOR network in the Lightweight-PUF is expected to further increase the non-linearity in the CRPs mapping. Earlier, l-XOR Arbiter-PUF was proposed in [22] which consists of l parallel Arbiter-PUFs as illustrated in Figure 2.6. The susceptibility of the Arbiter-PUF and its derivatives to ML-attacks will be discussed in detail in Section 2.8.3.

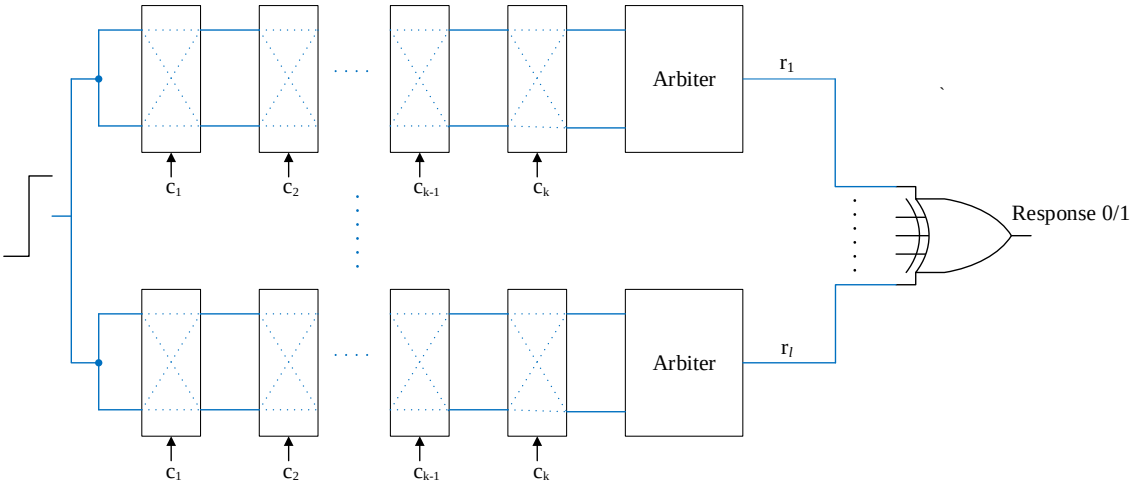


Figure 2.6: The construction of the l-XOR Arbiter-PUF as proposed in [22].

2.4.2 Ring Oscillator PUF

Another type of delay-based PUF is the ring oscillator PUF (RO-PUF). The working principle of the RO-PUF is that by measuring the frequencies of digital oscillating circuits that randomly vary due to the uncontrollable effects of silicon process variations on the logic gates and interconnect delays. The notion of a PUF which uses an oscillating circuit was first proposed by Gassend *et al.*, [16]. Further, Suh *et al.*, [22] proposed an RO-PUF architecture which consists of an array of n nominally identical ROs. The architecture also contains two frequency counters to count the number of rising edges at the selected RO pairs. Two n-to-1 multiplexers control which pair of ROs is currently applied to both counters. Hence, the selection signals of these multiplexers become the PUF's challenge. The construction of an RO-PUF is depicted in Figure 2.7. Both frequency counters are enabled for a fixed time interval and the resulting counter values are compared to generate a response, '0' or '1', based on which oscillator from the selected RO pair is faster. Since the process variations affect the frequencies of the n RO arrays, the resulting comparison bit will be random and device-specific.

Each comparison of a pair of oscillators generates a bit. Therefore, given an array of n oscillators, a total of $\binom{n}{2} = \frac{n(n-1)}{2}$ pairs can be compared. However, the number of independent bits that can be produced is less than $\frac{n(n-1)}{2}$. For example, if oscillator A is faster than B and B is faster than C, clearly that A will also be faster than C which results in a correlated pair-wise comparison. Suh and Devadas [22] concluded that the maximum number of uncorrelated comparisons is limited by the number of possible orderings in which the oscillators can be ordered. Given n oscillator frequencies which are independent and equally distributed, there exist $n!$ equally likely possible orderings. Hence, the maximum entropy (i.e., uncorrelated pair-wise comparisons) of an RO-PUF is $\log_2 n!$. It is also possible to avoid any correlation by simply comparing a pair of ROs once, resulting in only $\frac{n}{2}$ response bits.

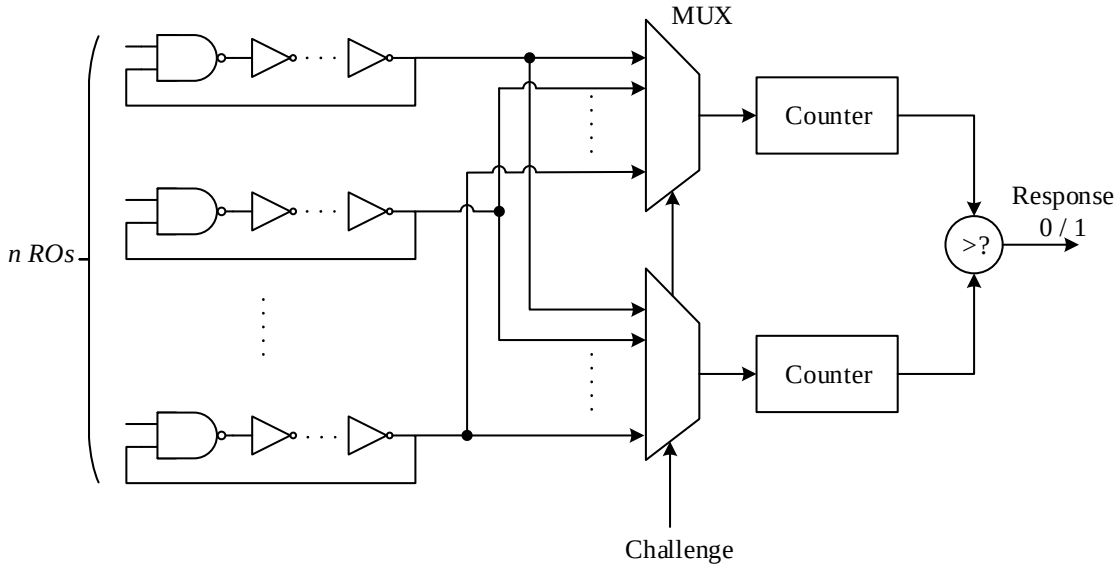


Figure 2.7: The construction of an RO-PUF as proposed in [22].

2.4.3 SRAM-PUF

The previously discussed PUFs are all delay-based PUFs which require additional circuitry to use them as on-chip hardware security. Another idea is to use on-chip resources as a PUF, particularly using SRAM, which is available in any computing system [23, 24]. Guajardo *et al.*, [23] exploited SRAM memory as an intrinsic PUF for use as a secret key generator for FPGA bitstream encryption/decryption to support an IP protection. Holcomb *et al.* [24], targeted wider applications of SRAM as PUFs to support resource-constrained security-critical applications such as contactless credit cards etc. Generally, SRAM memory is constructed based on rows and columns of bit cells. The number of available bit cells in an SRAM represents its storage size. Each bit cell of SRAM is typically a six-transistor CMOS circuit which consists of two cross-coupled inverters (MP1, MP2, MN1, and MN2) and two access transistors (MN3 and MN4), as illustrated in Figure 2.8.

The transistors forming the cross-coupled inverters are also known as a bi-stable element, in which each node Q and QB can be in either of two states, '0' or '1'. The power-up or SUVs of bi-stable elements depend on the device mismatches, which result from process variations. During power-up of an SRAM cell, as the supply voltage increases, the current flowing through MN1 and MN2 will slowly pull up the voltage at nodes Q and QB. Because of the random process variations, transistor MP2 has a slightly higher threshold voltage compared to that of MP1. Therefore, the current that flows through MN1 is slightly higher than through MN2, thus turning ON the MN2 and pulling down node QB to GND. At the same time when node QB is discharging, MP1 is turned ON and pulls up node Q to V_{dd}. As shown in Figure 2.9, the nodes Q and QB settle at '0'

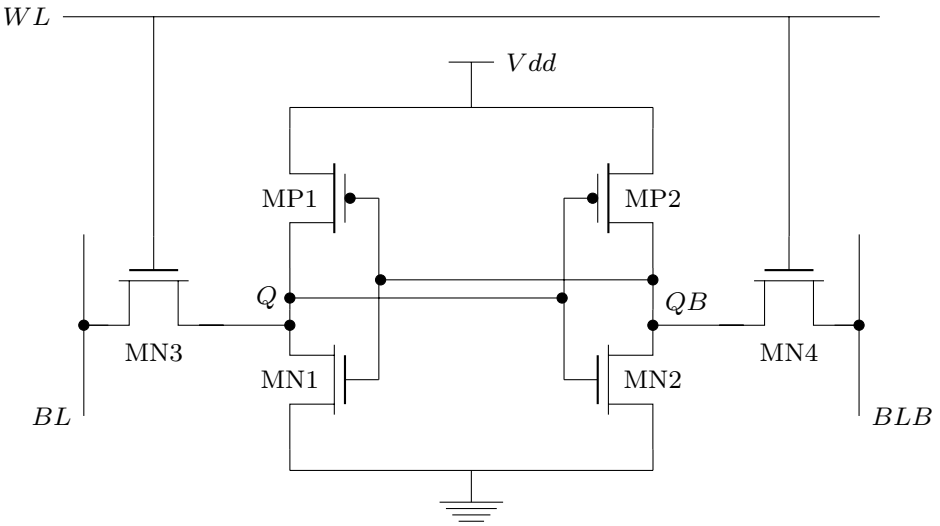


Figure 2.8: 6-T SRAM cell circuit.

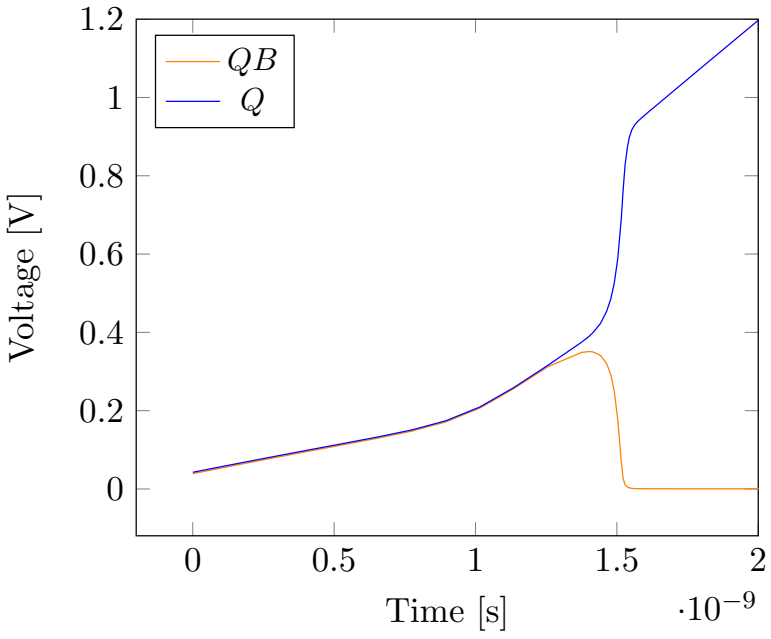


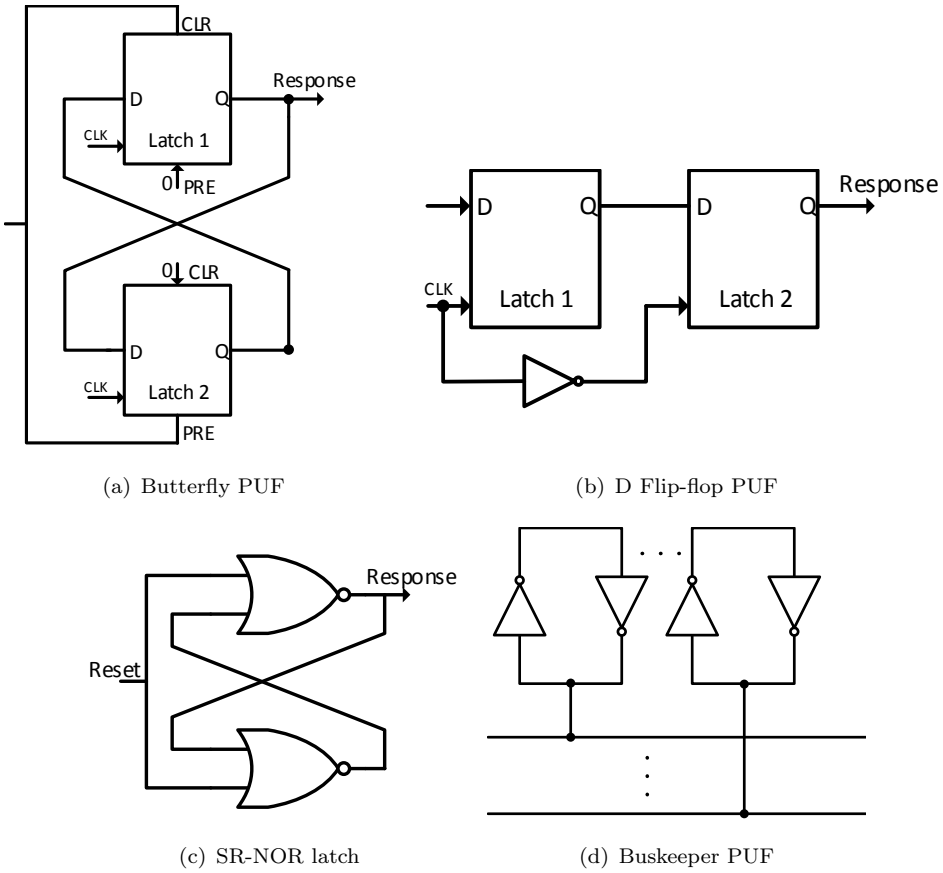
Figure 2.9: Bi-stable SRAM internal nodes, Q and QB resolving to '1' and '0' during power-up process.

and '1', respectively. When powering-up the SRAM, the SUVs across different memory blocks within an SRAM and across multiple SRAMs show device-specific and random patterns, which are the desired qualities to be used as a PUF [23, 24].

2.4.4 Flip-flop, Latch and Buskeeper PUFs

PUFs based on flip-flops and latches are derived from the same working principles and physical effects as SRAM-PUFs. Kumar *et al.*, [25] proposed a Butterfly PUF which consists of a pair of cross-coupled latches, forming a bi-stable circuit as illustrated in Figure 2.10(a). When the preset (PRE) and clear (CLR) signals are simultaneously set to high, the Butterfly PUF can be forced into an unstable state and converges into a stable state when both signals are set low after a few clock cycles. The Butterfly PUF was proposed to overcome the disadvantage of the SRAM-PUF on FPGA platforms, whereby the SRAM on most commercial FPGAs is cleared after power-up. However, the Butterfly PUF requires symmetrical routing to minimise the impact of design mismatch which is not trivial to achieve due to the routing constraints on FPGAs [37]. Elsewhere, a PUF based on the power-up behaviour (i.e., similar to the SRAM PUF) of clocked D flip-flops is proposed by van der Leest *et al.*, [32]. Most D flip-flops are based on two latches as depicted in Figure 2.10(b). The advantage of D Flip-flop PUF is that the location of the individual flip-flop can be randomly spread across a design and their signal lines connecting them to the read-out circuitry can be obfuscated which increases the defence against invasive attacks such as probing attacks.

Su *et al.*, [33] proposed a device identification technique based on the two cross-coupled NOR-gates which constitute a simple SR-NOR latch as shown in Figure 2.10(c). When the reset signal is high, the SR-NOR latch enters an undefined state and converges to a stable state depending on the internal mismatch between the NOR gates when the reset signal is low. The SR-NOR latch uses the same working principle as SRAM-PUF, however does not rely on a power-up state that depends on the supply voltage. Therefore, it provides more flexibility because it can generate unpredictable and reproducible responses when the reset signal is (re)asserted any time that the secret key is required during the on-time of the device. Another variant of latch-based PUF is the Buskeeper PUF, proposed in [26], which is constructed using bus keeper cells. The basic structure of a bus keeper cell is a cross-coupled inverter with a weak drive-strength, connected to a bus line, as shown in Figure 2.10(d). A bus keeper cell is used to maintain the last driven state on the bus line and prevents the bus line from floating. Similar to SRAM cells, the power-up state of bus keeper cells is determined by the device mismatches which result from process variations. An advantage of the Buskeeper PUF is the low area overhead in comparison to other types of memory based-PUFs such as D Flip-flop PUFs.



(a) Butterfly PUF

(b) D Flip-flop PUF

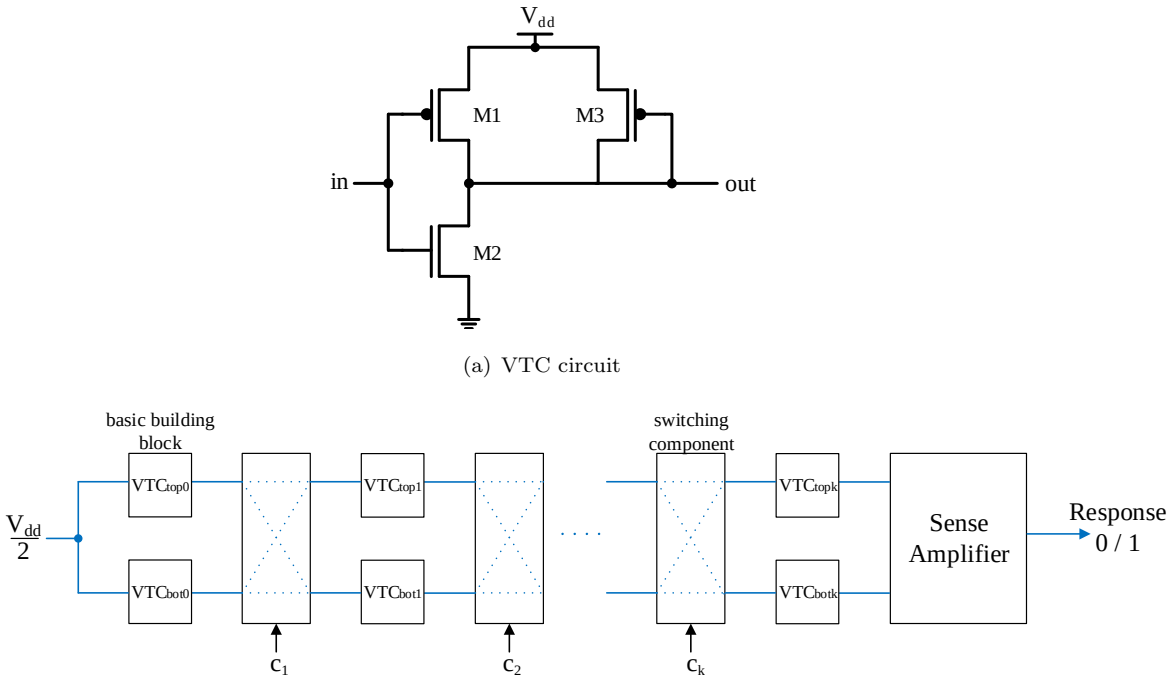
(c) SR-NOR latch

(d) Buskeeper PUF

Figure 2.10: The construction of Butterfly PUF [25], D Flip-flop PUF [32], SR-NOR latch PUF [33] and Buskeeper PUF [26].

2.4.5 Mixed-Signal PUFs

A PUF with a mixed-signal structure, namely the VTC-PUF, was proposed by Vijayakumar *et al.*, [34], which exploits the variability in a voltage transfer characteristic (VTC) circuit as shown in Figure 2.11(a). The VTC circuit has a non-linear relationship between input and output voltages in which the non-linearity is controlled by the feedback transistor, M3. Adapted from the architecture of Arbiter-PUF, the VTC circuit is used as a basic building block in which it is coupled with a switching component in each stage, creating a cascaded circuit as depicted in Figure 2.11(b). The switching components are created from a simple transmission gate based circuit. The input node is connected to $\frac{V_{dd}}{2}$. A voltage sense amplifier is used to sense the difference of the propagated input voltages and generate a response, '1' or '0', based on the sign of the differential output at the final stage.



(a) VTC circuit

(b) Top level of VTC-PUF

Figure 2.11: The construction of VTC-PUF [34].

Kumar *et al.*, [35] proposed a Current Mirror-PUF which is also adapted from the Arbiter-PUF architecture, as in Figure 2.11. However, the basic building block is the current mirror circuit illustrated in Figure 2.12. I_{out} is the mirrored current of I_{ref} which will be varied slightly due to the process variations experienced by the transistors, M1-M4. Similar to the VTC-PUF, the switching components are created by a simple transmission gate based circuit, but the input is connected to a constant current source. A current sense amplifier is used to sense the difference of the propagated input currents and generate a response, '1' or '0', based on the sign of the differential output at the final stage.

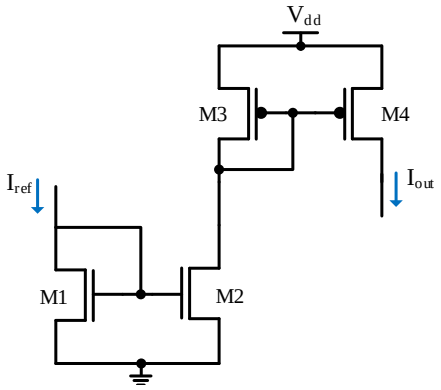


Figure 2.12: The construction of the current mirror circuit in the Current Mirror-PUF [35].

Elsewhere, Cao *et al.*, [40] proposed a mixed-signal PUF based on a CMOS image sensor circuit. The image sensor function is not affected when operated in a PUF mode. The core of a CMOS image sensor is a pixel array, where its typical structure is depicted in Figure 2.13. This PUF exploits the variations of the pixel output voltage during the reset phase which varies due to the variations in the threshold voltages of transistors M_{RS} and M_{SF}. The two reset voltages of two pixels are compared to generate a response, '0' or '1', based on which reset voltage is larger. A predefined threshold is introduced during the comparison of reset voltages which effectively increases the reliability of the response by about 10.8%.

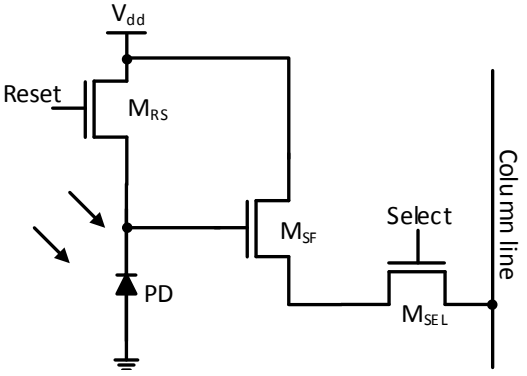


Figure 2.13: 3-T pixel circuit.

Saha *et al.*, [41] proposed a fast and lightweight mixed-signal PUF which exploits the susceptibility of the threshold voltage of MOSFETs to process variations, hence, the name "Threshold Voltage PUF" or TV-PUF. The core circuit in the TV-PUF is a block consisting of n cascaded transistors as depicted in Figure 2.14. When V_{IN}=V_{dd}, the output voltage at the source terminal of the cascaded series of transistors becomes $V_{out} = V_{dd} - V_{th1} - V_{th2} - ..V_{thn}$. Clearly, the output voltage depends on the threshold voltages of n cascaded transistors which are susceptible to random process variations. Due to this, the output voltages of two nominally identical blocks are different and are compared using the sense amplifier to generate a 1-bit response. A decoder circuit is used to set the level of V_{IN} to $HIGH$ (V_{dd}). For a unique challenge (i.e., an input decoder), only one decoder output is $HIGH$, which is connected to two blocks of n cascaded transistors.

2.4.6 Emerging Nanotechnology-based PUFs

All of the above PUF designs focus on exploiting process variations intrinsic to CMOS technology. Recently, nanotechnology-based PUFs have made some progress since the scaling down to the nano-region has resulted in an increased variability in nanoelectronic devices. PUFs that have been proposed include the Carbon-nanotube field-effect transistors (CNFET) based PUF [42], the phase change memory (PCM) based PUF [43],

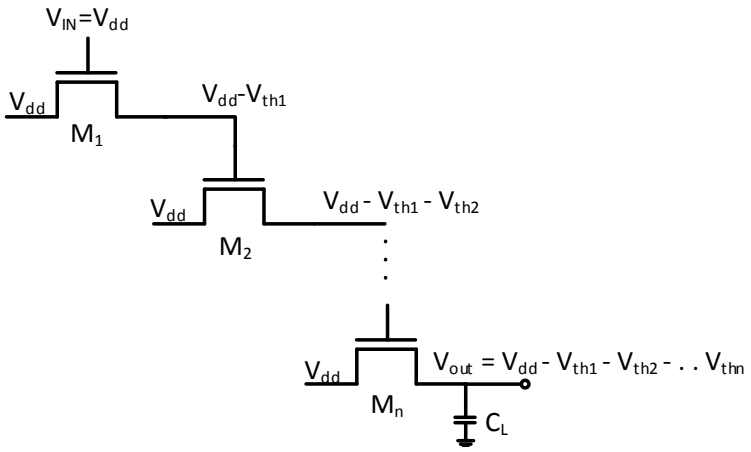


Figure 2.14: The construction of n cascaded transistors of one block in TV-PUF [41]

and the Memristor-based PUF [44]. Nanotechnology-based PUFs offer a few advantages over CMOS based PUFs, such as substantial process variations, small foot-prints, and lower energy consumption [11]. Since this study focuses on CMOS silicon-based PUFs, one may refer to [11] for a comprehensive discussion on PUFs based on emerging nanotechnology devices.

2.5 PUF Quality Metrics

All of the above discussed the proposed techniques for different types of PUFs implementations. Several parameters were defined to quantify the performance of the proposed PUFs, but the parameter or evaluation criteria differ for each implementation. Therefore, a standard set of parameters is needed to quantify the performance of these PUFs. Maiti *et al.*, [45] has systematically refined the quality parameters to evaluate and compare the performance of PUFs. These parameters are uniqueness, reliability, and uniformity. Next, the definition of these parameters are described and later in this section, the quality metrics for PUFs discussed in Section 2.4 are summarised in a performance table.

2.5.1 Uniqueness

Uniqueness is the ability of one PUF instance to be uniquely distinguished from another PUF. The Hamming Distance (HD) is used to evaluate the uniqueness performance and is called the 'Inter-HD'. The HD between two binary strings of equal length is the number of positions at which corresponding bits are different. If the same challenge C is applied to two chips, i and j ($i \neq j$), and n-bit responses are generated, $R_i(n)$ and $R_j(n)$, respectively, the average Inter-HD among k chips is defined as [45]:

$$\text{Inter–HD} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\text{HD}(R_i(n), R_j(n))}{n} \times 100\% \tag{2.1}$$

To achieve a high probability of uniquely identifying a PUF from a group of PUFs of a similar type, it is desirable to have truly random PUF responses, where the Inter-HD is centred around 50%.

2.5.2 Reliability

The reliability of the PUF determines the consistency of the responses, given the same challenge at different ambient temperatures and/or supply voltage fluctuations. The HD is used to evaluate the reliability and is called the 'Intra-HD'. For the challenge C, a single chip, represented as i, has an n-bit reference response $R_i(n)$ at room temperature and a nominal supply voltage (i.e., the reference condition). The same challenge C is applied to the chip i at different condition to obtain the n-bit response, $R'_{i,j}(n)$. Hence, the average Intra-HD for m samples is defined as, [45]:

$$\text{Intra–HD} = \frac{1}{m} \sum_{j=1}^{m} \frac{\text{HD}(R_i(n), R'_{i,j}(n))}{n} \times 100\% \tag{2.2}$$

From the Intra-HD value, the reliability of a PUF can be defined as:

$$\text{Reliability} = 100\% - \text{Intra–HD} \tag{2.3}$$

From Eq. (2.3), a small Intra-HD is desired to achieve high reliability.

2.5.3 Uniformity

The uniformity of the PUF is the proportion of 0's and 1's in the response bits of a PUF. In other words, the uniformity characterises the randomness of the PUF's response and the ideal value of uniformity is 50%. The Hamming Weight (HW) is used to evaluate the uniformity which measures number '1' bits in the binary sequence, and is described as below, [45]:

$$(\text{Uniformity})_i = \frac{1}{n} \sum_{j=1}^{n} r_{i,j} \times 100\% \tag{2.4}$$

where $r_{i,j}$ is the j-th binary bit of an n-bit response from a chip i.

Uniformity and uniqueness quality metrics as discussed above are independent parameters. For k chips of a similar type of PUF with an n-bit response from each chip, the average uniformity which is $\frac{1}{k}\sum_{i=1}^{k}(\text{Uniformity})_i$, can be close to an ideal value of 50% but that does not guarantee 50% uniqueness. For example, k chips of the worst PUF could generate k similar n-bit responses which have a balanced distribution of 0's and 1's in their n-bit responses. On the other hand, k chips of a similar type of PUF with an n-bit response from each chip can achieve a uniqueness close to an ideal value of 50% but the average uniformity is not necessarily at 50%. For example, one or more of the k chips of the worst PUF could generate all 1's or 0's in their corresponding n-bit responses.

All of the above quality metrics for a number of silicon PUFs discussed in Section 2.4 are summarised in Table 2.1.

Table 2.1: Performance of surveyed silicon PUF constructions in Section 2.4

PUF Class	PUF Type	Sim/FPGA/ Silicon	Technology	# PUF Instances	Nominal Condition	Uniqueness (%)	Reliability Condition	Worst case bit error (%)	Uniformity (%)
Delay-based	Arbiter-PUF [12]	Silicon	180-nm	37	V_{dd}: 1.8V T_{nom}: 27°C	23	V_{dd}: 1.8V ±2% T_{env}: 27°C to 67°C	4.8	NR
	Arbiter-PUF [38]	Silicon	45-nm	40	V_{dd}: 1.0V T_{nom}: 25°C	38.9	V_{dd}: 1.0V ±10% T_{env}: 25°C to 75°C	9	NR
	RO-PUF [22]	Xilinx Virtex4 FPGA	90-nm	15	V_{dd}: 1.2V T_{nom}: 20°C	46.15	V_{dd}: 1.2V ±10% T_{env}: 20°C to 120°C	0.48 [1]	NR
	RO-PUF [46]	Simulation	90-nm	100	V_{dd}: 1.2V T_{nom}: 25°C	44	V_{dd}: 0.8V to 1.4V T_{env}: 0°C to 100°C	5.52	NR
Memory-based	SRAM-PUF [23]	FPGA	NR	17	V_{dd}: NR T_{nom}: 20°C	49.97	T_{env}: -20°C to 80°C	12	NR
	SRAM-PUF [47]	Silicon	90-nm	68	V_{dd}: 1.2V T_{nom}: 20°C	≈ 50	V_{dd}: 1.2V ±10% T_{env}: -40°C to 80°C	19	NR
	SRAM-PUF [39]	Silicon	65-nm	4	V_{dd}: 1.2V T_{nom}: 25°C	49.7	T_{env}: -40°C to 85°C	8	NR
	Buskeeper PUF [26]	Silicon	65-nm	192	V_{dd}: 1.2V T_{nom}: 25°C	49.02	V_{dd}: 1.2V ±10% T_{env}: -40°C to 85°C	20	NR
	Butterfly PUF [25]	Xilinx Virtex5 FPGA	NR	36	V_{dd}: NR T_{nom}: 20°C	≈ 50	T_{env}: -20°C to 80°C	6	NR
	D Flip flop PUF [32]	Silicon	130-nm	40	V_{dd}: NR T_{nom}: 20°C	36	T_{env}: -40°C to 80°C	13	≈ 75

[1] 1-out-of-8 mask scheme is used to for reliability enhancement.
NR=not reported.

Continued from previous page

PUF Class	PUF Type	Sim/FPGA/ Silicon	Technology	# PUF Instances	Nominal Condition	Uniqueness (%)	Reliability Condition	Worst case bit error (%)	Uniformity (%)
Memory-based	SR-NOR Latch PUF [33]	Silicon	130-nm	19	V_{dd}: 1.0V T_{nom}: 20°C	50.13	V_{dd}: 0.9V to 1.2V T_{env}: 0°C to 80°C	5.47	50.16
Mixed-signal	VTC-PUF [34]	Simulation	45-nm	100	V_{dd}: 1.0V T_{nom}: 25°C	49.80	V_{dd}: 1.0V ±10% T_{env}: 0°C to 85°C	2.1	50.10
	Current Mirror-PUF [35]	Simulation	32-nm	1000	V_{dd}: 1.1V T_{nom}:25°C	49	V_{dd}: 1.0V ±0.1V T_{env}: 0°C to 75°C	2	47
	CMOS Image Sensor PUF [40]	Silicon	180-nm	5	V_{dd}: 3.3V T_{nom}: 25°C	49.37	V_{dd}: 3V to 3.6V T_{env}: 15°C to 115°C	12	NR
	TV-PUF [41]	Simulation	45-nm	100	V_{dd}: 1.0V T_{nom}: NR	50.02	V_{dd}: 1.0V ±20% T_{env}: -55°C to 125°C	4	49.7
Nanotechnology	CNPUF [42]	Simulation	14-nm	100	V_{dd}: 0.8V T_{nom}: NR	49.67	V_{dd}: 0.8V ±22.5% T_{env}: -20°C to 80°C	3.5	NR
	mrPUF [44]	Simulation	90-nm	100	V_{dd}: 1.0V T_{nom}: 27°C	50.17	V_{dd}: 1.0V ±20% T_{env}: -20°C to 85°C	4.4	50.34
	PCM-based rPUF [43]	Silicon	180-nm	10	V_{dd}: NR T_{nom}: NR	≈ 50	T_{env}: 25°C to 85°C	9.7	NR

NR=not reported.

2.6 The Effect of Ageing on PUFs

This section describes the CMOS device ageing mechanism. From our perspective, the goal is to gain a concise understanding of the ageing mechanism which then allows us to understand its impact on circuit level performance. Later in this section, a previous ageing study of PUFs is discussed.

2.6.1 CMOS Device Ageing

In Section 2.5.2, the reliability metric was discussed, wherein the variations in operating temperature and/or supply voltage affect the reproducibility of PUF response. The errors in PUF responses introduced by these variations are considered as reversible temporal variabilities in which the errors may disappear if the cause of variation is withdrawn [48]. Nevertheless, there is another factor that can affect the reliability of a PUF's response, in particular, CMOS device ageing. Ageing causes irreversible changes in circuit behaviour over an extended period, and thus leads to a permanent reliability issue for a PUF [48]. If a CMOS device is used continuously, the device suffers from ageing processes such as BTI and HCI. BTI is more dependent on the vertical electrical field whereas HCI is more influenced by the lateral electrical field [15]. With CMOS technology scaling, the distance between the drain and source junctions is reduced, thus the impact of lateral and/or vertical electric fields is more significant. BTI affects nMOS as Positive Bias Temperature Instability (PBTI) and pMOS as Negative Bias Temperature Instability (NBTI). The ageing investigation of PUFs which is described in Chapter 3 and 5 addresses the NBTI ageing mechanism, as that is more pronounced in low-k or $SiON$ dielectric materials [49].

According to the trapping/de-trapping (T-D) theory, when a pMOS device is negatively biased, the amount of trapped charges in the gate dielectric increases and leads to NBTI stress [50, 15]. When the stress is relieved such that the bias to a pMOS device is relaxed, the trapped charges may be emitted, resulting in NBTI recovery [15]. NBTI manifests itself as an increase in the V_{th} over time while the pMOS device is continuously used. The V_{th} shift is a strong function of the duty cycle or stress time, supply voltage and temperature [15, 46]. Figure 2.15 shows the effect of NBTI for a CMOS inverter at different duty cycles, simulated by HSPICE MOSRA using a low-k TSMC 65-nm technology at a nominal supply voltage, 1.2V, and an ambient temperature of 25^{o}C. In Figure 2.15, the duty cycle is referred to the time when the pMOS device is under negative biased (i.e., V_{gs}=$-V_{dd}$). V_{gs} is the gate-to-source voltage and V_{dd} is the supply voltage. A duty cycle close to 1 implies a larger amount of stress and a smaller amount of recovery.

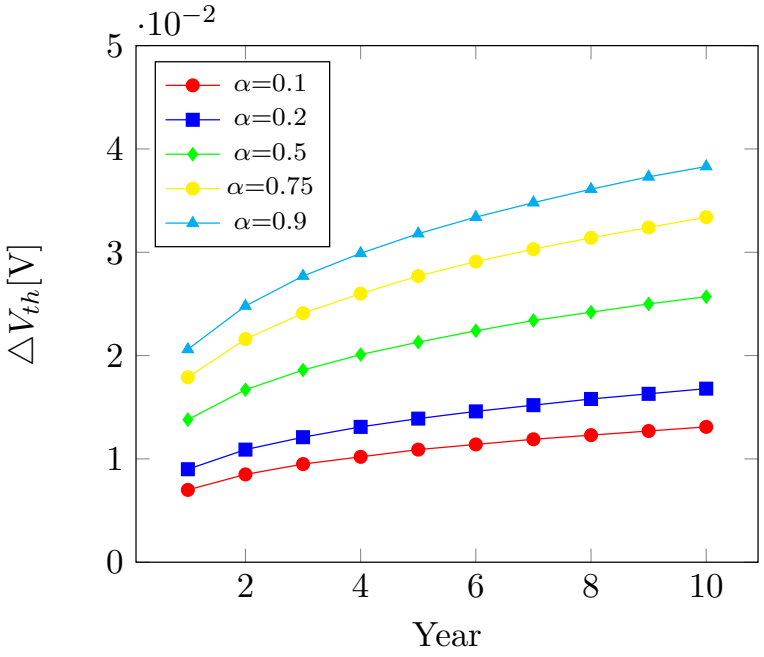


Figure 2.15: Threshold voltage degradation ($\triangle V_{th}$) over the time for CMOS inverter at different duty cycle due to NBTI effect.

As the MOSFET current is a function of V_{th}, an increase in V_{th} due to NBTI stress results in a reduction of the drive current of the pMOS transistor. An increase in V_{th} also reduces the overdrive voltage of a pMOS transistor. V_{th} is the minimum voltage required between the gate and source to allow the current to flow from the drain to source [51]. The overdrive voltage is defined as V_{gs} in excess of V_{th}, and is expressed as $V_{ov} = V_{gs} - V_{th}$. The reduction in the overdrive voltage, on one hand, causes the aged pMOS transistor to turn ON slower than a fresh pMOS transistor. On the other hand, it will turn OFF faster than a fresh pMOS transistor. The reduction in current and overdrive voltage changes the switching behaviour of CMOS logic circuits. In combinational logic circuits, a signal propagating through each path has to go through many logic inversions (i.e., NAND, NOR, etc). Therefore, to understand the impact of NBTI on signal propagation in combinational logic circuits, 11 CMOS inverters were constructed, as shown in Figure 2.16(a). As the falling signal edge propagates through the inverters, the impact of NBTI is magnified, resulting in a significant delay in the rising signal at the primary output, as shown in Figure 2.16(b). A similar situation is observed when a rising signal edge is asserted at the primary input as depicted in Figure 2.16(c). The increase in the delay of both the rising and falling signal edges due to NBTI could potentially affect the timing, thus leading to a catastrophic failure of the system.

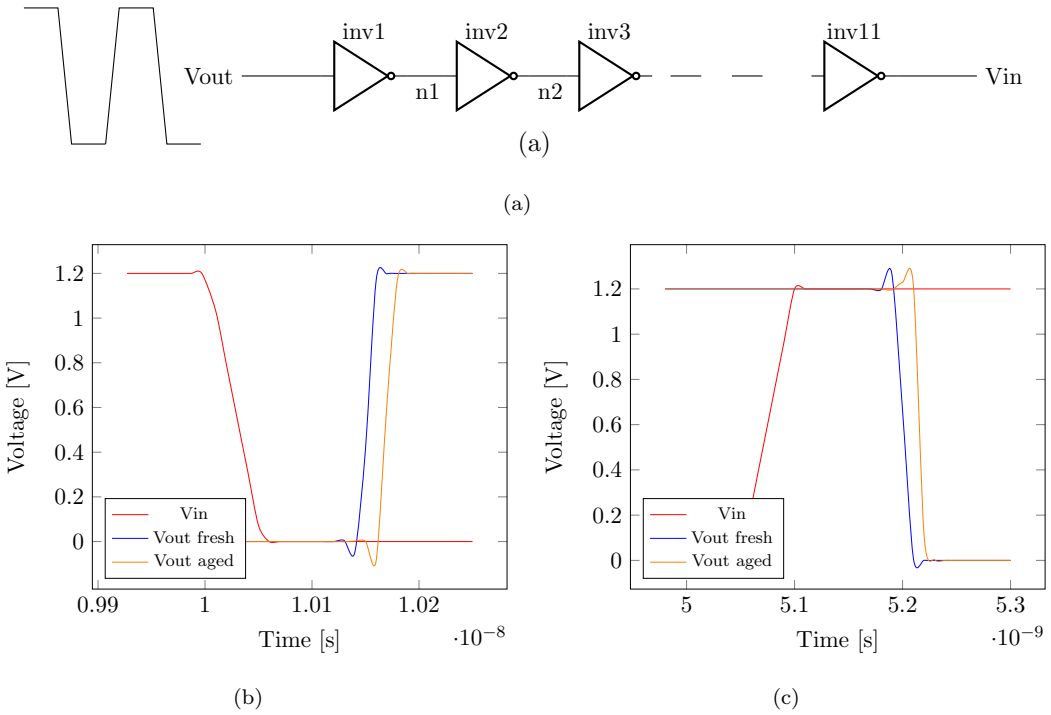


Figure 2.16: Transient behaviour of CMOS inverter under NBTI (a) a chain of 11 inverters, (b) assertion of falling signal at the primary input, (c) assertion of rising signal at the primary input.

In the context of a PUF, the reduction in the MOSFET current due to NBTI could potentially generate unreliable PUF responses. Therefore, a similar approach can be used to measure the impact of NBTI on the reliability of PUF responses, where the HD is used to compare the fresh and aged responses and is given as:

$$\text{Intra–HD}_{\text{NBTI}} = \frac{1}{m}\sum_{i=1}^{m}\frac{\text{HD}(R_{i.fresh}(n), R_{i.aged}(n))}{n} \times 100\% \tag{2.5}$$

2.6.2 Related Works of Ageing on PUFs

In the early research of PUFs, Lim *et al.* briefly discussed the ageing of an Arbiter-PUF, [30], but the ageing test was limited to one month under normal operating conditions and no significant performance degradation was found. Guajardo *et al.*, [23] performed an ageing test on an SRAM-PUF in an FPGA under normal operating conditions by continuously writing ones or zeros in the SRAM cells for over 10 minutes at a time. Subsequently, the FPGA was restarted, followed by a reading of the SRAM-PUF responses and compared with a reference measurement, which was taken prior to the ageing test. No significant impact was observed on the reliability of the SRAM-PUF when it was subjected to this ageing. Both studies [30, 23] indicate that within a limited time-frame

and under normal operating conditions, the impact of ageing on PUFs is inconclusive. Hence, this led researchers to further study the impact of ageing on PUFs over a longer period as compared to ageing studies in [30, 23].

Selimis *et al.*, [47] conducted an extensive ageing study on an SRAM-PUF which was fabricated using a 90-nm CMOS technology node. An accelerated ageing test was performed by applying high temperatures and supply voltages to induce the NBTI effect on the SRAM-PUF. Prior to the ageing test, one reference measurement (i.e., the PUF response) was measured at a temperature of 20^oC and a nominal supply voltage of 1.2V, with which all other measurements taken during the ageing test are compared. From the ageing test experiment, a 14% bit error was experienced by SRAM-PUF after 4.7 years. Elsewhere, Maes *et al.*, [39] performed a similar accelerated ageing test to induce NBTI effects on memory-based PUFs, including an SRAM-PUF, implemented in a 65-nm technology ASIC. After 4.5 years, the SRAM-PUF suffers about 8% bit error.

In a recent study [48], an accelerated ageing test was performed on RO-PUFs implemented on a Xilinx Spartan 3E FPGA. The total effective ageing period is about 13 years and within that time-frame, the RO-PUFs experienced 8.6% bit error. Rahman *et al.*, [46] discussed the impact of BTI and HCI on RO-PUFs at transistor level in a 90-nm technology node, simulated using HSPICE MOSRA. With the assumption that a PUF will only be used 20% of the time, BTI and HCI caused about 12.76% bit error on RO-PUFs after 10 years. An ageing-resistant RO-PUF (ARO-PUF) has been proposed in [46] to mitigate the impact of ageing on RO-PUF and ARO-PUF, which successfully reduces the bit error to 3.83% in 10 years. From the ageing analysis of RO-PUFs in [48] and [46], both agreed that ageing caused different degradation on the RO frequencies. It is an interesting observation because RO-PUFs are built from arrays of identically laid-out ROs. The only difference is the process variations on each RO. Although no claim has been made in [48], Rahman *et al.*, [46] claims that because of the process variations, each RO has a different degradation rate when the ROs are subjected to ageing.

All of the above studies focused on quantifying the bit error rates experienced by PUFs when they are subject to ageing. Elsewhere, the device ageing effect is exploited to improve the uniformity and reliability against environmental variations of SRAM-PUFs [52]. Xu *et al.*, [53] use a similar approach of device ageing to enhance the reliability of an Arbiter-PUF under environmental variations. A processor-based PUF is proposed in [54], which utilises the built-in arithmetic logic units (ALUs) such as n-bit ripple carry adder in a two-core processor architecture as a PUF. The outputs (i.e., the sum) of two ripple carry adders (i.e., one from each core) are fed to the arbiters to generate n-bit response. Due to a limitation to achieve symmetric placement of the arbiters and routing between each ALU to the arbiters, the PUF suffers bias responses. To overcome this, a device ageing technique is applied to intentionally increase the gate delays of a particular path which led to the bias response.

2.7 PUF Applications

Since PUFs were introduced in [21, 16, 12, 23, 24], they have been receiving a lot of attention because of their potential for being robust and cost-efficient hardware-based security devices. Based on the sub-types of silicon PUFs as discussed in Section 2.3.2, PUFs have been proposed for use in two primary applications, [22], which are: 1) low-cost identification and authentication; and 2) cryptographic key generation. The details of these two applications are explained in the next sub-sections.

2.7.1 Low-Cost Identification and Authentication

Identification and authentication are two essential processes in any security field. Identification occurs when a subject claims an identity such as a username, a process ID, or a smart card that can uniquely identify a subject. Authentication is the process of proving an identity by providing appropriate credentials. In IC application which is used to process sensitive and user-specific data, it is critical for an IC to be identified and authenticated securely. As the PUF output is device-specific and unpredictable, it is suitable for use by a challenge-and-response protocol to identify and authenticate an IC. As explained in Section 2.3.2, a Strong PUF has an exponentially large number of CRPs. Hence, it is the right type of PUF for a challenge-and-response protocol for IC identification and authentication.

Figure 2.17 illustrates the PUF-based authentication process. An enrolment phase takes place prior to the authentic device being deployed in the field. In the enrolment phase, a verifier that a trusted party applies randomly chosen challenges to obtain unpredictable responses of an authentic device and then stores these CRPs in a database j (d_j) for future identification and authentication. In the field, when device j, which contains a PUF, is requested for authentication, a verifier selects a challenge from d_j and obtains the PUF response from the device j. To avoid an exhaustive comparison with all CRP profiles in the database during the authentication process, the verifier performs a profile match to retrieve a d_j to which the response given by the device j is compared. The device j passes the authentication process if the response matches or is less than the HD threshold, ϵ, as compared to the stored value in the database. As the challenge-response protocol does not require an ECC by forgiving bit errors which are less than ϵ, it is extremely lightweight and suitable for resource-constrained devices such as RFID tags. These extremely lightweight PUF-based RFID tags can be promoted as a secure alternative to memory-based RFID tags and have been proposed as an anti-counterfeit solution for medical drugs, supply chain control, and secure access [55].

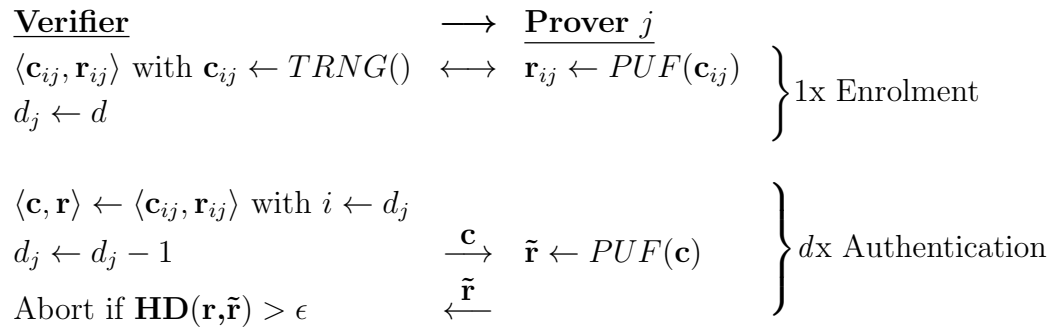


Figure 2.17: Low-cost PUF-based identification and authentication [22, 10].

In the PUF-based identification and authentication scheme shown in Figure 2.17, two types of error are possible to occur.

1. False negative - This occurs when the response of a valid PUF deviates significantly from its initial state stored in the database. It might be deemed to be a different PUF and rejected during an authentication process. The probability of this rejection is given as p_{reject} and can be computed using, [30, 22]:

$$p_{reject} = 1 - \sum_{i=0}^{\epsilon} \binom{n}{i} p_{intra}^{i} (1 - p_{intra})^{n-i} \tag{2.6}$$

   where p_{intra} denotes the bit error probability which is computed using Eq. (2.2), $\frac{\text{Intra-HD}}{100}$, n is the total number of bits in the response, and ϵ is the HD threshold.

2. False positive - This occurs when a wrong PUF is issued to a server, and the server authenticates it by mistake. The probability of this misidentification is given as p_{mis} and can be computed using, [30, 22]:

$$p_{mis} = \sum_{i=0}^{\epsilon} \binom{n}{i} p_{inter}^{i} (1 - p_{inter})^{n-i} \tag{2.7}$$

   where p_{inter} denotes the bit unique probability which is computed using Eq. (2.1), $\frac{\text{Inter-HD}}{100}$, n is the total number of bits in the response, and ϵ is the HD threshold.

Suh *et al.*, [33] further described that the unreliable bits may have an impact on the probability of misidentification. Given two chips, k and l, a misidentification will occur when $\text{HD}(\text{r}_l,\tilde{\text{r}}_k) \leq \text{HD}(\text{r}_k,\tilde{\text{r}}_k)$ due to the effect of the unreliable bits. Assume that all unreliable bits will always evaluate to the worst possible case such that:

$$\text{HD}(r_l, \tilde{r}_k) = \text{HD}(r_k, r_l) - \text{HD}(r_k, \tilde{r}_k), \tag{2.8}$$

the condition of misidentification can be re-written as:

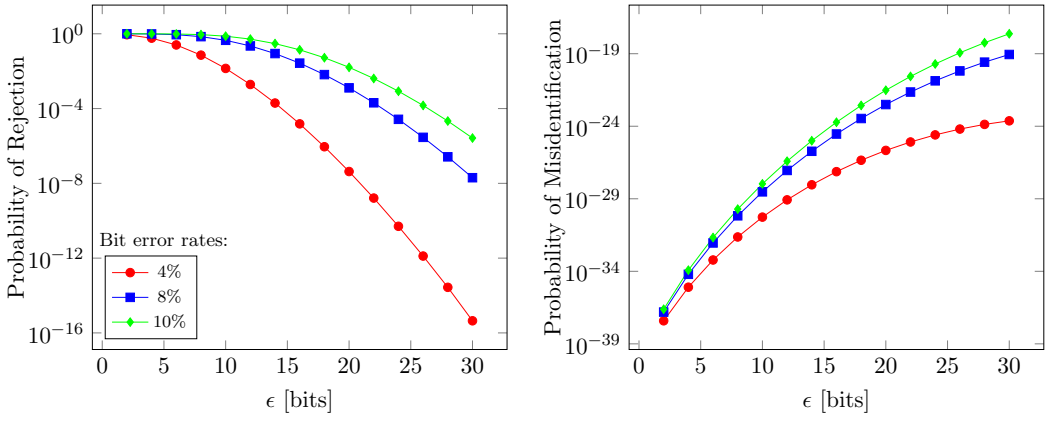
$$\mathrm{HD}(r_l, r_k) - \mathrm{HD}(r_k, \tilde{r}_k) \leq \mathrm{HD}(r_k, \tilde{r}_k) \tag{2.9}$$

$$\mathrm{HD}(r_l, r_k) \leq 2 \times \mathrm{HD}(r_k, \tilde{r}_k). \tag{2.10}$$

Therefore, the probability of misidentification is the product between the probability of a particular HD and the misidentification chance at that particular HD.

$$p'_{mis} = \sum_{i=0}^{2\cdot\epsilon} \binom{n}{i} p_{inter}^{i} (1 - p_{inter})^{n-i} \cdot \left[1 - \sum_{i=0}^{\epsilon} \binom{2\cdot\epsilon}{i} p_{intra}^{i} (1 - p_{intra})^{2\cdot\epsilon - i} \right] \tag{2.11}$$

Based on Eq. (2.6) and (2.11), Figure 2.18 shows the probability of rejection and misidentification at different bit error rates for a 128-bit identifier and the ideal uniqueness of 50%, under variations of ϵ. As can be seen from Figure 2.18, if ϵ is set too low, the probability of authentic PUFs being rejected increases, while setting ϵ too high increases the probability of misidentification. The setting of ϵ further impacts the vulnerability to an ML-attack [56]. An adversary only needs to achieve at least $\left(1 - \frac{\epsilon}{n}\right) \times 100\%$ prediction accuracy. It is always desirable for a PUF to achieve low bit error rates to reduce the probability of rejection/misidentification and improve security.



(a) Rejection

(b) Misidentification

Figure 2.18: Probability of rejection and misidentification at different bit error rates and ϵ for n=128-bit and p_{inter}=0.5.

The above challenge-response protocol is targeted to low-cost applications, hence, during the communication between the verifier and device j, the CRPs are clearly sent without any additional protection. Therefore, a man in the middle can intercept the challenge

and get the response from the PUF. Subsequently, the obtained CRPs is used to spoof device j. Therefore, it is suggested in [22], to never reuse the CRPs to avoid replay or man-in-the-middle attacks. In an extreme case, it is also possible that the attacker can get possession of device j and perform a non-invasive CRPs measurement. A low-cost challenge-and-response protocol can only be realised with a very strong assumption that Strong PUFs are resilient against ML-attacks. The susceptibility of Strong PUFs against ML-attacks will be described in Section 2.8.3.

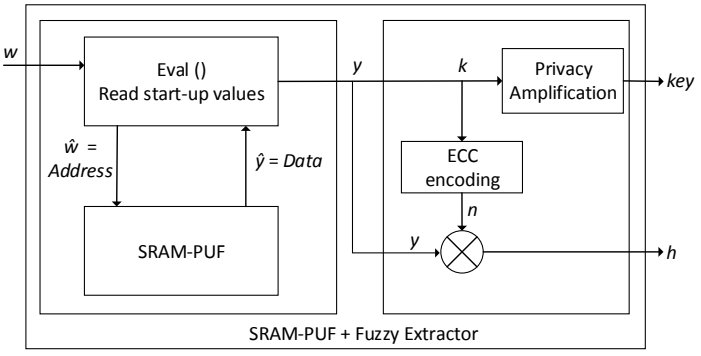
As mentioned earlier, a database is required to store the CRPs for a given PUF which might pose a limitation in terms of the physical space overhead in the verifier. The idea of using a compact or mathematical model for the verifier to save space has been suggested but not described in detail [55, 8, 57]. Nevertheless, as argued in [55], several key features of PUFs are lost if a compacted model is needed. For example, Strong PUFs are no longer resilient against model-building and the verifier needs to be a trusted entity to maintain the integrity.

2.7.2 Cryptographic Key Generation

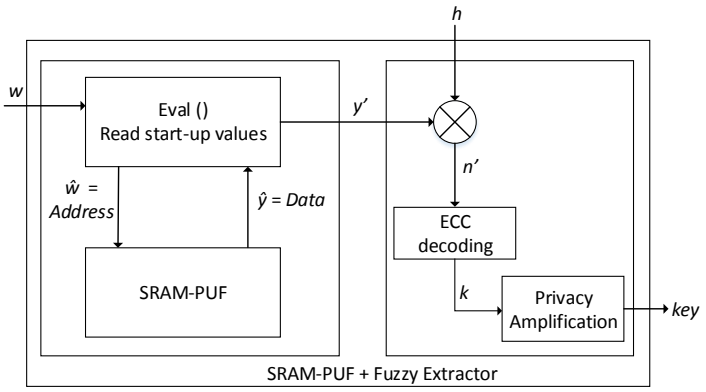
Another class of applications for PUFs is cryptographic key generation. Memory-based PUFs such as SRAM-PUF, Butterfly-PUF and Buskeeper-PUF are considered to be Weak PUFs and they are typically used for generating cryptographic keys [58, 59, 60]. However, outputs from the PUF circuit are known to be noisy due to environmental variations and ageing [48], hence, direct use of the secret key for cryptographic primitives is not feasible since these security applications require that every bit of a key stays constant [22]. Figure 2.19 shows the practical example of cryptographic key generation based on an SRAM-PUF. An ECC with the helper data, h, is used to generate error-free cryptographic keys. However, a partial information on the PUF response, k, could be recovered by the attacker because of the information from the helper data. Hence, privacy amplification is used to maximise the uncertainty of the generated keys [26]. Guajardo *et al.*, [23] suggested a hash function to be used as a privacy amplification module.

The procedure of cryptographic key generation is divided into two phases: *Enrolment* and *Reconstruction*. *Enrolment* occurs just once when a new key is generated or stored. The challenge to the SRAM-PUF is considered as an SRAM memory address range, given as w. y is the extracted SUVs of the SRAM-PUF. During this phase, an ECC receives k, which is a subset of the SUVs, y, and generates a codeword n. k is input to the privacy amplification module to generate a *key* and the helper data, h, is computed as $n \oplus y$. At the end of this phase, the helper data, h, and the memory address range, w, are stored in the memory. In the *Reconstruction* phase, the same PUF is measured again (i.e., the noisy response y') and a noisy codeword n' is computed as

$y' \oplus h$. An ECC is used to correct n' and the *key* that has been programmed during the *Enrolment* phase is then recovered after privacy amplification.



(a) Enrolment



(b) Reconstruction

Figure 2.19: The procedure of cryptographic key generation based on SRAM-PUF [26].

Unfortunately, ECC implementations require significant area overheads which increases as the bit error rate increases [61, 62]. The increase in the area overhead can hinder the widespread adoption of PUFs as a lightweight security entity. As mentioned earlier, although temperature and supply voltage variations can cause bit errors, the bit errors due to the ageing are more severe and worse for a prolonged time. In Chapter 5, we explore the feasibility of using an SRAM as both memory and PUF. Further, we propose a bit selection technique in a dual use SRAM to mitigate the ageing impact and reduce the area overhead of the ECC.

Based on the proposed usage of PUFs as mentioned above, other potential applications of PUFs such as IP protection in FPGAs [23, 25], remote activation of ICs [63], hardware-software binding [64, 65], and hardware obfuscation [66] have been suggested. Today, PUF-based hardware security has stepped out of the research labs into next-generation security products. For example, NXP uses PUF as a secure key storage to improve the security level of their incoming products [67].

2.8 Known Attacks to PUFs

As suggested by the name, a "Physical Unclonable Function", in which the most important feature is "Unclonable", is considered as a promising solution to deal with insecure key storage in NVM, hardware fingerprinting, and many other security problems. However, PUFs are threatened as many different successful attacks have already revealed vulnerabilities in certain silicon PUFs, with attack techniques such as invasive, semi-invasive, and non-invasive. Invasive attacks refer to attacks on physical systems where the physical properties of the chip are irreversibly modified. Common techniques for invasive attacks include micro-probing, Scanning Electron Microscope (SEM), and Focus Ion Beam (FIB), which require a sample preparation such as decapsulation and depassivation. Non-invasive attacks, however, requires no sample preparation. Non-invasive attacks only exploit the available information externally such as input and output values, running time, power consumption etc. For semi-invasive attacks, partial or complete removal of the device packaging is necessary. Unlike invasive attacks, no destructive modifications are required in semi-invasive attacks. Recent attacks on PUFs are discussed next and categorised based on the attack technique.

2.8.1 Invasive Attacks

Helfmeier *et al.*, [68] successfully cloned an SRAM-PUF using an FIB circuit edit. The cloning process has been performed on SRAM memory in the ATmega328P micro-controller, with a feature size of approximately 600-nm. After the package and excess bulk silicon of the device backside were removed, a Photonic Emission Analysis (PEA) was deployed through the backside of the IC to capture extremely weak photo-emissions from switching transistors during the power-up process. Based on the captured emission image, an FIB circuit edit is performed to alter the transistor characteristics according to the fingerprint of the target device. Although this shows that an SRAM-PUF could be cloned, producing a physical clone with these techniques remains economically infeasible for devices with limited financial value. Besides, modern ICs with small feature sizes require complex and expensive PEA techniques [69]. Elsewhere, an invasive attack on an SRAM-PUF through device ageing has been analysed and evaluated [70]. The fingerprint generated from an SRAM-PUF could be erased through a device ageing process, hence making it susceptible to a denial of service (DoS) attack.

2.8.2 Semi-invasive Attacks

A semi-invasive attack has been proposed in [71] wherein a laser stimulation (LS) technique is used to read-out the SUVs of SRAM-PUFs. The experiment has been conducted on SRAM memory in AtMega328P and ATXMEga128A1 micro-controllers, with feature

sizes of around 600-nm and 300-nm respectively. The package and excess bulk silicon of the device backside were removed prior to the LS process. The cloning of the PUF can be continued further as in [68] using an FIB circuit edit method, however, only the extraction method using the LS technique is discussed in which this technique could be accomplished down to a 180-nm technology node [71].

Tajik *et al.*, [72] use a PEA technique to physically characterise the Arbiter-PUF and extract its delay parameters. The 8-bit Arbiter-PUF was implemented on a Complex Programmable Logic Device (CPLD) manufactured in a 180-nm technology. Prior to the PEA process, the device was decapsulated and the bulk silicon material of the device was thinned down. The delay characterisation of the Arbiter-PUF was based on the time difference measured between enabling the PUF and photon emission at the outputs of the last stage (i.e., before the arbiter circuit) in which the challenge 00000000 was chosen as a reference measurement. Subsequently, each challenge bit is flipped (HD=1) and the procedure is repeated. Finally, the delay for each stage is revealed by subtracting the delay of the reference measurement and the delay of each challenge with HD=1. As the overall delay at the outputs of the last stage is the sum of the delay in each stage, the measured delay parameters further can be used to compute the responses for arbitrary challenges.

2.8.3 Non-invasive Attacks

Another attack model on a PUF is the non-invasive attacks. This type of attack is believed to be the most plausible attack as it is financially inexpensive for an adversary to perform an attack [16]. The attacker only has to access the interface of the device. Therefore, an attacker is restricted to non-invasive CRPs measurement and can apply a polynomial number of challenges to the device to collect the corresponding responses. With the measured CRPs of a particular PUF in hand, the adversary tries to build a numerical model of the PUF. Hence, a non-invasive attack is also known as a model-building attack. Several techniques have been proposed in the literature to perform model-building attacks such as ML [36, 73], side-channel [74, 60], and hybrid (i.e., combination of side-channel and ML) [75, 55].

An ML-attack is most applicable to Strong PUFs [73] and it was first demonstrated in [36] to predict the response of an Arbiter-PUF by using a Support Vector Machine (SVM). As mentioned in Section 2.4.1, the delay parameters of switching components in an Arbiter-PUF can be modelled with an additive linear model. Hence, an Arbiter-PUF can be predicted with high accuracy using an SVM as discussed in [36]. As countermeasures, the Feed-forward Arbiter-PUF, [30], XOR Arbiter-PUF, [22], and Lightweight-PUFs, [31] were proposed to introduce a non-linearity into the Arbiter-PUF. Further, Rührmair *et al.*, [73] performed a comprehensive ML-attack using SVM, Logistic Regression (LR), and Evolution Strategies (ES) on the aforementioned PUFs. Their initial

analysis was conducted on simulated data but they later verified the results by using measurements taken from an FPGA and an ASIC, which showed that ML techniques were able to model the Feed-Forward Arbiter-PUF, XOR Arbiter-PUF, and Lightweight-PUF with high accuracy [73].

In a recent study, Becker, [55] shows that an attacker who is in possession (i.e., has access to the primary inputs) of a PUF-based RFID tag can collect CRPs through a non-invasive measurement and perform an ML-attack. The parameters derived by the ML-attack was built into the software on a programmable RFID smart-card emulator and used to successfully clone a PUF tag. However, prior to the ML-attack, software reverse-engineering was performed to discover the configuration of the PUF model in the reader such as the exact linear feedback shift register (LFSR) and XOR-ing functions. As the internal configurations were known, the ML-attack was performed based on the mapping of the internal CRPs by using LR.

Based on all the above, the ML techniques such as SVM, LR and ES were used to perform an ML-attack on an Arbiter-PUF and its derivative. Elsewhere, Hospodar *et al.*, [56] compared an ML-attack performance between ANN and SVM on Arbiter-PUF and XOR Arbiter-PUF and the results show that ANN outperforms SVM. Vijayakumar *et al.*, [76] explored the learn-ability of ML techniques such as SVM and LR as the non-linearity element in PUF increases. Bagging and Boosting were also used for the ML-attack analysis [76], since these techniques have the potential to generate a strong classifier through the combination of several classifiers prediction. Based on previous ML-attack analyses [30, 55, 56, 73, 76], ANN, LR, ES, and Boosting are the most favourable to solve the non-linearity problem.

Delvaux *et al.*, [74] exploited repeatability imperfections in Arbiter-PUF responses due to CMOS device noise as side-channel information to perform a model-building attack. The key insight is that repeatability measurements provide direct timing information. If the delay difference, Δt, for a given challenge is very large, it is unlikely that the noise changes the sign of Δt. In contrast, if Δt is close to zero (i.e., entering the metastability state as discussed in Section 2.4.1), the response of the Arbiter-PUF will be influenced by the noise, resulting in the changes of Δt sign. Although the timing information is all relative, Delvaux *et al.*, [74] successfully built the model of Arbiter-PUF with >85% prediction accuracy.

Elsewhere, Zeitouni *et al.*, [60] exploited remanence decay in volatile memory as side-channel information to attack SRAM-PUFs, which were implemented in 65-nm CMOS technology. The approach in this attack is to recover the PUF response in a device after overwriting the SRAM-PUF with some data that are known to the adversary. In this study [60], the assumptions that the adversary has the ability to initialise the memory with a known values and that the adversary knows that the targeted PUF-based system uses a MAC were made. With these assumptions, the adversary can recover the PUF

state, based on the encryption observed in a series of data remanence experiments. The secret key is successfully recovered but this attack is likely to be impractical in terms of the timescale as it takes approximately two CPU-months. Besides, this attack is limited by the precision of the equipment to control the remanence decay in the SRAM.

Furthermore, a hybrid technique is proposed in which side-channel and ML techniques are combined [75]. As the CRP mapping complexity of Strong PUFs increases, the ML technique reaches its limit (i.e., the training time increases exponentially as the number of XORs increase) when applied to Lightweight-PUFs or XOR Arbiter-PUFs with a bit-length of 256 or more and with 6 XORs or more [73]. Rührmair *et al.* exploited the power and timing traces of an XOR Arbiter-PUF and a Lightweight-PUF as side-channel information to overcome the limitations in ML techniques [75]. Based on the power and timing side-channels, the cumulative number of zeros and ones in the outputs of the XOR Arbiter-PUF and Lightweight-PUF before the XOR gate is estimated. Further, the adapted ML technique is used to exploit this information and used to successfully attack the XOR Arbiter-PUFs and Lightweight PUFs for up to 16 XORs and for a bit-length of up to 512 (timing side-channel) and 128 (power side-channel) with high accuracy and polynomial training time.

2.8.4 Challenges

Thus far, the recent attacks on PUF which include invasive, semi-invasive, and non-invasive attacks were discussed and they are summarised in Table 2.2. One of the PUF core properties is that random process variations cannot be cloned, atom-by-atom, with current fabrication technologies [75]. Therefore, generating a perfect clone is still infeasible. Nevertheless, as discussed in Section 2.8.1, an SRAM-PUF can be physically cloned by editing the circuit using a FIB method. To the best of our knowledge, none other PUFs architectures have been physically cloned using the same method.

As can be seen in Table 2.2, invasive and semi-invasive attacks require complex techniques to perform an attack such as PEA, FIB, LS, and burn-in. These techniques have a relatively higher cost as compared to ML techniques used to perform non-invasive attacks. Nevertheless, if sufficient financial and technical investments are offered, an obvious threat to the PUF system is that its function is clonable using one of the attack types in Table 2.2. An intense competition between code-makers and code-breakers in the area of Strong PUFs and Weak PUFs is expected in the near future [73]. Nevertheless, proposing various attack methods help to validate the quality of a PUF and motivates people to look for countermeasures in overcoming the weakness of that particular PUF. It is expected that at some point this competition will converge to solutions that are resilient against the known attacks.

Table 2.2: Summary of known attacks to PUFs.

Attack Type	PUF Type	Platform	Technology	Technique
Invasive	SRAM-PUF [68]	ATmega328P	600-nm	PEA and FIB
	SRAM-PUF [70]	AS6C6264 SRAM IC	NR	Burn-in
Semi-invasive	SRAM-PUF [71]	AtMega328P	600-nm	LS
		ATXMEga128A1	300-nm	
	Arbiter-PUF [72]	Altera Max V CPLD	180-nm	PEA
Non-invasive	Arbiter-PUF [36]	ASIC	180-nm	SVM
	5-XOR Arbiter-PUF [73]	ASIC	45-nm	LR
	Lightweight-PUF (5 XORs) [73]	Simulation	NR	LR
	Feed-Forward Arbiter-PUF [73]	Simulation	NR	ES
	Arbiter-PUF [55]	PUF-based RFID tag	NR	LR
	Arbiter-PUF [74]	ASIC	65-nm	Side-channel
	SRAM-PUF [60]	ASIC	65-nm	Side-channel
	16-XOR Arbiter-PUF [75]	Xilinx Spartan-6 FPGA	NR	Side-channel and LR
	Lightweight-PUF (16 XORs) [75]	Xilinx Spartan-6 FPGA	NR	Side-channel and LR

NR=not reported.

2.9 Advantages of PUFs

As discussed in Chapter 1, PUF technology is a promising solution to answer the weakness of current cryptographic key storage technologies, which rely on storing secret keys in NVMs, such as tampering, programming, and high cost. Despite the attacks on PUF which have been discussed in Section 2.8, PUFs have a few advantages over current key storage solutions. The keys generated from PUFs are not programmed by the IC manufacturer or system owner but randomly generated based on intrinsic process variations. The key is device-specific and therefore, it is easy to manage and eliminates the potential threat of the key being compromised by an untrusted third party who is responsible for key programming. Although the PUF is functionally clonable, due to the nature of a device-specific key, an individual attack procedure is needed. In that context, mass-cloning tends to become impractical and PUFs help in reducing the damage. Moreover, a PUF does not require any special fabrication for storing the key. Hence, it is relatively low cost compared to the current key storage technologies [9]. PUFs can also be seen as scalable security entities, in which their size (i.e., the challenge bits) can be increased to introduce more randomness due to process variations and the size of the key can be increased as well.

2.10 Summary

This chapter has given an introduction to PUFs and has covered how these PUFs can be used for hardware fingerprinting, the types of PUF which are available in the literature and important metrics to evaluate the PUF. The impact of ageing is further discussed, which leads to a reliability issue for PUFs over an extended period. Two applications have also been discussed: low-cost IC identification and authentication through a challenge-and-response protocol, and cryptographic key generation. Nevertheless, the bit errors experienced by a PUF due to environmental variations and ageing introduce a trade-off between security and reliability in challenge-and-response protocols. In cryptographic key generation, the area overhead of ECC increases significantly as the bit error rate increases. This can hinder the widespread adoption of PUFs, particularly for resource-constrained devices. Furthermore, the review of known attacks on PUFs suggests that the most plausible attack is a non-invasive attack, as it is financially inexpensive. Invasive and semi-invasive attacks seem very promising but are limited by the precision and accuracy of the tools as CMOS technology scales down. The literature reviewed in this chapter suggests two open problems before PUF can be promoted as lightweight security entities, which are reliability under ageing and susceptibility to ML-attacks.

Chapter 3

PUF Implementation and Evaluation

As discussed in Section 2.4, silicon-based PUF structures can be generally categorised into four categories which are: delay-based PUFs, memory-based PUFs, mixed-signal PUFs, and emerging nanotechnology-based PUFs. For all known silicon PUF structures, there exists a performance trade-off between power, area, speed, and security. Since PUFs are also suggested for energy and resource-constrained applications such as RFID, as discussed in Section 2.7.1, some of the considered PUF structures focused on low-power and/or lightweight design techniques. PUFs have been proposed in [35, 77, 78, 79] to operate in the sub-threshold region instead of super-threshold region aiming to minimise the power consumption. Furthermore, the exponential current-voltage behaviour in the sub-threshold region could be exploited to increase the security level or resilience against an ML-attack, as in the sub-threshold current array PUF (SCA-PUF) [80]. Motivated by this potential, the "Two Chooses One" PUF (TCO-PUF) was proposed [81], which exploits the non-linear current-voltage behaviour in the sub-threshold region, and it has been implemented using a 180-nm predictive technology model. The process variation is modelled using a normal distribution with 10% tolerance in the transistor size. The architecture of the TCO-PUF is adapted from the SCA-PUF, [80]. Hence, Zufu, [81] claims that the resilience against an ML-attack is inherited, but this has never been quantified. Therefore, to achieve the first objective, described in Section 1.4, the TCO-PUF is used as a case study for PUF characterisation using a TSMC 65-nm technology node. This includes the analysis of the susceptibility to an ML-attack. A 65-nm CMOS technology has been used for our analysis as previous studies which are listed in Table 2.1 indicate that the process variations in 32-nm to 180-nm technology node could be exploited to build a PUF. Additionally, the impact of ageing on a TCO-PUF is explored. Since the TCO-PUF is based on a differential design similar to the Arbiter-PUF, it is also interesting to analyse and compare the results with the Arbiter-PUF which was proposed by Lee *et al.*, [12]. Both PUFs are built in a 65-nm technology and the bit

error rates due to ageing are compared with some of the reported data in the previous literature. The main contributions of this chapter are:

1. We show that the exponential current-voltage behaviour in the sub-threshold region, which is exploited by TCO-PUF architecture, achieved an insignificant improvement of its resilience against an ML-attack. This also shows that the exponential current-voltage behaviour in the sub-threshold region has no impact in increasing the complexity of the challenge to response mapping.

2. We show that PUFs based on a differential design technique such as the Arbiter-PUF and the TCO-PUF can mitigate the first order effect of ageing (i.e., the duty cycle and supply voltage). Nevertheless, a finite bit error still occurs, possibly as a result of second-order effects such as the interaction of variations in the threshold voltage due to the fabrication process and ageing phenomena.

This chapter is organised as follows: Section 3.1 describes the motivation to characterise the TCO-PUF and to study the impact of ageing on PUFs. Section 3.2 describes the fundamentals of MOSFET operation in the sub-threshold region. The implementation of the TCO-PUF including the transistor array and the voltage sense amplifier is discussed in Section 3.3. The experimental methodology and the evaluation of quality metrics of the TCO-PUF which includes uniqueness, reliability, and uniformity are described in Section 3.4. The susceptibility of TCO-PUF to ML-attack is discussed in Section 3.5. In Section 3.6, the impact of ageing on PUFs is discussed. Finally, the chapter is concluded in Section 3.7.

3.1 Motivation

Vivekraja *et al.*, [82] were among the first to explore and study the impact of operating PUFs in the sub-threshold region for a 90-nm technology node. An RO-PUF was used as a case study in which the supply voltage and body bias were varied. The focus of [82] is to exploit the high sensitivity of the circuit to process variations in the sub-threshold region to improve the performance of PUF characteristic such as uniqueness. The study is conducted in [38] for an Arbiter-PUF which was fabricated using a 45-nm CMOS technology had a similar focus. While all the above studies used previously proposed PUFs, elsewhere, new PUF architectures that work in the sub-threshold region were proposed, such as the common-source-amplifier-based PUF, [77], the "nanokey" PUF, [78], and the hybrid RO-PUF, [79]. The aim of the aforementioned PUFs is to minimise the total power consumption. Li *et al.*, [83], proposed a new PUF architecture which can be classified as a mixed signal PUF and has the capability to be operated with a supply voltage from 1.2V and below to the near sub-threshold region of 0.6V. This enables it to be integrated into an energy-constrained IC without separate supply distribution.

All the above PUFs are mainly aimed at improving the PUF quality metrics, such as uniqueness, or minimising the power consumption by operating the PUFs in the sub-threshold region. Mukund *et al.*, [80] proposed an SCA-PUF to improve the PUF performance in the aspect of security. The non-linear dependency of current and voltage in the sub-threshold operating region is exploited to increase the unpredictability of the SCA-PUF against an ML-attack. Mukund *et al.*, [80] showed that the SCA-PUF achieves lower predictability of about ≈92.5% over an Arbiter-PUF of about ≈99.3% prediction accuracy at 1000 CRPs (i.e., training data). However, as the number of CRPs increases up to 8000, the SCA-PUF can be predicted with a high accuracy of about 98%. As the TCO-PUF architecture is adapted from the SCA-PUF, it raises a major concern about the security level of the TCO-PUF against ML-attack which motivates us to investigate and quantify the security performance of the TCO-PUF.

Furthermore, the reliability is an important aspect to investigate. As explained earlier in Section 2.6, when an integrated CMOS device is used continuously the device ages because of NBTI. This will gradually degrade the circuit performance. For a PUF to be useful, it should reliably generate the same response for a given challenge. Unfortunately, ageing has a similar effect on PUFs as on other integrated CMOS devices. Ageing may flip a PUF response. Hence, this results in an increase in the bit error rate over time. For low-cost authentication, as illustrated in Figure 2.17, NBTI can potentially degrade the authentication capability after prolonged use, if the bit error rate goes beyond the HD threshold, ϵ. A differential design technique has a potential to mitigate and cancel out the first-order dependencies of ageing [84]. As the TCO-PUF is constructed based on a differential design technique and is categorised as a Strong PUF (i.e., suitable for low-cost identification and authentication), it is interesting to investigate its potential to achieve low bit error rates (with ageing), which is desirable for low-cost identification and authentication, as discussed in Section 2.7.1.

3.2 MOSFET in Subthreshold Region

As mentioned earlier in this chapter, the TCO-PUF architecture is based on the sub-threshold operation region. Therefore, this section provides an overview of the non-linear dependency of current and voltage in the sub-threshold region which is essential for the TCO-PUF. Unlike the strong inversion region, in which the drift current dominates, in the sub-threshold region, the channel of the transistor is not inverted and current flows by diffusion, [85, 86].

The basic model of sub-threshold current can be expressed as [85]:

$$I_d = I_o \exp\left(\frac{V_{gs} - V_{th}}{nV_t}\right) \tag{3.1}$$

where I_o is the drain current when $V_{gs} = V_{th}$, $I_o = \mu_o C_{ox} \frac{W}{L}(n-1)V_t^2$, V_{th} is the transistor threshold voltage, n is the sub-threshold slope factor ($n = 1 + \frac{C_d}{C_{ox}}$), and V_t is the thermal voltage, $V_t = k_B T/q$.

By considering the short-channel effect, V_{th} roll-off and the coefficient of Drain-Induced Barrier Lowering (DIBL), η, the current equation becomes:

$$I_d = I_o \exp\left(\frac{V_{gs} - V_{th} + \eta V_{ds}}{nV_t}\right)\left(1 - \exp\frac{-V_{ds}}{V_t}\right) \tag{3.2}$$

The sub-threshold slope of the transistor is defined as, $S = nV_t ln10$. Substitution into Eq. (3.2) gives the sub-threshold current in the form of:

$$I_d = I_o 10^{\frac{V_{gs}-V_{th}+\eta V_{ds}}{S}}\left(1 - 10^{\frac{-nV_{ds}}{S}}\right) \tag{3.3}$$

Equation (3.3) shows that the sub-threshold current varies exponentially with V_{th} and V_{gs}. As discussed in Section 2.2, as the transistor size scales down, the threshold voltage variation increases due to RDF. This indicates that even small changes in V_{th}, result in an exponential fluctuation in the sub-threshold current.

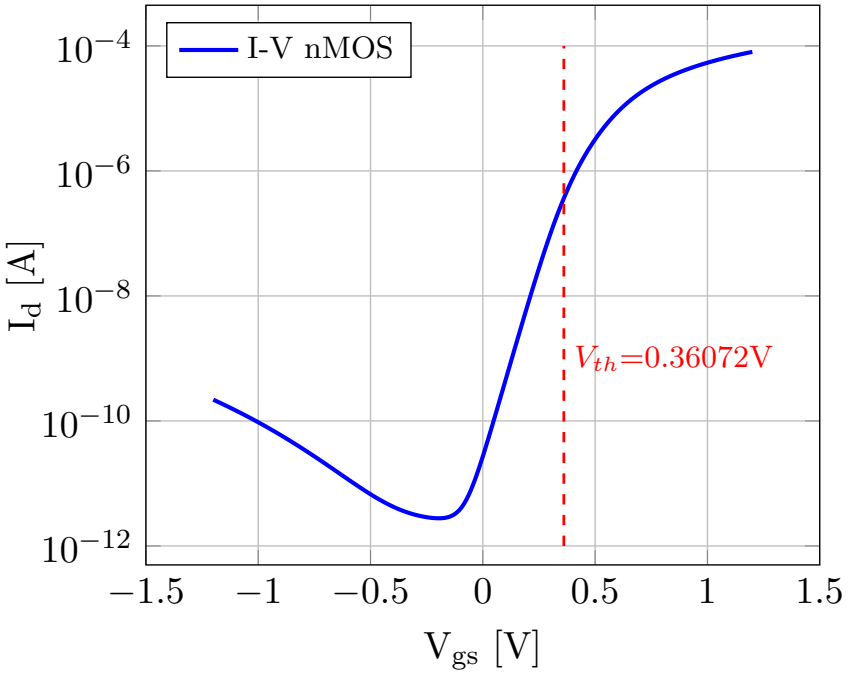


Figure 3.1: I-V characterisation for nMOS using a low-k TSMC 65-nm CMOS technology node.

Figure 3.1 shows the current-voltage (I-V) characterisation for an nMOS using a low-k TSMC 65-nm CMOS technology node, plotted as a semi-log graph. The V_{th} for nMOS is

0.36072V. As V_{gs} reduces slightly below V_{th}, the device enters the sub-threshold region [86]. The exponential relationship of the current and voltage can be seen clearly in the range of $0 \leq V_{gs} \leq V_{th}$, in Figure 3.1. The device approaches the flat band voltage region as V_{gs} becomes negative. In this region, there is no charge because of large energy barriers and the current is no longer non-linearly dependent on V_{gs} and V_{th} [86].

3.3 TCO-PUF Architecture

The top-level architecture of the TCO-PUF is depicted in Figure 3.2. The top level consists of two main blocks which are an analogue unit and an ADC unit. The analogue unit consists of two identical transistor arrays, namely TCOA and TCOB, that are driven by the same inputs, and thus in the absence of variability produce identical output voltages. All the transistors in both arrays operated in the sub-threshold region are subject to stochastic variability in their threshold voltages. The random variations in analogue output voltages of the two transistor arrays are caused by current mismatches resulting from the threshold voltage variations. A voltage sense amplifier is used as an ADC that sense the difference between these two voltages and a response, '1' or '0' is generated based on the sign of the differential output of the transistor arrays.

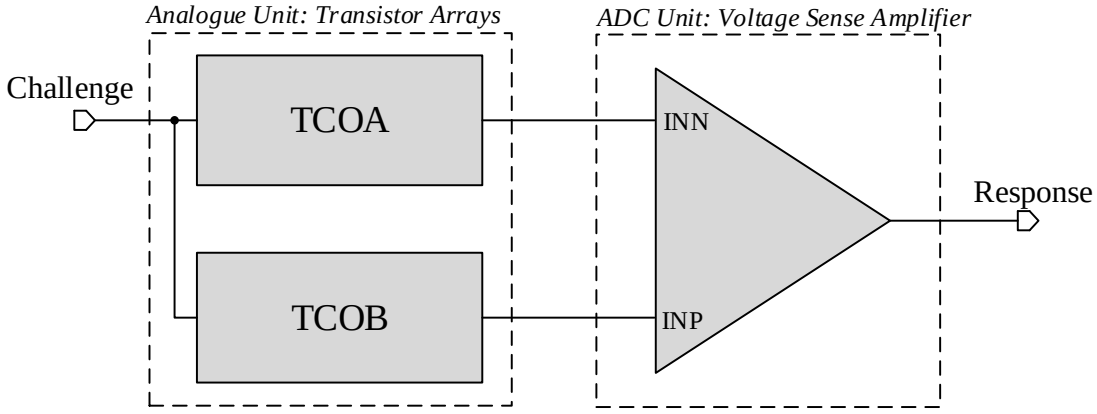


Figure 3.2: Top level of TCO-PUF architecture.

3.3.1 Design of Transistor Arrays

The details of the construction of the transistor array are shown in Figure 3.3. There are two networks in the array, which are the nMOS network (upper) and pMOS network (lower). In each network, the array is composed of k columns and n rows ($k \times n$ array). A non-stochastic transistor, which is indicated by 'x' (e.g., P11x), is in parallel with a stochastic transistor (e.g., P11). Each row in each column consists of two pairs of parallel non-stochastic and stochastic transistors, where each pair is indicated with and without a prime symbol ($'$), respectively. The prime symbol represents a challenge c', the inversion

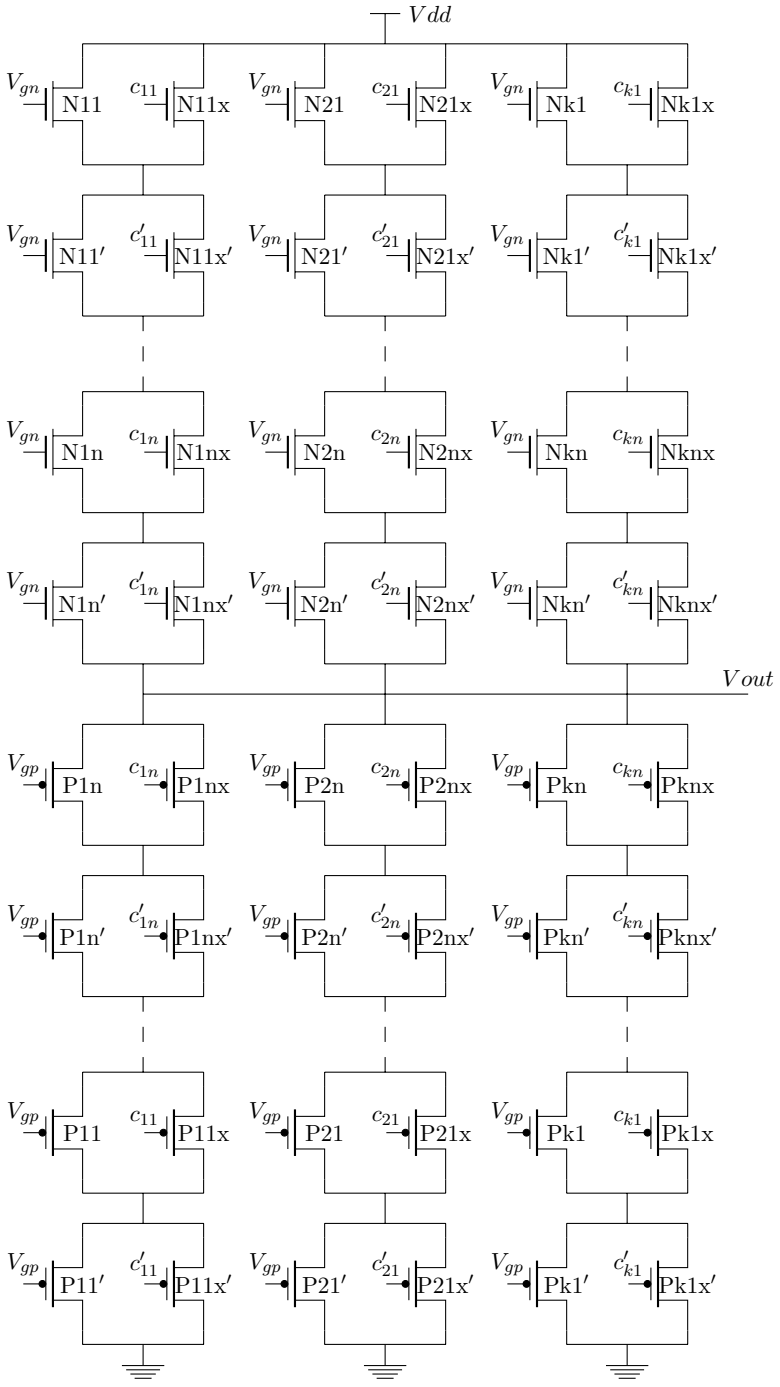


Figure 3.3: The construction of the transistor array of TCO-PUF.

of c and all transistors related to c' are marked with the ($'$) symbol. The term 'stochastic transistor' refers to a transistor with high variability in the threshold voltage. According to Ye *et al.*, [20], the standard deviation of the threshold voltage is more pronounced when the transistor width and length are minima. Hence, all stochastic transistors in the transistor array have a minimum width and length. For the non-stochastic transistor, the sub-threshold current needs to be negligible and at the same time, it has to provide a small on-state resistance. Thus, the width and length of the non-stochastic transistors are set to $10W_{min}$ and $10L_{min}$ [81], respectively. Increasing the width and length of the non-stochastic transistors will also reduce the errors due to geometrical mismatches [51, 78]. Therefore, there will be less variation in the electrical behaviour which is desirable for the non-stochastic transistors. By performing a DC sweep analysis, a proper bias is found for V_{gn} and V_{gp} to ensure the stochastic transistors always operate in the sub-threshold region.

Based on the above configuration, the non-stochastic transistor acts as a switch either to remove the effect of the stochastic transistor when ON, or to include it when OFF. For this architecture, there is always a contribution from the stochastic transistor regardless of whether the challenge bit is '0' or '1'. This is because the non-inverted (c) and inverted (c') bits of the same challenge are each connected to a non-stochastic transistor. Thus, in any condition, one out of two stochastic transistors will be selected and be part of the network. Therefore, this architecture is known as "Two Chooses One" or TCO-PUF. As can be seen from Figure 3.3, both nMOS and pMOS networks are connected to the same bits of the challenge input. Therefore, the transistor array acts as a voltage divider in which the V_{out} fluctuates within the range of half of the supply voltage, V_{dd}. For the TCO-PUF architecture, the size of the CRP set is 2^{kn}, which is one of the criteria for a Strong PUF [28].

3.3.2 Design of Voltage Sense Amplifier

As mentioned earlier, a voltage sense amplifier is required to sense the difference of the output voltages from the transistor arrays and digitise the response to '1' or '0'. Figure 3.4 shows the construction of the latch-type voltage sense amplifier which is also known as a dynamic comparator [87]. This structure offers rail-to-rail output swing and no static power consumption [87]. Each of the transistor array outputs is connected to INN or INP. During the reset phase when $CLK = 0$, both output nodes, $outP$ and $outN$, are pulled to V_{dd} by the reset transistors M7 and M8. The transistor M_{tail} is OFF in the reset phase. In the comparison phase, when $CLK = V_{dd}$, the transistors M7 and M8 are OFF and M_{tail} is ON. Also, in this phase, the transistors M1 and M2 sense the difference of the output voltages from the transistor arrays. Subsequently, the output nodes, $outP$ and $outN$, which have been pre-charged to V_{dd}, started to discharge at different rates depending on the corresponding input voltage (INN/INP). When

$V_{INP} > V_{INN}$, the drain current of the transistor M2 is higher than M1's drain current. Therefore, $outP$ discharges faster (through transistors M2 and M4) than $outN$ (through transistors M1 and M3). When $outP$ is pulled down and as it reaches $V_{dd} - |V_{thp}|$ before $outN$, the transistor M5 will turn on initiating the latch regeneration caused by the back-to-back inverters (M3, M4, M5, and M6). Hence, output node $outN$ is pulled to V_{dd} and $outP$ discharged to ground. The comparator circuits work in the opposite manner if $V_{INN} > V_{INP}$. The above operation can be seen more clearly from the transient behaviour as depicted in Figure 3.5, in which the sense amplifier in Figure 3.4 was simulated using a TSMC 65-nm technology. In this case, INP is 0.605V, and INN is 0.6V, which gives a difference in the input voltage, ΔV_{in} of 5mV. As can be seen from Figure 3.4, the outputs $outN$ and $outP$ are only valid during the comparison phase. Hence, both outputs of the voltage sense amplifier are connected to an SR-latch which is used to capture each output transition in the comparison phase and to hold the state until the next comparison phase, such that a response, '1' or '0', is generated at every clock cycle.

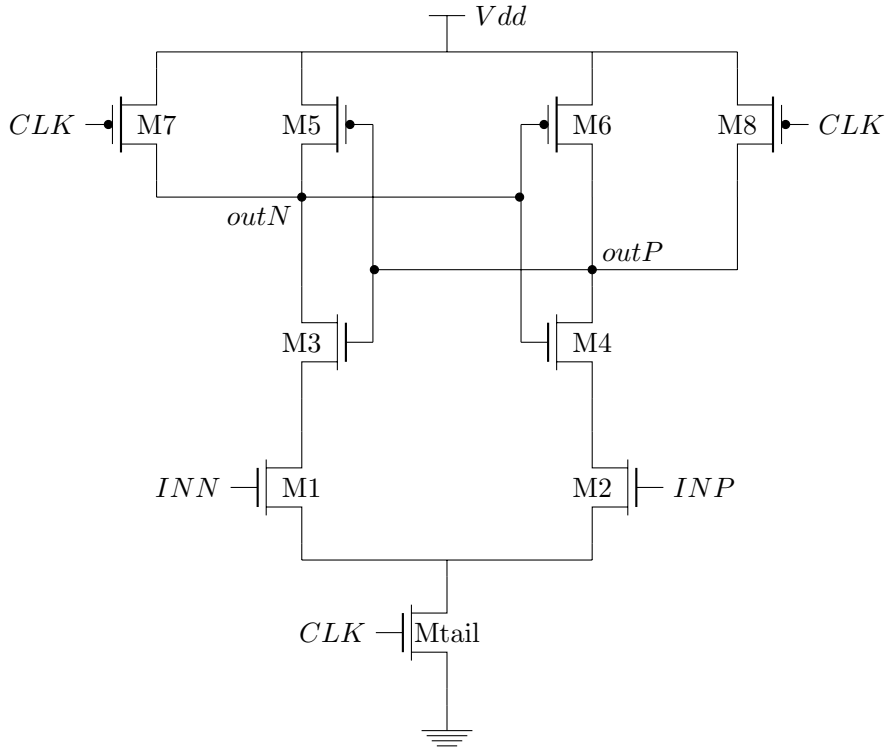


Figure 3.4: The construction of the voltage sense amplifier.

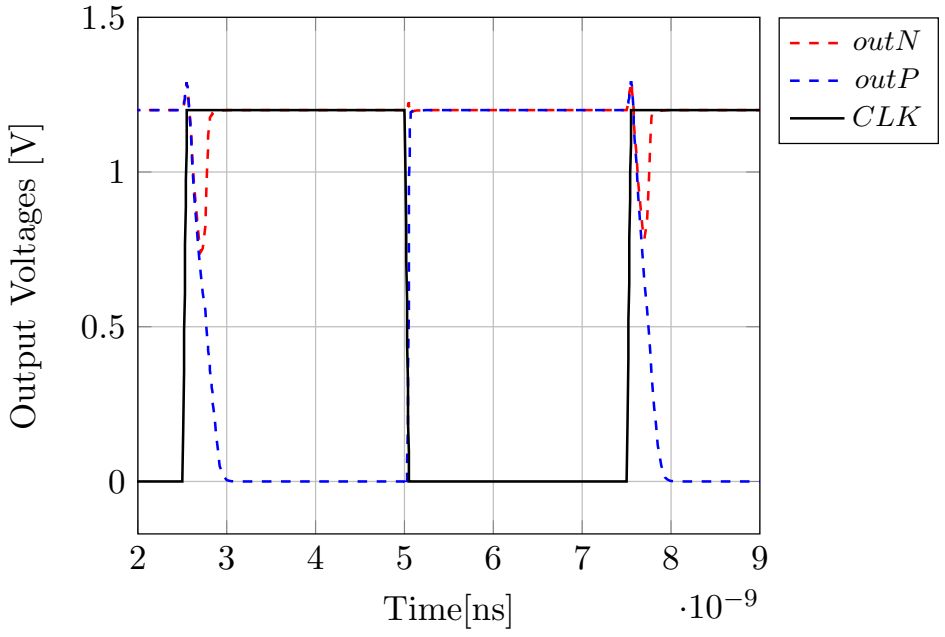


Figure 3.5: Transient simulation of the voltage sense amplifier for input voltage difference $\Delta V_{in} = 5\text{mV}$ and $V_{dd} = 1.2\text{V}$.

3.4 Simulation Results and Analysis

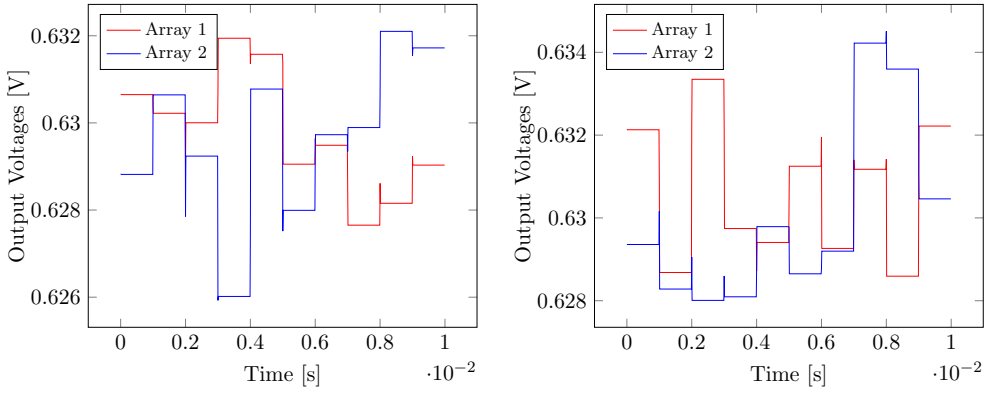
In this section, the simulation setup, the analysis of transistor arrays voltages and the evaluation of PUF quality metrics such as uniqueness, reliability, and uniformity are discussed.

3.4.1 Setup

A 32-bit (k=8 and n=4) TCO-PUF has been designed in a low-k TSMC 65-nm technology node and simulated using the BSIM4 (V4.5) transistor model with a nominal supply voltage of 1.2V and a room temperature of 25^{o}C. Intrinsic variations such as oxide thickness and threshold voltage are modelled in Monte Carlo simulations using the built-in fabrication standard statistical variation (3σ variations) in the technology design kit. Based on the previous studies which were conducted in the simulation setting (see Table 2.1), in our study, a Monte Carlo simulation was performed for 100 TCO-PUF instances. A 32-bit response is generated from each of the 100 PUF instances. Subsequently, MATLAB is used to post-process the responses.

3.4.2 Analysis of Transistor Arrays

A transient analysis was performed on both transistor arrays in Figure 3.2 and their output voltages are recorded. As mentioned in Section 3.3.1, a transistor array acts as a voltage divider. As can be seen from Figure 3.6, the output voltages of the transistor arrays always fluctuate within the range of 50% of the supply voltage. Effectively, the non-stochastic transistors are used to set the voltage level ($\frac{V_{dd}}{2}$) and the output voltage fluctuations are caused by the random variations of the threshold voltage in the stochastic transistors. As in the sub-threshold region relies heavily on leakage current for charging and discharging the capacitance, a clock period of 1ms is used in this simulation to obtain a stable and distinct output voltages from the transistor arrays as depicted in Figure 3.6[1]. These outputs are input to the voltage sense amplifier to generate a response, '1' or '0'.



(a) PUF instance 1

(b) PUF instance 2

Figure 3.6: Output voltages from transistor arrays of two TCO-PUF instances.

3.4.3 PUF Metrics Evaluation

In order to quantify the quality of the PUF, several parameters, as discussed in Section 2.5, need to be evaluated. These parameters are uniqueness, reliability, and uniformity.

3.4.3.1 Uniqueness

Based on the 32-bit response generated from each of the 100 TCO-PUF instances at a condition of 1.2V and 25^{o}C, the uniqueness is measured by the Inter-HD using Eq. (2.1). Figure 3.7 shows the Inter-HD distribution which has a mean of 0.50 and standard deviation of 0.1466. Although the TCO-PUF has a mean value of uniqueness around 50%, it suffers from a large standard deviation. An HD close to 0 indicates that the

[1]The waveforms in Figure 3.6 have been recorded up to only 10ms for better visibility.

PUF instances have almost similar responses which increases the possibility of misidentification. Conversely, an HD close to 1 increases the susceptibility to a simple guessing attack as the response of one PUF instance is opposite of the other. As mentioned in Section 2.5.1, the ideal distribution of Inter-HD is centred around 50%.

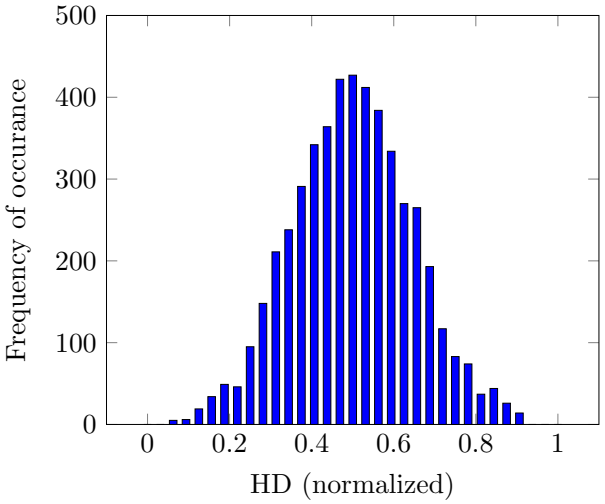


Figure 3.7: Uniqueness of 32-bit TCO-PUF for 100 instances.

3.4.3.2 Reliability

For the reliability evaluation, the temperature was varied within the range of -40^oC to 85^oC and the supply voltage of 1.2V varied by $\pm$ 10%. These temperature and supply voltage variations correspond to the corner values that are typically assumed for consumer-grade IT products [88]. Within the specified range of the temperature and the supply voltage, in total, 12 conditions were set as depicted in Figure 3.8. In each condition, a 32-bit response is generated from each of the 100 TCO-PUF instances. A measured response at a nominal supply voltage of 1.2V and a temperature of 25^oC has been used as a reference, to which all other measured responses have been compared. Equation (2.2) is used to calculate the bit error rate due to the temperature and supply voltage fluctuations. The reliability of the 32-bit TCO-PUF is depicted in Figure 3.8. As can be seen from Figure 3.8, under temperature variations at a nominal supply voltage of 1.2V, the reliability of TCO-PUF is above 93%. However, the reliability of TCO-PUF reduces when it is subjected to both variations. Overall, the average reliability of TCO-PUF under temperature variations of -40^oC to 85^oC and a supply voltage of 1.2V $\pm$10% is 92.38% (i.e., 7.62% bit error rate). It can be concluded that on average, 2 bits out of a 32-bit response of TCO-PUF were flipped. An ideal value of the reliability is 100% in which the response is stable across both temperature and supply voltage variations.

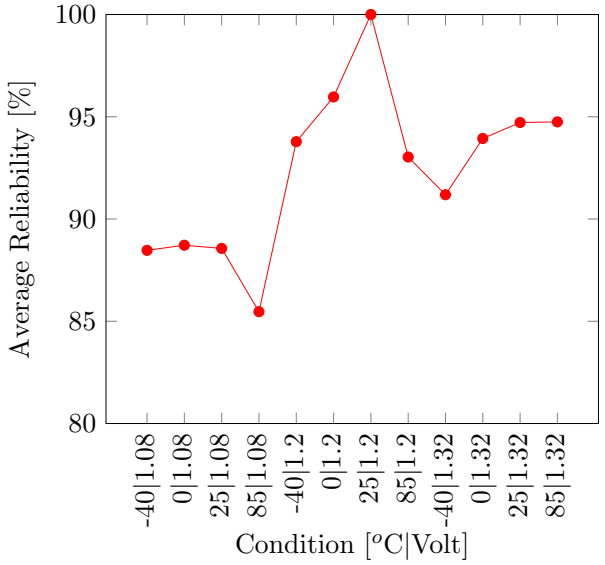


Figure 3.8: Reliability for 32-bit TCO-PUF under temperature variations of -40^{o}C to 85^{o}C and a supply voltage of 1.2V ±10%.

3.4.3.3 Uniformity

Based on the 32-bit response generated from each of the 100 TCO-PUF instances at the conditions of 1.2V and 25^{o}C, the uniformity is measured by using Eq. (2.4). Figure 3.9 shows the uniformity distribution which has a mean of 0.5281. As of uniqueness, the uniformity metric also suffered from a dispersion which has a large standard deviation of 0.2494. An HW is defined as the number '1' bits in the response. Therefore, the mean and standard deviation for uniformity metric of TCO-PUF indicate a lack of randomness in the responses.

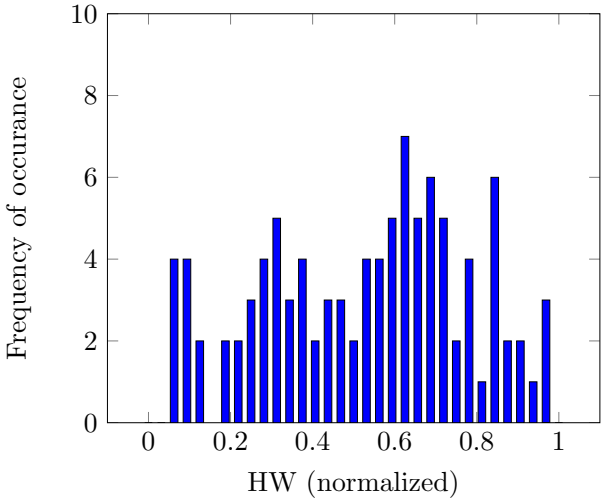


Figure 3.9: Uniformity of 32-bit TCO-PUF for 100 instances.

3.5 ML-Attack Susceptibility

As defined in Section 2.3.2, a Strong PUF must be resilient against a model building attack. Hence, the security of the TCO-PUF against an ML-attack is discussed in this section. For the ML-attack evaluation, we employed an ANN, as it is one of the best ML techniques able to solve non-linear problems, as discussed in Section 2.8.3. Although the ML-attack analysis is introduced in this chapter, the details of the ANN setup are described in Chapter 4 to align with the context of that chapter. For the ML-attack analysis, 32000 randomly generated challenges were applied on a 32-bit TCO-PUF and their corresponding responses are generated following the simulation setup as discussed in Section 3.4.1. The test set for prediction accuracy computation, which consists of 2000 CRPs, has been chosen randomly, and it is not part of the training set. The number of CRPs in the training set is incrementally increased (i.e., a step size of 1000 CRPs) and for each training set, the ML-attack is performed using MATLAB (see Appendix A.6).

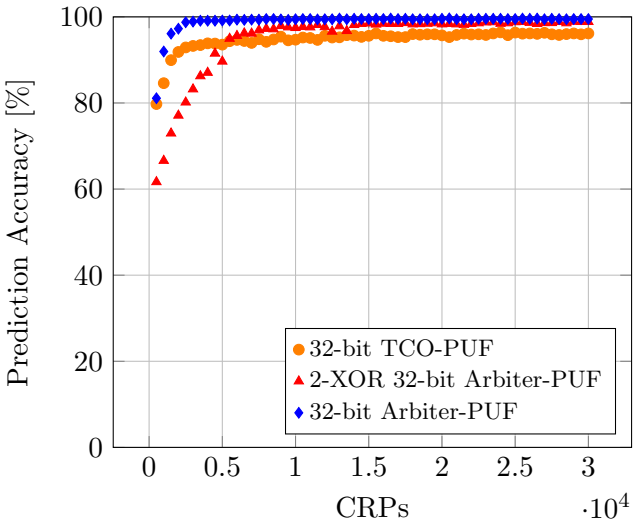
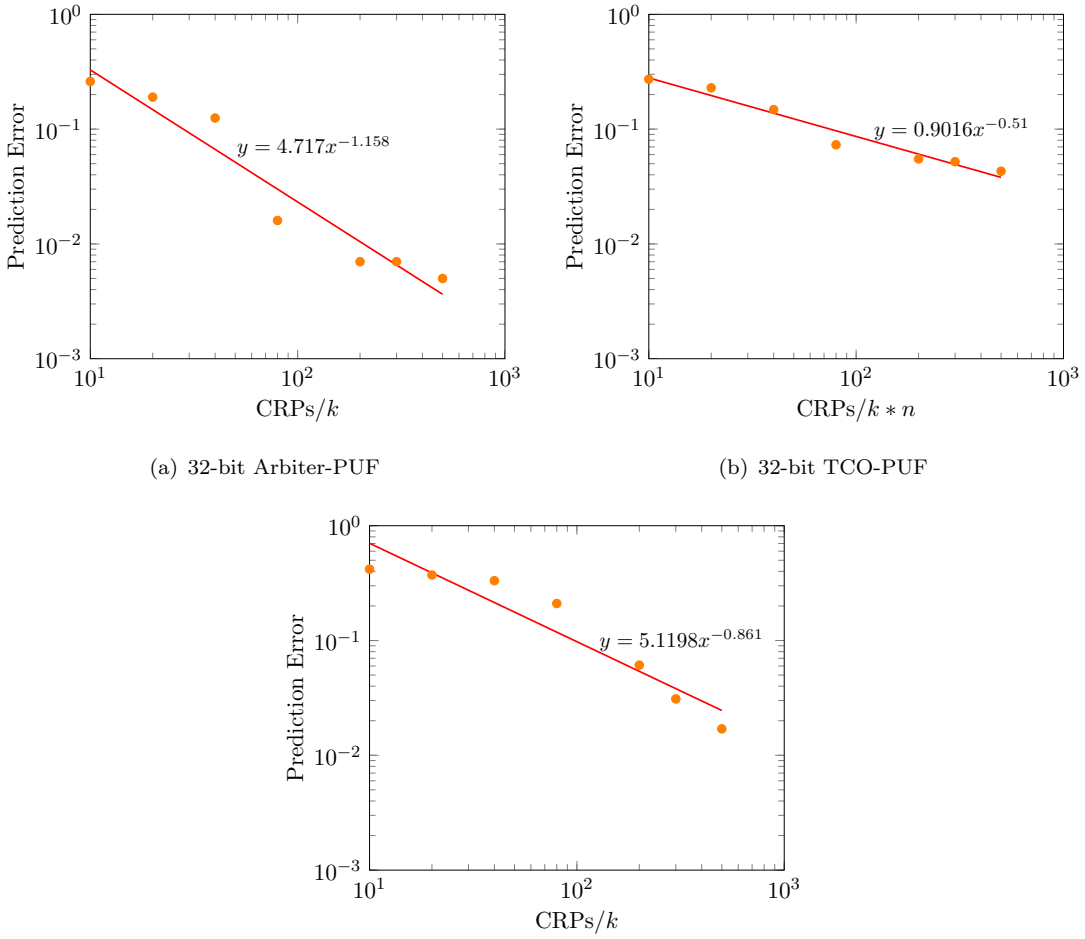


Figure 3.10: ML-attack on 32-bit TCO-PUF and comparison with other PUFs.

Figure 3.10 shows the prediction accuracy of the 32-bit TCO-PUF. The prediction accuracy increases as the number of CRPs in the training set increases. As can be seen from Figure 3.10, ANN was able to predict the CRP mapping of TCO-PUF with ≈96% prediction accuracy at 30000 CRPs. The predictability of the TCO-PUF is slightly lower than the SCA-PUF which has a prediction accuracy of 98% at 8000 CRPs [80]. For further comparison, the ML-attack results for the Arbiter-PUF and its derivative (as discussed in detail in Chapter 4) are included in Figure 3.10. The prediction accuracy of TCO-PUF is about 3% lower than Arbiter-PUF and XOR Arbiter-PUF, which is considered insignificant. Therefore, it can be concluded that the TCO-PUF which exploits a non-linear current-voltage behaviour is not resilient against an ML-attack.

It is always desired to extract the relevant parameters in order to have comparable results among different PUFs such as the required number of training CRPs to achieve a certain predictability value. One possible way is to find out the correlation of prediction error rate, CRPs, and bit-length of the challenge [28]. Prediction error rate is given as $\left(1 - \frac{\text{Prediction Accuracy}}{100}\right)$. Figure 3.11 shows the prediction error rate on the ratio of training CRPs and bit-length of the challenge for the aforementioned PUFs. Based on the approximation linear functional dependency as depicted in Figure 3.11, one can estimate the required number of training CRPs given a targeted prediction error rate for a particular PUF.



(a) 32-bit Arbiter-PUF

(b) 32-bit TCO-PUF

(c) 2-XOR 32-bit Arbiter-PUF

Figure 3.11: Prediction error rate on the ratio of training CRPs and bit-length of the challenge.

3.6 Impact of NBTI on PUFs

In Section 3.4.3.2, we have shown that the reliability of a PUF is affected by temperature and supply voltage fluctuations. However, CMOS device ageing, particularly NBTI, presents a more severe impact that can cause permanent reliability issues for a PUF over a prolonged time. As mentioned in Section 2.6.1, the V_{th} degradation due to NBTI is strongly dependent on the duty cycle, supply voltage and temperature. Herder *et al.*, [84] stated that a differential design technique can mitigate and cancel out the aforementioned first-order dependencies of ageing. Recall that a TCO-PUF employs a differential design technique which consists of two nominally identical transistor arrays and a comparator. The principle of the TCO-PUF in generating random responses is to compare voltage mismatches resulting from the threshold voltage variations of two identical transistor arrays. Its principle is similar to that of the Arbiter-PUF, except that the Arbiter-PUF compares the delay mismatch between two identical delay paths to generate random responses. In this section, we investigate the robustness of PUFs with differential architectures, such as the TCO-PUF and Arbiter-PUF, under the influence of NBTI ageing. As the TCO-PUF and the Arbiter-PUF are categorised as Strong PUFs (i.e., suitable for low-cost identification and authentication), it is interesting to investigate their potential to achieve low bit error rates (with ageing) , which is desirable for low-cost identification and authentication, as discussed in Section 2.7.1.

3.6.1 Ageing Evaluation Methodology

To evaluate the impact of NBTI, HSPICE MOSRA has been used to determine the reliability for each PUF after 10 years [89]. The MOSRA analysis can be divided into two phases, which are the pre-stress simulation phase and the post-stress simulation phase. Both phases can be executed in the same simulator run or independently, as needed. During the pre-stress simulation phase, MOSRA computes the stress (i.e., threshold voltage degradation ($\triangle V_{th}$)) of every single pMOS in the circuit based on the electrical simulation condition of each targeted device and its built-in T-D NBTI stress model, [89]. Subsequently, the $\triangle V_{th}$ values are translated to performance degradation at the circuit level during the post-stress phase. As explained in the Section, 3.4.1, Monte Carlo analysis has been used to model the process variations for PUFs. However, due to the limitations of the MOSRA flow, it is not possible to run Monte Carlo analysis within the same simulator run during the pre and post-stress phases. Hence, a simulation strategy has been implemented, as shown in Figure 3.12. In this strategy, the MOSRA flow is employed in the pre-stress phase. For post-stress simulation phase, the $\triangle V_{th}$ is extracted from the degradation file generated by MOSRA during the pre-stress phase and applied to the voltage source which is connected to the gate terminal of each pMOS transistor. This is called the aged netlist. By reducing the gate-to-source voltage (V_{gs}), we emulated the increase in V_{th} that is induced by NBTI. Then, the fresh and aged

netlists are simulated with Monte Carlo transient analysis and the results are compared. An important parameter that could influence the degradation rate during the pre-stress phase simulation is the activity factor. As claimed in [46], PUFs are not highly active in reality. For example, a PUF might be used only a few times in the IC identification application over its lifetime. Following [46], therefore, a 20% activation time is used throughout our analysis, unless otherwise stated.

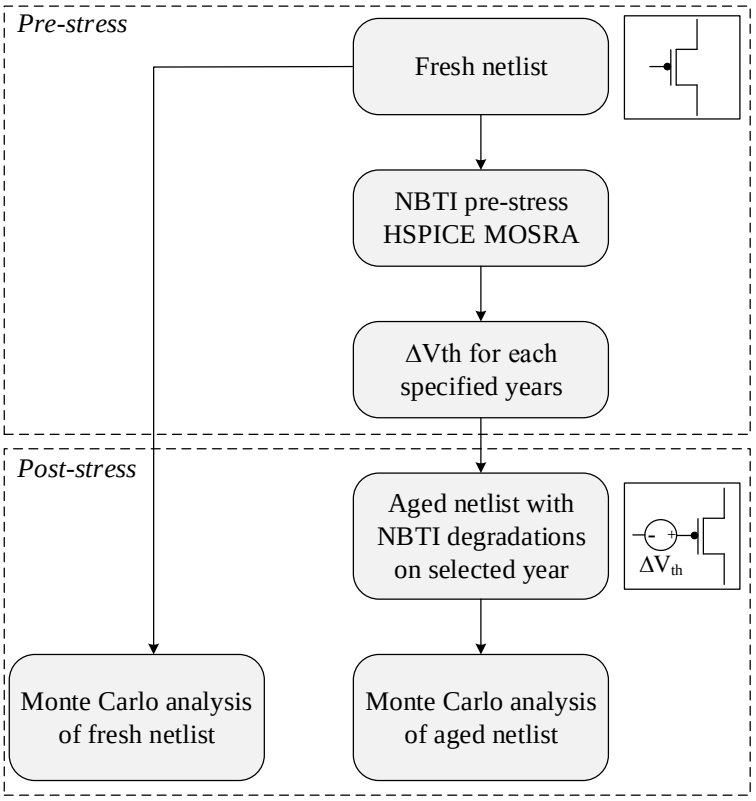


Figure 3.12: NBTI simulation strategy.

In the next section, we first validate our methodology by reproducing the bit error rate due to ageing for the RO-PUF. We expect to get a similar observation and comparable bit error rate as previously reported, [46, 48]. Then, we analyse and discuss the impact of ageing on the reliability of the TCO-PUF and Arbiter-PUF. Finally, we compare the results with the findings available in the literature.

3.6.2 NBTI Impact on RO-PUF

The RO-PUF circuit is implemented as in Figure 2.7, with 64×11-stage ROs, 2×64-to-1 multiplexer, 2×counter, and 1×8-bit comparator. A smaller number of RO pairs and inverting stages has been chosen in our study, as compared to [46] to significantly reduce the runtime, as we found that simulating an RO-PUF is time-consuming, mainly due to the size of the ROs. Monte Carlo simulation is used to model 100 RO-PUF instances.

A time interval of 50ns (i.e., the comparison time) is fixed, in which the counters count the number of oscillations and their resulting counter values are compared to generate a 1-bit response. In total, a 32-bit response is generated from each PUF instance.

In our study, the number of oscillations is used as a metric to quantify the effect of ageing, as illustrated in Figure 3.13, which shows the number of oscillations for 20 ROs (only 20 out of 64 ROs are shown for better visualisation) for a fixed time interval of 50ns. The gap between the two marks in the plot indicates the degraded number of oscillations experienced by each RO due to NBTI. As can be seen from Figure 3.13, generally, most of the ROs experience uniform degradation. Nevertheless, some of the ROs experience different degradation rates. For example, at time $t = 0$, RO#15 has a higher frequency compared to RO#14. However, after 10 years, RO#14 has a higher frequency compared to RO#15. From our analysis of ageing on the RO-PUF, we found that ageing causes different degradation rates of RO frequencies, as observed in previous studies [46, 48]. Recall from Section 2.6.2, both studies [46, 48] agreed that ageing causes different degradation on the RO frequencies. It is an interesting observation because RO-PUFs are built from arrays of identically laid-out ROs. The only difference is the process variations on each RO. Although no claim has been made in [48], Rahman *et al.*, [46] claims that because of process variations (i.e., second-order effect), each RO has a different degradation rate when the ROs are subjected to ageing. This situation leads to the generation of bit errors in the RO-PUF response.

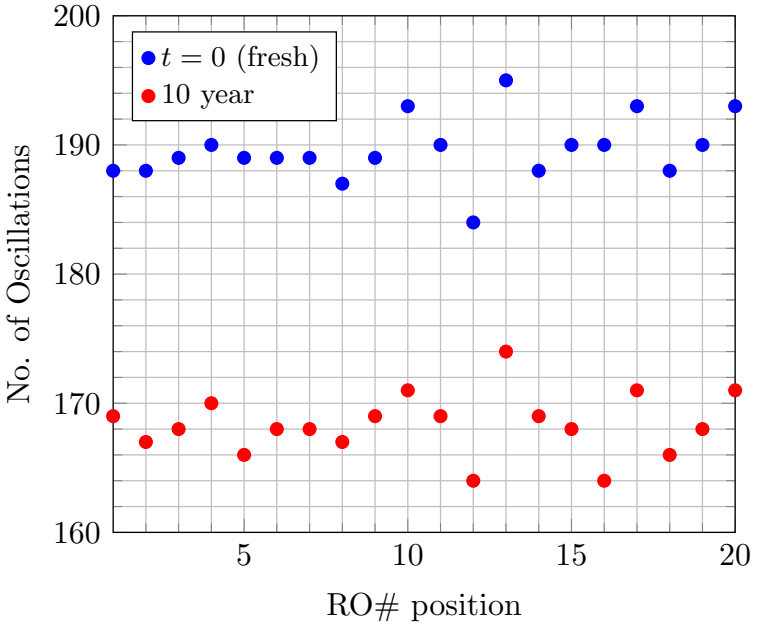


Figure 3.13: RO oscillation degradation under NBTI stress.

To quantify the bit error of the RO-PUF due to NBTI, a reference response has been measured at a room temperature of 25°C and a nominal supply voltage of 1.2V at time

$t = 0$ (fresh) and again after 10 years. Equation (2.5) has been used to calculate the HD between the two measurements, which indicates the total number of errors generated by each PUF. Figure 3.14 shows the distribution of errors due to ageing for 100 RO-PUF instances over 10 years. The average bit error rate is about 12.59% in 10 years and is comparable with the findings in [46].

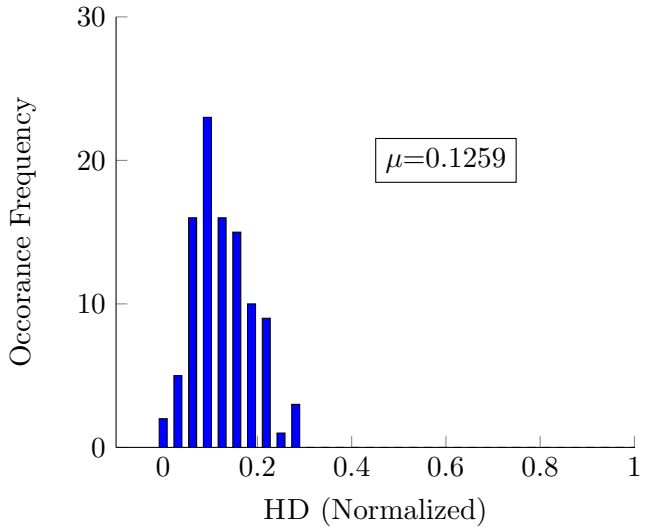
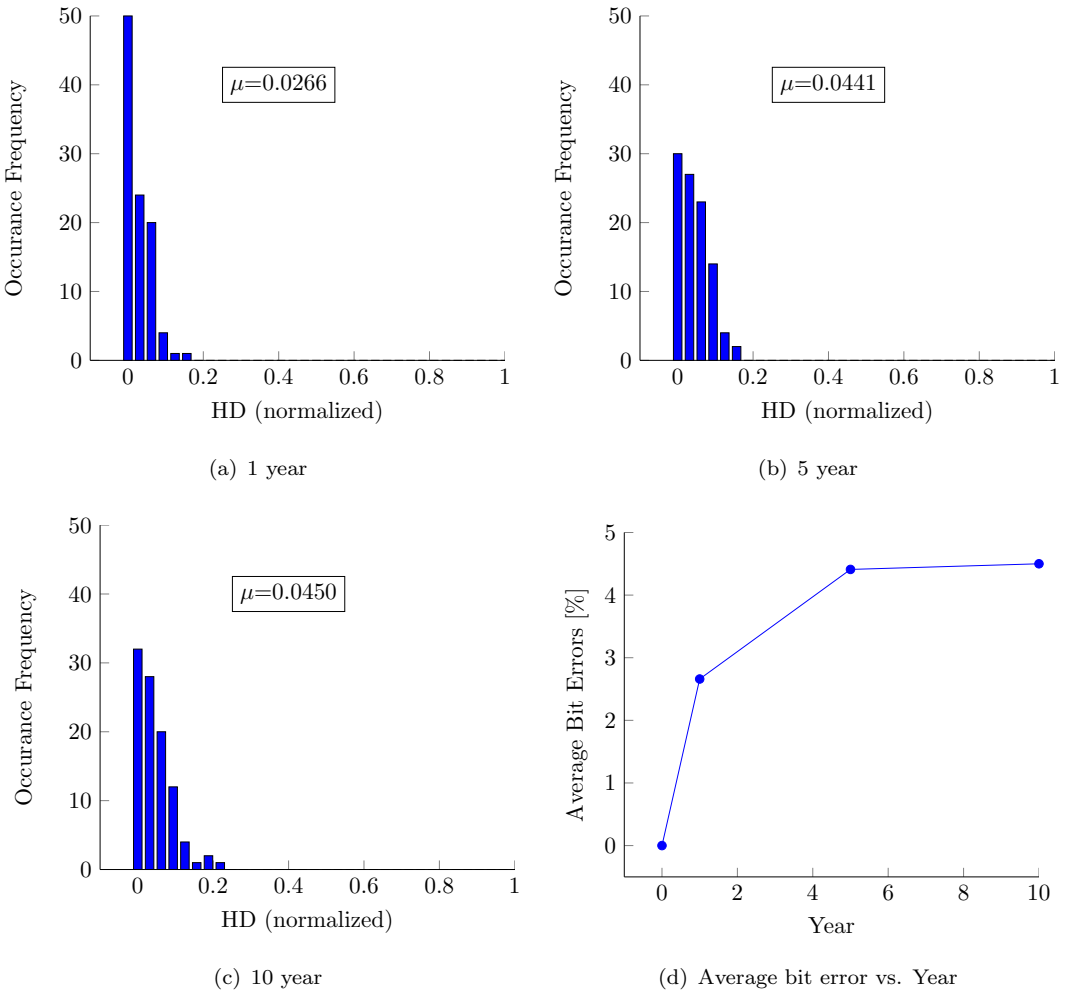


Figure 3.14: Distribution of bit errors due to NBTI for 100 RO-PUFs in 10 years with 20% activity factor.

3.6.3 NBTI Impact on PUF-based Differential Architectures

In Section 3.6.2, we have validated our methodology on the RO-PUF, and obtained similar observation and comparable bit error rate as previously reported [48, 46]. In this section, the impact of ageing on PUF-based differential architectures, namely the TCO-PUF and the Arbiter-PUF are compared. A similar setup as defined in Section 3.4.1 has been used to study the impact of NBTI on the TCO-PUF. The TCO-PUF has been subjected to ageing by using the methodology described in Figure 3.12. A 32-bit response has been measured at time $t = 0$ (fresh) and again after 1, 5 and 10 years. Equation (2.5) is used to measure the number of bit errors for 32-bit fresh and aged responses for all 100 instances of the TCO-PUF. Figure 3.15 shows the distribution of bit errors for the TCO-PUFs due to NBTI after 1, 5, and 10 years with 20% activity factor. By taking the average error value in Figure 3.15(a), 3.15(b), and 3.15(c), the effect of ageing on TCO-PUF over the years can be observed clearly in Figure 3.15(d). On average, the bit error increases with time giving 2.66%, 4.41%, and 4.5% bit errors after 1, 5, and 10 years, respectively.



(a) 1 year (b) 5 year

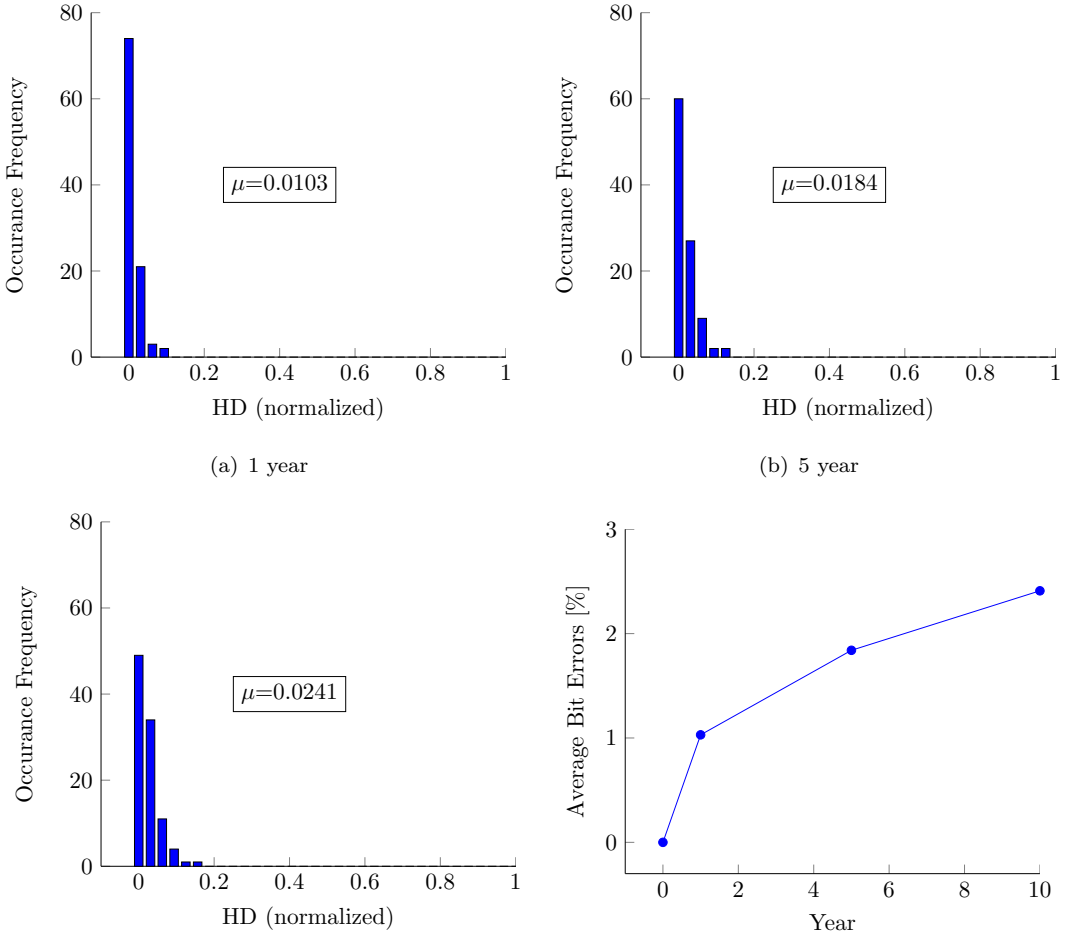
(c) 10 year (d) Average bit error vs. Year

Figure 3.15: Distribution of bit errors due to NBTI for 100 TCO-PUFs in 1, 5, and 10 years with 20% activity factor.

In both transistor arrays, TCOA and TCOB of the TCO-PUF, the common inputs or challenges are applied to non-stochastic pMOS devices and a similar fixed bias voltage, V_{gp} is applied to all stochastic pMOS devices. Therefore, all pMOS devices in both transistor arrays experience a similar duty cycle. In the comparator structure, pMOS devices are used as reset transistors (M7 and M8) and as latch regeneration transistors (M5 and M6) in cross-coupled inverters as shown in Figure 3.4. Both of the reset transistors are driven by the same clock. For the latch regeneration transistors, the degradation due to NBTI depends on the frequencies of 0's at the *outN* and *outP* nodes. Section 3.4.3.3 has demonstrated that the uniformity of the TCO-PUF has an average value of slightly higher than 50%. This indicates that both pMOS latch regeneration transistors in the cross-coupled inverters experience an approximately symmetrical stress. All of the above shows that all pMOS devices in the TCO-PUF, experience a similar duty cycle, resulting in similar NBTI degradation. Hence, a differential design technique can be a mechanism to cancel out first-order NBTI dependencies on the duty cycle. Ideally,

it is expected that TCO-PUF will give the same response even after 10 years. However, there is a second-order effect that caused bit errors for the TCO-PUF, as discussed in Section 3.6.2. Overall, TCO-PUF shows good resiliency against NBTI effects.

Another PUF implementation that uses a differential design technique is an Arbiter-PUF. To study the impact of NBTI, the Arbiter-PUF has been implemented as in Figure 2.3 with $k = 16$ stages and a simulation setup as defined in Section 3.4.1. In our simulation, an SR-latch has been used as the arbiter block. Figure 3.16 shows the distribution of bit errors for 100 Arbiter-PUF instances when they are subject to NBTI after 1, 5, and 10 years. The average bit error over 10 years is detailed in Figure 3.16(d) which shows that the impact of NBTI becomes more significant with time, resulting in an increase in the bit error rate. On average, the bit error rates are 1.03%, 1.84%, and 2.41% after 1, 5, and 10 years respectively.



(a) 1 year

(b) 5 year

(c) 10 year

(d) Average bit error vs. Year

Figure 3.16: Distribution of bit errors due to NBTI for 100 Arbiter-PUFs in 1, 5, and 10 years with 20% activity factor.

To analyse the impact of NBTI on every pMOS transistor in the Arbiter-PUF architecture, a single block of the switching component is considered, with the propagation paths of the rising pulse as shown in Figure 3.17. For c_1=0, the top_0 and bot_0 signal propagations are indicated by the red path, whereas the blue path represents the signal propagation when c_1=1. In this simulation, random challenge inputs are generated with a probability of 0.5. Therefore, the pMOS in each transmission gate in both multiplexers experiences similar degradation due to NBTI. However, as far as the propagation paths of top_0 and bot_0 are concerned, regardless of whether the probability of the challenge inputs is 0.5 or not, both signals go through a similar path (i.e., either the red or blue path), which experiences similar NBTI degradations with respect to the transmission gates. The inputs for top_0 and bot_0 are common, therefore, $inv2$, $inv3$, and $inv4$ in each multiplexer always experience similar degradation effects from NBTI. A similar case exists for $inv1$ in both muxes.

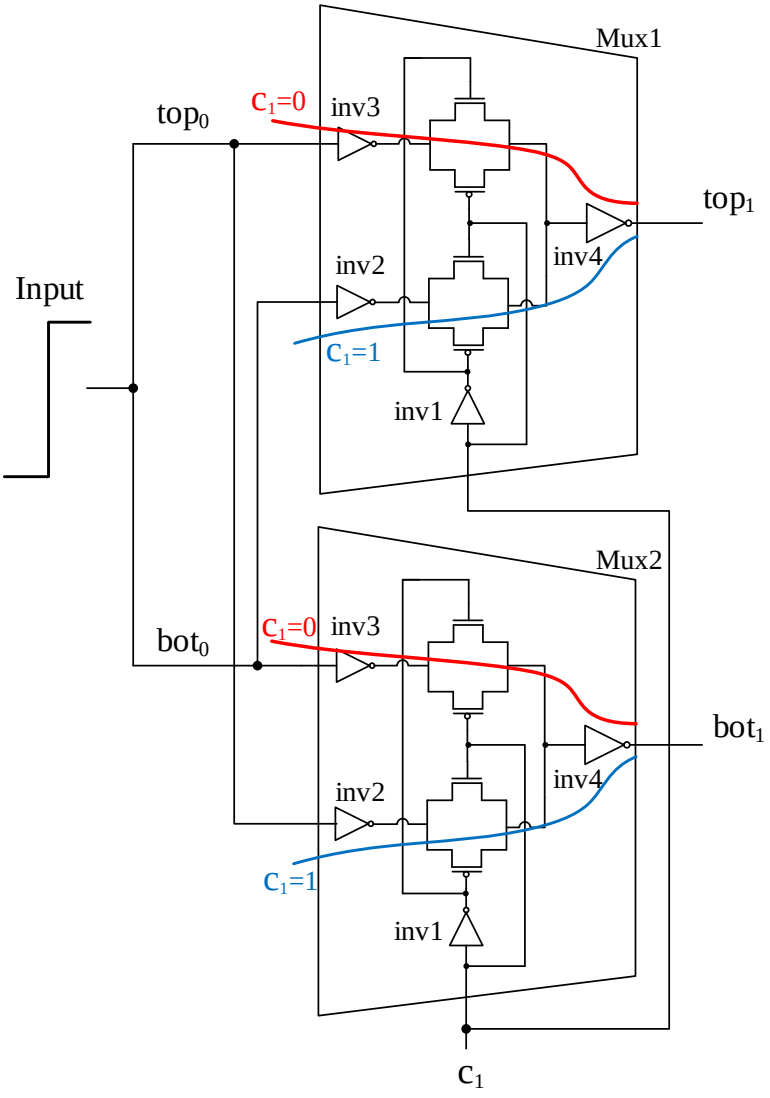


Figure 3.17: Path propagation in switching component [3].

Based on the path propagations in switching component blocks, the Arbiter-PUF circuit can be further simplified, as shown in Figure 3.18. The basic structure of the Arbiter-PUF is two parallel chains of inverters in series. Both inverter chains experience similar NBTI degradation, therefore it is expected that the response of the Arbiter-PUF is robust against NBTI effects.

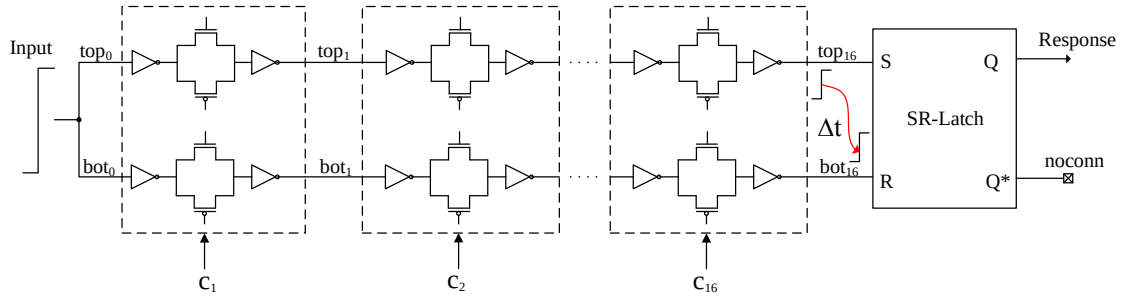
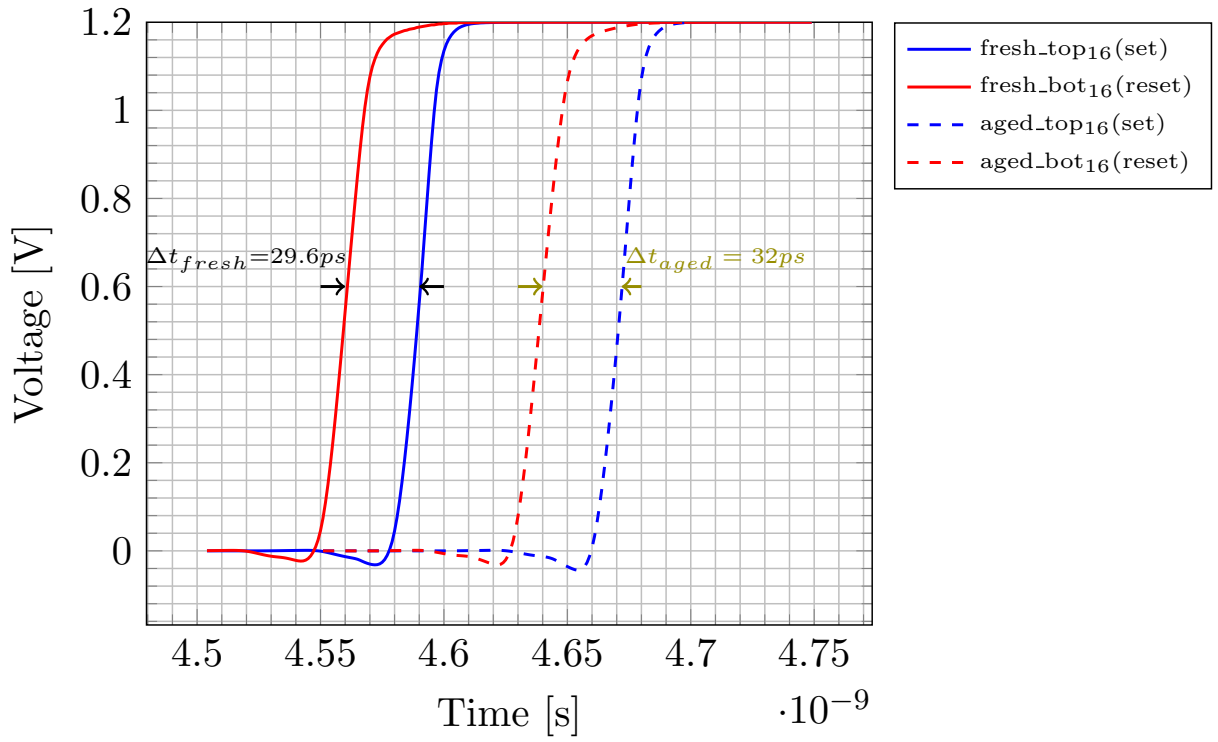


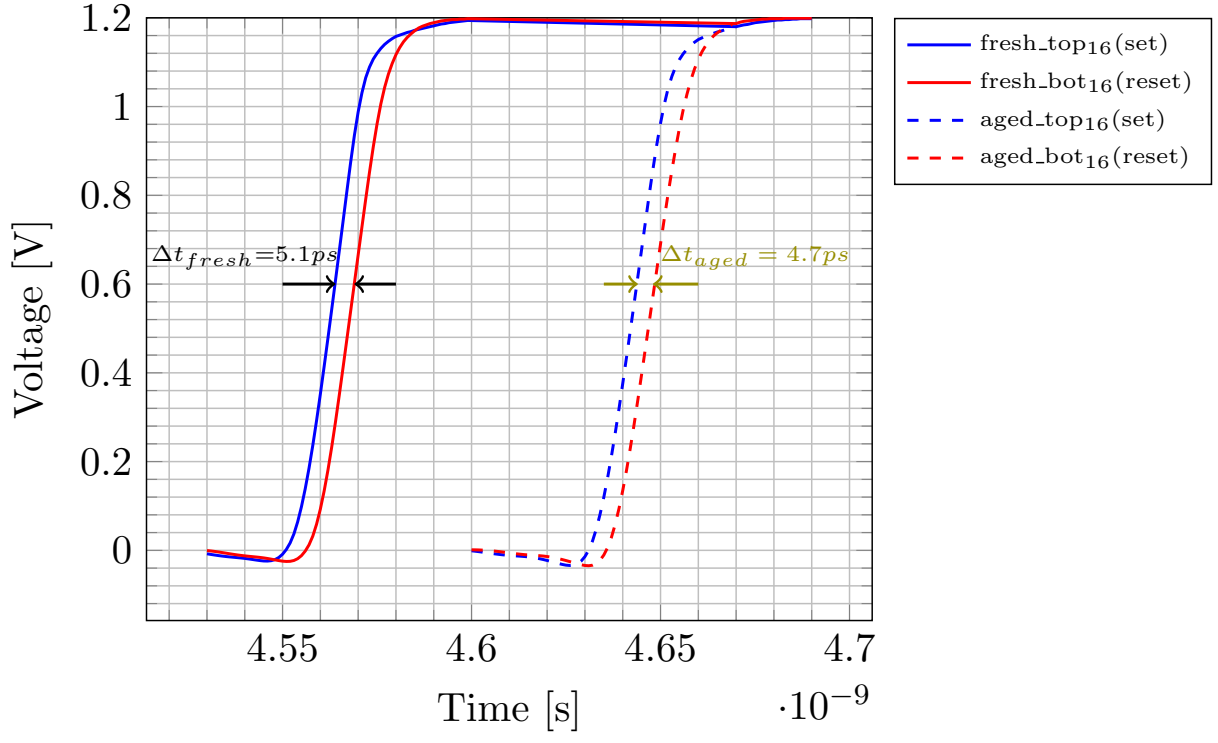
Figure 3.18: Simplified circuit of Arbiter-PUF.

However, we observe different degradation rates in the propagation delays of the two parallel signals under NBTI stress (i.e., a similar observation, as discussed in Section 3.6.2 and in [46, 48]), resulting in a delay difference at $t = 0$ and after ageing. Based on our analysis, due to the different degradation rates, the degraded delay of the two parallel signals can either increase or decrease the arbitration window of the SR-latch, given as Δt in Figure 3.18. To illustrate this, Figures 3.19(a) and 3.19(b) show the rising pulses before the SR-latch of two parallel signals of two PUF instances, respectively, under a similar challenge at $t = 0$ and after ageing. While a majority of the occurrences show an increase in Δt, which also helps to increase the reliability of the response, a decrease in Δt occurs a few instances, which has the potential to generate a bit error in the Arbiter-PUF response since the decrease can lead to a metastable state. Based on our simulation, the setup time for the SR-latch is 5ps. As depicted in Figure 3.19(b), the degraded arbitration window, Δt_{aged}, is below the setup time of the SR-latch. Hence, the PUF enters a metastable state. The error happens when a metastable state resolves to a stable state which is opposed to the generated response at $t = 0$. We observed only a few events of the type shown in Figure 3.19(b). Moreover, due to the symmetric circuit topology of the SR-latch, it experiences symmetrical NBTI stress. All of the above explains the low bit error rate due to NBTI stress for the Arbiter-PUF.

As reported in [28], increasing the delay stages, k, in the Arbiter-PUF has a positive impact on the reliability of PUF's responses. Based on this, we assume that as k increases, the randomness due to process variations increases and reduces the probability of the metastable state occurring. We proved the above claim in our ageing simulation, by showing that the 32-bit Arbiter-PUF reduces the probability of the event of Figure 3.19(b) happening, therefore, it achieves on average low bit error rates as compared to the 16-bit Arbiter-PUF, illustrated in Figure 3.20. Overall, an Arbiter-PUF shows good resilience against NBTI effects.



(a) PUF instance 1



(b) PUF instance 2

Figure 3.19: Arrival time of a rising pulse before the SR-latch under NBTI stress for two Arbiter-PUF instances with a similar challenge.

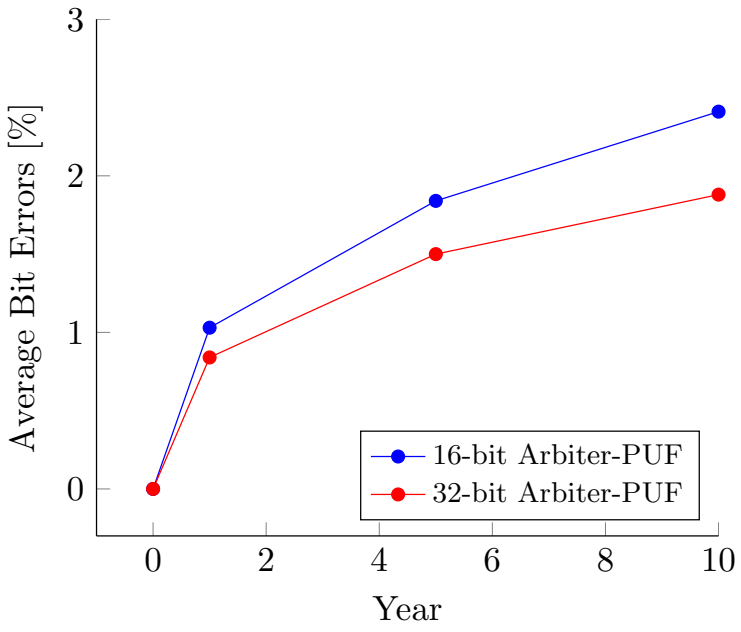


Figure 3.20: Average bit error rates for 16-bit and 32-bit Arbiter-PUFs.

Based on the findings for the bit error rate for the RO-PUF (Section 3.6.2), TCO-PUF and Arbiter-PUF, the different degradation rates of the delay due to NBTI are the cause of errors, but the errors produced are much higher in the case of the RO-PUF. Revisiting the RO-PUF architecture, a single RO is built from a chain of inverters in series and a feedback loop which makes it oscillate (see Figure 2.7). Figure 3.18 also shows that an Arbiter-PUF can be simplified as a chain of inverters, similar to an RO circuit but without a feedback loop. Hence, we infer that the significant bit error rates experienced by the RO-PUF are due to the feedback loop, which magnifies the different degradation rates of the delay due to NBTI.

Table 3.1 compares the average percentage of bit errors for the TCO-PUF and the Arbiter-PUF with other types of PUF as reported in the literature. The comparison shows that the TCO-PUF is $\approx 2.8\times$ and the Arbiter-PUF is $\approx 5\times$, better than the RO-PUF for a 20% activation time, after 10 years, as reported in [46]. In comparison with the experimental ageing study on RO-PUF, [48], although the test parameters (i.e., activity factor and year) are more stringent, Maiti *et al.*, [48] achieve lower bit errors compared to the ageing simulation study on RO-PUF, [46]. A mitigation technique to reduce the ageing impact on RO-PUF has been proposed, [46]. The architecture is called an ageing-resistant RO-PUF (ARO-PUF) and has an average bit error rate of 3.83% over 10 years. However, the Arbiter-PUF and TCO-PUF are inherently ageing-resilient because of the architecture which uses a differential design technique. The experimental ageing findings, [47, 39], are also listed in Table 3.1. Though the ageing test conditions are slightly different from ours, it is still appropriate to compare the impact of ageing on the TCO-PUF and the Arbiter-PUF with that for a memory PUF.

Table 3.1: Ageing impact comparison

PUF Type	Technology (nm)	Ageing Mechanism	Measurement Condition	Activity Factor (%)	Year	Average Bit Errors (%)
RO-PUF [46]	90	BTI and HCI	25°C, 1.2V	20	10	12.76
ARO-PUF [46]	90	BTI and HCI	25°C, 1.2V	20	10	3.83
RO-PUF [48]	90	NBTI	25°C, 1.2V	≈90	13	8.6
32-bit Arbiter-PUF	65	NBTI	25°C, 1.2V	20	10	2.41
32-bit TCO-PUF	65	NBTI	25°C, 1.2V	20	10	4.5
SRAM-PUF [39]	65	NBTI	25°C, 1.2V	100	4.5	min. 7 to max. 8
SRAM-PUF [47]	90	NBTI	20°C, 1.2V	100	4.7	<14

3.7 Summary

In this chapter, a sub-threshold PUF architecture known as TCO-PUF, which exploits the non-linear dependency of current and voltage in the sub-threshold region, is characterised. The characterisation results of the 32-bit TCO-PUF show that the mean values of uniqueness and uniformity are close to 50% but it suffers from lack of randomness in its response, hence it has a high standard deviation in the distribution of both metrics. On average, with temperature variations of -40^{o}C to 85°C and a supply voltage of 1.2V ± 10%, a 32-bit TCO-PUF achieves 92.38% reliability. An ML-attack has been performed using ANN and the result shows that the response of the 32-bit TCO-PUF can be predicted with a very high accuracy of about 96% at 30000 CRPs.

Moreover, the impact of CMOS device ageing, in particular, NBTI is investigated on the TCO-PUF and the Arbiter-PUF. The results show that the reliability of the 32-bit TCO-PUF and the 32-bit Arbiter-PUF decrease with age and they experience bit errors of about 4.5% and 2.41%, respectively, after 10 years. In comparison with other types of PUF such as RO-PUF and SRAM-PUF, TCO-PUF and Arbiter-PUF are inherently ageing-resilient as they use a differential design technique.

Chapter 4

A Lightweight Technique for ML-Attack Resistant PUFs

As discussed in Section 2.7.1, Strong PUF based authentication is suitable for RFID tags in which a challenge-response protocol and an HD threshold, ϵ, are introduced to avoid using ECC to reduce the area and power consumption. These extremely lightweight PUF-based RFID tags can be promoted as a secure alternative to memory-based RFID tags and are proposed as anti-counterfeiting for medical drugs and supply chain control [55]. Also discussed in Section 2.7.1, to achieve an optimum value of ϵ, the Strong PUF must be reliable and secure against an ML-attack. The reliability of Strong PUFs with differential architecture under ageing has been analysed in Chapter 3. The Arbiter-PUF has shown to be more robust against ageing in 10 years. Therefore, an Arbiter-PUF is used as a case study in this chapter for ML-attack analysis. Besides, recent literature, [8, 35, 34, 57, 73, 90, 91] were focused on increasing the resilience of an Arbiter-PUF and its derivative against an ML-attack. Therefore, using an Arbiter-PUF as a case study enables a good comparison of the ML-attack susceptibility with the previous studies, which will be discussed later in Section 4.4.5. In this study, we focus on such commercial resource-constrained PUF-based RFID tags in which the Arbiter-PUF combined with a permutation technique is used to provide a certain level of security against an ML-attack. Hence, it is not worthwhile for attackers to spend their resources counterfeiting these products. Our goal is to improve the security of an Arbiter-PUF against an ML-attack and to achieve a low-cost implementation. The Arbiter-PUF has been constructed as previously proposed, [12], using a TSMC 65-nm technology. The main contributions of this chapter are:

1. We show that using a challenge permutation technique can alter the output transition probability of the Arbiter-PUF, resulting in an increase of the resiliency against an ML-attack. A challenge permutation technique can be implemented by obfuscating the routing at the input and introduces no additional hardware.

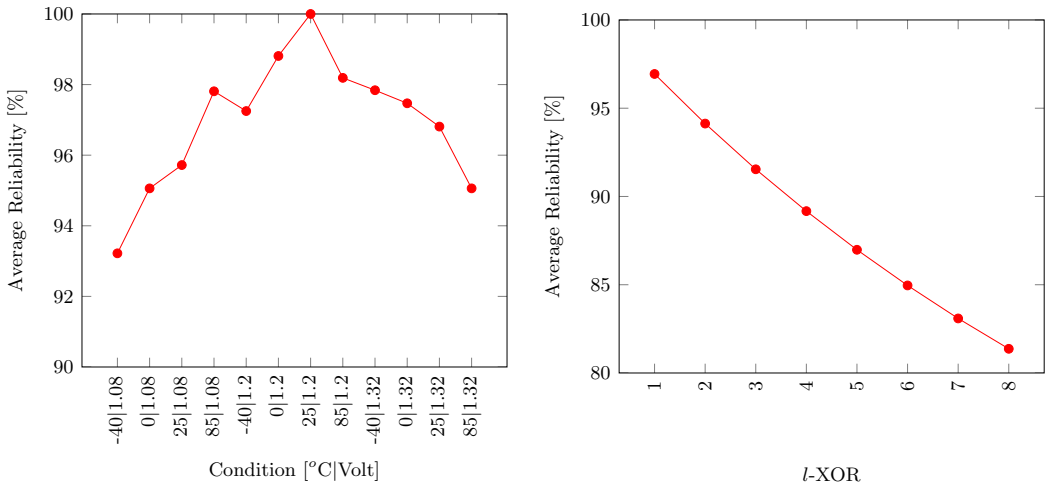
Unlike the XOR technique, this technique has no impact on the reliability of the Arbiter-PUF.

2. We also show that a random permutation is required to maximise the complexity of the output transition probability of the Arbiter-PUF. Hence, higher unpredictability for an Arbiter-PUF can be achieved.

This chapter is organised as follows: Section 4.1 describes the motivation to explore a new technique against an ML-attack. The experimental methodology is discussed in Section 4.2. Section 4.3 presents the properties of the Arbiter-PUF. Section 4.4 describes in detail the proposed permutation technique including the area, power, and predictability compared to the previous techniques. Finally, this chapter is concluded in Section 4.5.

4.1 Motivation

As reviewed in Section 2.8, an ML-attack is the most plausible way to attack Strong PUFs since it offers great advantages to the adversary in terms of cost efficiency and high prediction accuracy. ML-attack resistance describes the complexity of the challenge to response mapping for a particular PUF. As defined in Section 2.3.2, Strong PUFs are a type of PUF that is able to generate an exponential number of CRPs, as in the Arbiter-PUF. However, the Arbiter-PUF can be easily modelled by ML due to the linear addition of the inherent delay values [36]. Several PUFs have been derived from the Arbiter-PUF to introduce non-linearity into the mapping function of the CRPs, such as the Feed-forward Arbiter-PUF [30], XOR Arbiter-PUF [22], and Lightweight-PUF [31]. However, ML techniques are still able to model them with high accuracy [73]. One might disable the ML-attack by implementing the XOR Arbiter-PUF and Lightweight-PUFs with $l \geq 8$ (i.e, the number of XORs in output network) [28, 73]. Nevertheless, the reliability of the Arbiter-PUF reduces as l increases. To demonstrate the reliability of the XOR Arbiter-PUF, first, the average reliability of the Arbiter-PUF has to be found. A simulation setup as in Section 3.4.1 is used to evaluate the reliability for the 32-bit Arbiter-PUF under temperature variations from -40^oC to 125^oC and 1.2V$\pm$10% supply voltage fluctuations; the result is depicted in Figure 4.1(a). The average reliability of the 32-bit Arbiter-PUF (i.e., l=1) over temperature and supply voltage variations is 96.94%, as depicted in Figure 4.1(b).



(a) 32-bit Arbiter-PUF

(b) l-XOR 32-bit Arbiter-PUF

Figure 4.1: Reliability for 32-bit Arbiter-PUF under temperature and supply voltage variations.

As the average reliability for the 32-bit Arbiter-PUF is now known, the average reliability of the XOR-Arbiter-PUF can be estimated using Eq. (4.1) (see Appendix C.1 for the derivation).

$$P(t_{XOR} = reliable) = 1 - (P(A) + P(B) - 2P(A)P(B)) \tag{4.1}$$

Figure 4.1 shows as l increases to 8, the average reliability reduces to ≈80%. To realise the implementation of Strong PUF based authentication, as discussed in Section 2.7.1, the HD threshold, ϵ must be as small as possible to avoid false positives and to reduce the vulnerability against ML-attack. On the other hand, it must be large enough to tolerate the bit error rates and reduce false negatives. Evidently, there is a trade-off between security and reliable performance for an XOR network technique. To achieve high security against an ML-attack, one has to use a large number of XORs but one has to increase the ϵ value to reduce the probability of false negatives. However, the high value of ϵ nullifies the high resilience against an ML-attack achieved using an XOR network technique. The parallel architecture of the XOR network technique further increases the total area. Similar considerations applied for Lightweight-PUFs.

Recently, Delvaux *et al.*, [10] surveyed the previously proposed scheme of Strong PUF based authentication. From their survey, most of the proposed schemes require a hash function and/or NVM which obviously have a large area overhead. A few scheme used a PUF to feed another PUF but suffers from reliability issues, similar to the XOR network technique, since it is known that PUF outputs are susceptible to temperature and supply

voltage variations. The use of a hash function to increase the security of Strong PUFs against ML-attacks started with the concept of the Controlled PUF, introduced by Gassend *et al.*, [27]. As explained in Section 2.3.2, a Controlled PUF uses a secure one-way hash function to break the relationship between challenges and responses as illustrated in Figure 4.2.

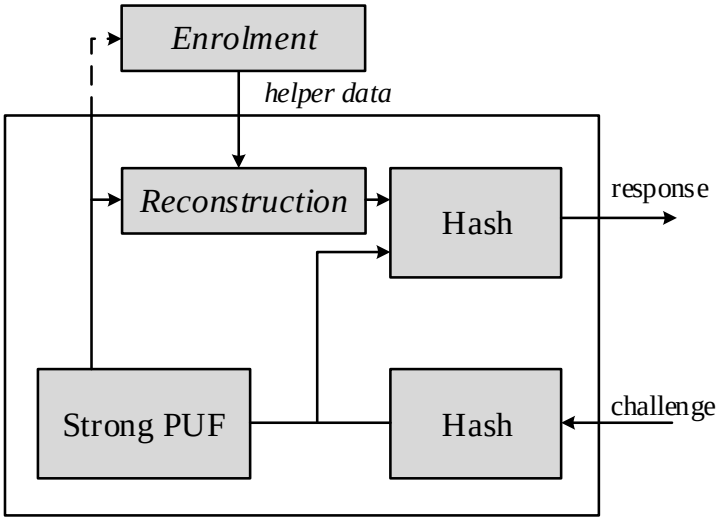


Figure 4.2: The concept of Controlled PUF [27].

With the assumption of an adversary which is limited to non-invasive CRPs measurement, the Controlled PUF successfully disables the ML-attack because it is known to be hard or almost impossible to invert a one-way hash function [28]. However, a one-way hash function consumes large area and is power-hungry. To demonstrate this, we compiled readily-available SHA-1 and SHA-256 Verilog code, [92], using Synopsys Design Compiler. The area and power are listed in Table 4.1. Clearly, one-way hash functions are too costly although they improve security for low-cost Strong PUF based authentication. It is important to note that the SHA-1 has been broken [93]. The results in Table 4.1 only prove that the hash functions are costly for resource-constrained devices. If a PUF-based system needs to use the hash function, consider using safer alternatives such as SHA-256, or SHA-3.

Table 4.1: Area and power of hash function

Type	Area [GE]	Power [mW]
SHA-1	9567	1.256
SHA-256	12980	1.688

Several works focus on how to increase the resilience of an Arbiter-PUF against an ML-attack. In recent studies, Ye *et al.* proposed an obfuscation logic based PUF (OPUF)

[90] and randomised challenge PUF (RPUF) [91] to increase the resiliency of an Arbiter-PUF against the ML-attack. However, an OPUF suffers reliability issues as described above since it adapts the XOR network technique at the output stage. RPUF also has a potential issue of bias in the random number generator (RNG), which is used to randomise the challenge. Elsewhere, Gao *et al.*, [8] proposed an Obfuscated-PUF (OB-PUF) in which a partial challenge is sent by the verifier to the OB-PUF (i.e., the prover). Subsequently, within an OB-PUF, a partial challenge is padded with a random pattern generated by RNG to make up a full-length challenge. Earlier, Rostami *et al.*, [57], proposed a sub-string matching technique in which only a subset of PUF response strings is sent to the verifier during authentication. Generally, both works, [8, 57], use the same technique by only exposing a subset of either challenges or responses. However, this might increase the authentication time to run the matching algorithm, as well as the area, on the verifier side. One might argue, however, that the area is not a concern since the verifier has always been assumed to be resource rich. As discussed in Section 2.4.5, mixed-signal PUFs which adapt the architecture of Arbiter-PUF have been proposed, such as the Current Mirror-PUF, [35] and the VTC-PUF, [34]. Both exploit the non-linearity in the current mirror and voltage transfer characteristics, respectively, to increase the resilience against an ML-attack. Further, these studies, [35, 34] show that the non-linearities introduced in the respective circuits reduce the predictability by about 25% as compared to the Arbiter-PUF in which SVM has been used for the ML-attack analysis.

From all of the above, the resilience of the Arbiter-PUF against an ML-attack can be improved. However, the area and/or reliability need to be compromised, which undermines the realisation of low-cost Strong PUF based identification and authentication. Motivated by this, we explore a simple yet effective technique, which is a challenge permutation. This technique will be implemented and evaluated for an Arbiter-PUF.

4.2 Methodology

In this section, we discuss the simulation setup for CRP generation, the PUF configuration, the ML algorithm, and the threat model used to evaluate the challenge permutation technique.

4.2.1 CRP Generation

16-bit, 32-bit, and 64-bit Arbiter-PUFs have been implemented in a low-k 65-nm technology node and simulated using the BSIM4 (V4.5) transistor model, with a nominal supply voltage of 1.2V and at room temperature of 25^{o}C. An SR-latch has been used as the arbiter block. Intrinsic variations such as oxide thickness and threshold voltage

are modelled in Monte Carlo simulations using the built-in fabrication standard statistical variation (3σ variations) in the technology design kit. An arbitrary challenge is further applied to the Arbiter-PUF to generate a corresponding response. For a 16-bit Arbiter-PUF, a maximum length of 65534 CRPs (excluding all 0's and 1's) has been generated for ML-attack analysis. A simulation of a maximal length for the 32-bit and 64-bit Arbiter-PUFs is prohibitive, and therefore, only a total of 32000 CRPs has been generated for the ML-attack analysis.

4.2.2 PUF Configuration

There are two ways to construct an m-bit response from a 1-bit response from the Arbiter-PUF:

1. A challenge is applied to m parallel Arbiter-PUFs; this could simply increase the area overhead of the whole system.
2. A challenge is used as a seed for an LFSR and it generates m sub-challenges sequentially to be applied to the underlying Arbiter-PUF. Hence, it improves the area overhead significantly but the evaluation time increase by a factor of m.

In this study, we chose the second configuration because it offers area efficiency. An increase in the evaluation time has no major impact as Lee *et al.*, [12], shows that a silicon implementation of a 64-bit Arbiter-PUF using a TSMC 180-nm has a potential throughput of 20Mb/s. A Fibonacci LFSR configuration has been used in this study and Table 4.2 lists the generator polynomials for maximal-length sequence generation.

Table 4.2: Generator polynomials for maximal-length sequences

Type	Generator Polynomial
16-bit	$X^{16} + X^{15} + X^{13} + X^4 + 1$
32-bit	$X^{32} + X^{31} + X^{30} + X^{10} + 1$
64-bit	$X^{64} + X^{63} + X^{61} + X^{60}$

4.2.3 Artificial Neural Network

As discussed earlier in Section 2.8.3, the ANN is one of the best performing ML algorithms to solve non-linear problems. Therefore, the ANN has been chosen and the built-in ANN algorithm in MATLAB (R2016b) (see Appendix A.6 for the MATLAB script of an ML-attack) has been used in this study. Therefore, this section describes

an overview of the basic concepts of the ANN and how it is setup to perform an ML analysis of a Strong PUF.

An ANN is an adaptive system formed from interconnected computing nodes called neurons. A typical structure of an ANN is a feed-forward network which can be constructed as a single-layer perceptron (SLP) or a multilayer perceptron (MLP). SLPs are the simplest form of feed-forward network and are only capable of solving linearly separable problems. MLPs are required for solving non-linear problems [94] and therefore have been chosen for our ML analysis. The ML analysis of Strong PUF can be considered as a binary classification problem since it has a single target, either 0 or 1. Hence, a 2-layer network structure of MLPs as illustrated in Figure 4.3 has been used in this study. It consists of two layers with tunable weights and bias which are a hidden layer and an output layer. Each node in one layer is connected to all nodes in the next layer and there are neither connections between nodes in the same layer nor feedback between different layers.

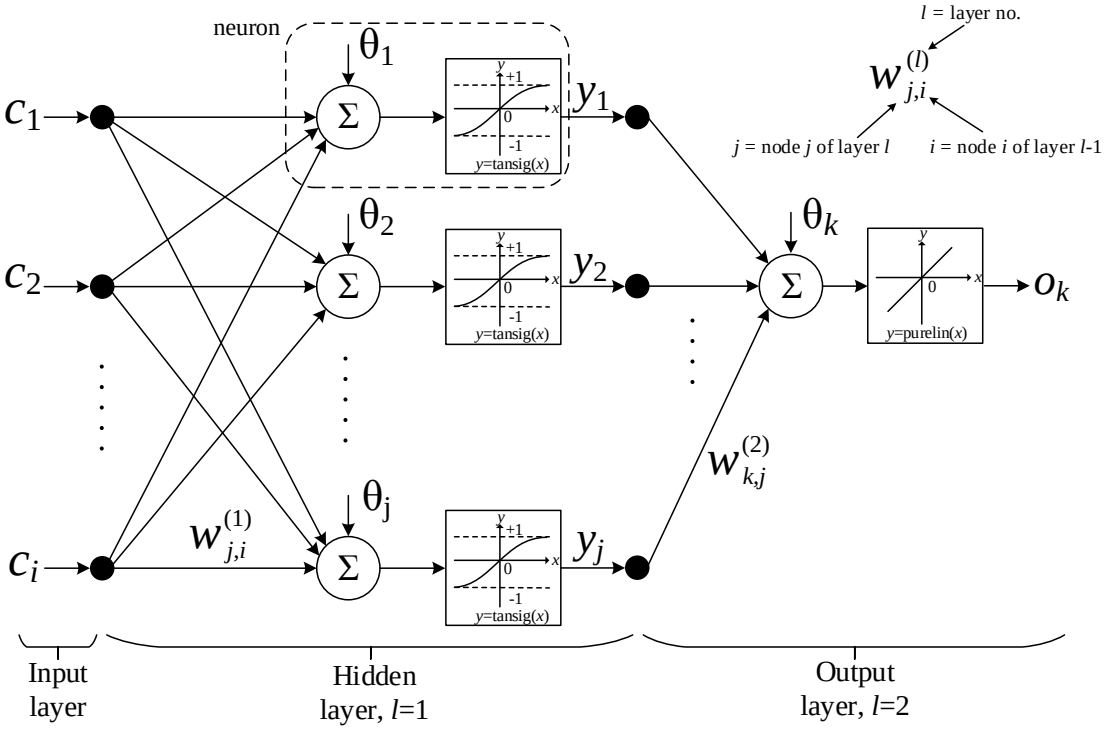


Figure 4.3: MLP feed-forward network structure for binary classification problem.

Except for the input nodes, each node is a neuron that uses an activation function, $f(.)$ to limit the output range. In the hidden layer, tan-sigmoid activation function is used given as $f(x) = \frac{2}{1+e^{-2x}} - 1$. This function is bound to the range (-1,1), as shown in Figure 4.3. The linear transfer function is used at the output layer. Based on Figure 4.3, the activation of node j in the hidden layer, y_j and node k in the output layer, o_k, respectively can be computed as:

$$y_j = f(\sum_i w_{j,i}^{(1)} c_i + \theta_j) \tag{4.2}$$

$$o_k = f(\sum_j w_{k,j}^{(2)} y_j + \theta_k) \tag{4.3}$$

where w is an assigned weight (i.e., the detail of the notation w can be seen in Figure 4.3) to every connection between nodes and θ is an additional bias. Following the rule of thumb in [94] for selecting the number of neurons in the hidden layer, we found that 32 neurons is an optimal number and this is used throughout all the ML-analysis. During training, an error resulting from the difference between a predicted and observed value is propagated back through the network and the neuron's weights and bias are adjusted and updated. The training process stops when the prediction error reaches a predefined value or a predetermined number of epochs (i.e., iteration) is completed. Once the MLP is trained, the MLP is ready for use to compute the outputs of new input patterns. Based on our ML experiments, resilient back-propagation has been chosen as the best training algorithm considering the prediction accuracy and fast convergence time, consistent with the explanation elsewhere, [56]. A test set has been chosen randomly and it is not part of the training data. For the 16-bit Arbiter-PUF, a total of 2534 CRPs is used as a test set. For the 32-bit and 64-bit Arbiter-PUFs, a total of 2000 CRPs is used as a test set.

4.2.4 Threat Model

An adversary may have several motives for attacking deployed low-end devices, such as gaining secure access through RFID or to counterfeit products. Becker, [55], shows that an attacker who is in possession of the PUF-based RFID tag and has an access to the primary inputs can collect CRPs through non-invasive measurement and perform an ML-attack. The parameters derived by the ML-attack are further programmed on a programmable RFID smart-card emulator and a PUF tag was successfully cloned. However, prior to the ML-attack, software reverse-engineering was performed to discover the configuration of the PUF model in the reader, such as the exact LFSR and XOR-ing functions. As the internal configurations are known, the ML-attack has been performed based on the mapping of the internal CRPs.

Based on the above situation, we derive our threat model. In order to achieve one of the adversary goals, we assume that the attacker is restricted to a non-invasive CRP measurement. The attacker only has an access to the primary inputs of the PUF-based RFID tag, and can apply a polynomial number of challenges to the device to collect the corresponding responses. Subsequently, the attacker tries to derive a numerical model

from the CRPs data by using ML techniques, as described in Section 4.2.3. As opposed to Becker's study, we would like to emphasize that in our study, we assume that the verifier/reader is a trusted party/tool and is capable of resisting any related attacks concerning the PUF configuration, model, or database in the verifier/reader.

A man-in-the-middle or eavesdropping to collect CRPs during a communication between verifier and prover could be one of the threats as well. However, both eavesdropping as well as physical access to the PUF is part of the established, general attack model for PUFs. Only one study , [55], performed a physical access attack on actual hardware which contained a PUF. In addition, the type of PUF is not necessarily confidential as revealed by NXP Semiconductors on the use of an SRAM-PUF as a hardware security device, but the configuration of intrinsic key generation remains secret [14]. Therefore, in this study, we also assume that the use of an Arbiter-PUF is publicly known and only the exact configuration of the CRP generation is kept confidential.

4.3 Arbiter-PUF Properties

This section describes the functionality and output transition probability of the Arbiter-PUF. In the subsequent sections, both will be used for the ML analysis.

4.3.1 Functionality Description

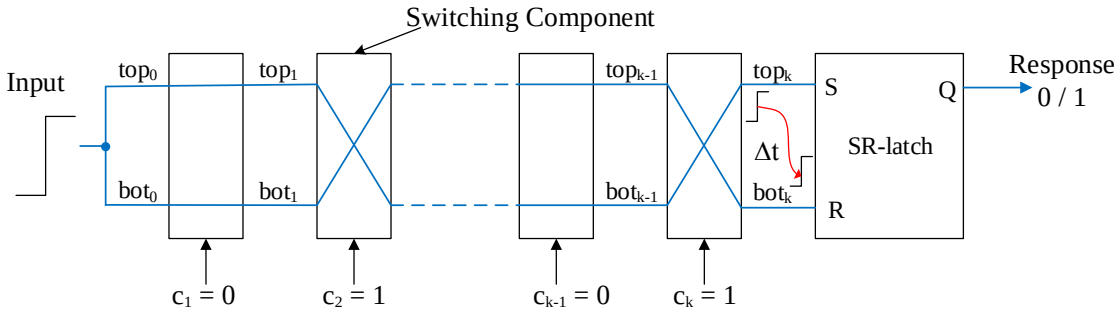


Figure 4.4: k-bit Arbiter-PUF.

The functionality of the Arbiter-PUF as depicted in Figure 4.4 can be described by an additive linear model, [36]. The total delays of both paths are modeled as the sum of the delays in each stage (switch components) depending on the challenge C=$\{c_1, c_2 ... c_k\}$. The final delay difference Δt between the two paths in a k-bit Arbiter-PUF can be expressed as:

$$\Delta t = \vec{w}^T \vec{\Phi} \tag{4.4}$$

where parameter $\vec{w}$ is the delay-determined vector and $\vec{\Phi}$ is the feature vector. Both parameters are the functions of the applied k-bit challenge with dimension $k+1$. As described in [8], $\delta_i^{1/0}$ is denoted as the delay in stage i for the crossed (1) and uncrossed (0), respectively. Hence, δ_i^1 is the delay of stage i when $c_i = 1$, while δ_i^0 is the delay of stage i when $c_i = 0$. Then

$$\vec{w} = (w^1, w^2, ...w^k, w^{k+1})^T \tag{4.5}$$

where $w^1 = \frac{\delta_1^0 - \delta_1^1}{2}$, $w^i = \frac{\delta_{i-1}^0 + \delta_{i-1}^1 + \delta_i^0 - \delta_i^1}{2}$ for all $i = 2, ..., k$, and $w^{k+1} = \frac{\delta_k^0 + \delta_k^1}{2}$. Furthermore,

$$\vec{\Phi}(C) = (\Phi^1(C), ..., \Phi^k(C), 1)^T \tag{4.6}$$

where $\vec{\Phi}^j(C) = \prod_{i=j}^{k}(1 - 2c_i)$ for $j = 1, ..., k$

From (4.5), the vector $\vec{w}$ encodes the delay in each stage of the Arbiter-PUF and via $\vec{w}^T\vec{\Phi} = 0$ determines the separating hyperplane in the space of all feature vectors, $\vec{\Phi}$. The delay difference, Δt, is the inner product of $\vec{w}$ and $\vec{\Phi}$. If $\Delta t > 0$, the response bit is '1', otherwise, the response bit is '0'. Determination of this hyperplane allows prediction of the PUF.

4.3.2 Output Transition Probability

According to Nguyen *et al.*, [95], a PUF with a k-bit challenge and a 1-bit response is said to satisfy the strict avalanche criteria (SAC) if its output transition occurs with a probability of 0.5 whenever a single challenge bit is complemented. The output transition probability for the Arbiter-PUF can be estimated as described in Algorithm 1. The CRP generation in steps c) and d) follows the setup as explained in Section 4.2.1. Subsequently, these CRPs are input to the MATLAB script (see Appendix A.5.1) to compute the output transition probability.

Figure 4.5 shows the computed output transition probabilities for 16-bit, 32-bit, and 64-bit Arbiter-PUFs. The challenge bit on the x-axis refers to the bit position (i.e., index) of the challenge which was flipped. As can be seen from Figure 4.5, all Arbiter-PUFs produce a generic pattern, where the output transition probability increases as the challenge bit index rises from 1 to k. However, the output transition probability for the 64-bit Arbiter-PUF is closer to a probability of 1 as compared to the 16-bit Arbiter-PUF for index=k. Following the discussion in Section 2.4.1, the effect of random process variations comes from the logic gates and interconnects. For the 16-bit Arbiter-PUF, as the rising pulse propagates through the delay stages, it experiences fewer random process variations resulting in significantly smaller delay differences, as compared to a 64-bit Arbiter-PUF. Due to this, the variations in the final delay stage are likely to compensate the flip in the delay difference of index k. Hence, the output of 16-bit Arbiter-PUF has a transition probability ≈ 0.54 when index=k, as can be seen in Figure

Algorithm 1 Computation of Output Transition Probability, $N = 1000$.

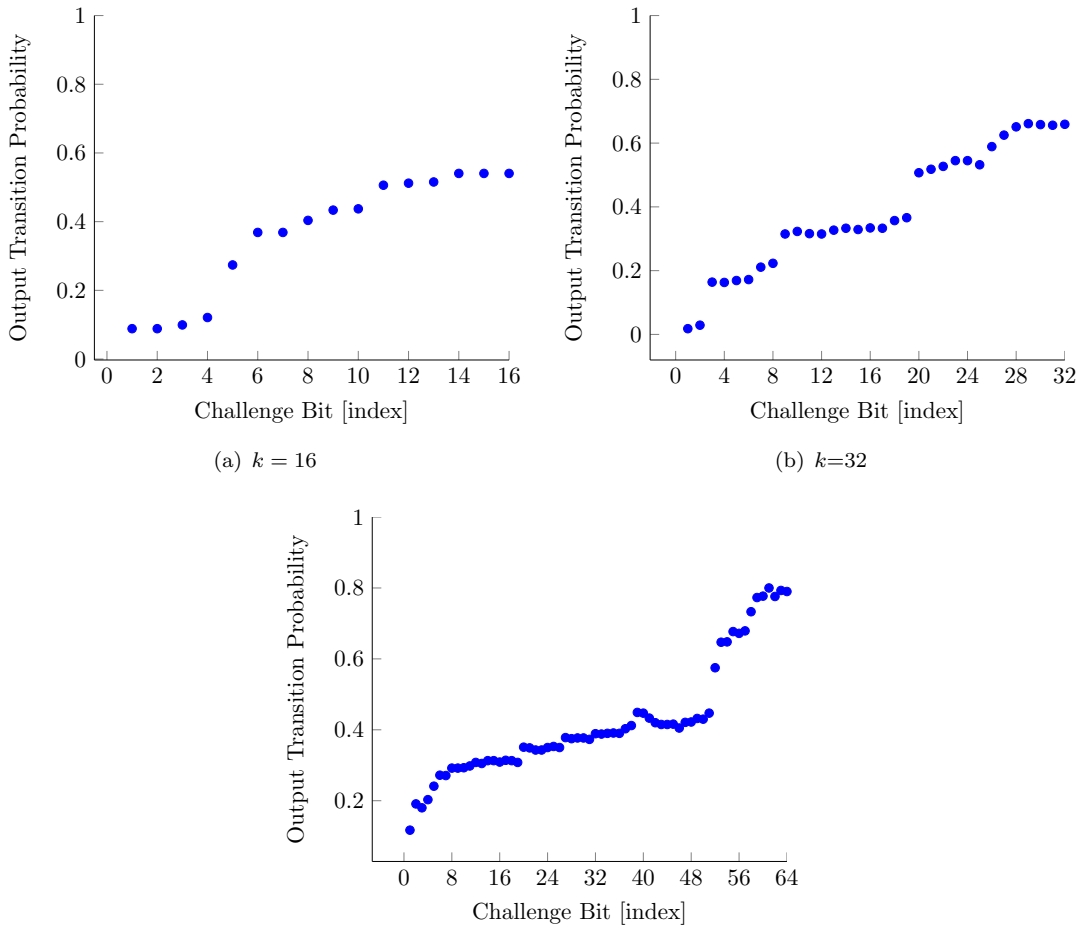
Input:

a) Generate N random challenges, $\mathbf{c}$.
b) Generate N challenges, $\hat{\mathbf{c}}$ from $\mathbf{c}$ with HD=1 for every challenge bit index.
c) Simulate k-bit Arbiter-PUF using a TSMC 65-nm technology node.
d) Collect CRPs for $\mathbf{r} \leftarrow PUF(\mathbf{c})$ and $\hat{\mathbf{r}} \leftarrow PUF(\hat{\mathbf{c}})$

Output:

Value of output transition probability

```
count ← 0
for index= 1 to k do
    for i = 1 to N do
        if r ≠ r̂ then
            count ← count + 1
        end if
    end for
    probability[index] ← count/N
    count ← 0
end for
```



(a) $k = 16$ (b) k=32

(c) k=64

Figure 4.5: Output transition probability for k-bit Arbiter-PUF.

4.5(a). Unlike the 64-bit Arbiter-PUF, the delay difference is more significant as the rising pulse propagates closer to the final delay stage and arbiter circuit. Hence, a flip in index=k will almost certainly flip the delay difference resulting in an output transition probability of ≈ 0.8, as shown in Figure 4.5(c). The probability values for the index=1 are observed to be significantly low, very close to zero. Based on this observation, given a set of CRPs, another set of CRPs can be generated by merely flipping the first-bit position of all the challenges where the corresponding responses remain the same. It can be concluded that an Arbiter-PUF does not fulfil SAC, resulting in CRP mapping that can be predicted easily using the ML technique, as we discuss in detail in the next section.

4.4 Analysis

In this section, the susceptibility of the Arbiter-PUF to an ML-attack is firstly discussed. Subsequently, the challenge permutation technique is introduced and the effect on the output transition probability of the Arbiter-PUF is discussed. Next, a random permutation required to further reduce the predictability of the Arbiter-PUF is proven.

4.4.1 ML-attack on Arbiter-PUF

As discussed in Section 4.2.4, we assume an attacker who has an access to primary inputs and is capable of doing a non-invasive CRP measurement. Besides, as pointed out in Section 4.2.2, the LFSR is used to generate an m-bit response from the Arbiter-PUF. Therefore, an LFSR could provide a non-direct access to the sub-challenges and an attacker can only map an input of the LFSR to an m-bit response of the Arbiter-PUF. As discussed in [96], the LFSR is vulnerable to a back propagation MLP technique and its internal sequence can be completely predicted. The seeds and their corresponding sequences (i.e., the first LFSR states generated from a given seed) have been used as a training set. As a result, the ANN was able to model the relationship between two consecutive LFSR states [96]. Following this finding, which was successfully attacked the standalone LFSR, it is interesting to see the ability of the ANN to learn our PUF configuration. To demonstrate the effectiveness of our MLP network (see Section 4.2.3) to attack our PUF configuration, a multi-label classification technique is used where multiple target labels (i.e., an m-bit response) can be assigned to each observation (i.e., a challenge to the LFSR). This is the simplest method of multi-label classification technique, called binary relevance, and therefore has been chosen. Other methods could be used such as label power set, random k-label set ensemble learning, etc., [97]. A multi-label classification technique adopted in our study is further defined as follows:

Given challenge $\mathcal{C} = \{0,1\}^k$, response $\mathcal{R} = \{0,1\}^m$ and a set of training examples $\mathcal{D} = \{(c^{(i)}, r^{(i)})\}_{i=1}^{N}=$,

$$\underbrace{\begin{bmatrix} c_1^{(1)} & c_2^{(1)} & \cdots & c_k^{(1)} \\ c_1^{(2)} & c_2^{(2)} & \cdots & c_k^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ c_1^{(N)} & c_2^{(N)} & \cdots & c_k^{(N)} \end{bmatrix} \begin{bmatrix} r_1^{(1)} & r_2^{(1)} & \cdots & r_m^{(1)} \\ r_1^{(2)} & r_2^{(2)} & \cdots & r_m^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ r_1^{(N)} & r_2^{(N)} & \cdots & r_m^{(N)} \end{bmatrix}}_{CRPs}$$

where
$c^{(i)} = [c_1, \cdots, c_k] \in \mathcal{C}$ is the representation of a challenge to LFSR,
$r^{(i)} = [r_1, \cdots, r_m] \in \mathcal{R}$ is the representation of m-bit response of an Arbiter-PUF for a given challenge, where

$$r_j = \begin{cases} 1, & \text{if } \Delta t > 0 \text{ to this challenge;} \\ 0, & \text{otherwise.} \end{cases}$$

As explained in Section 4.2.4, the use of the Arbiter-PUF is openly known and therefore, the attacker can describe the functionality of the Arbiter-PUF as in Section 4.3.1 and use Eq. (4.6) to increase the efficiency of an ML-attack. For each bit of the response, an ML-attack has been run with an incremental in the training set size (i.e., a step size of 1000 CRPs) and the corresponding prediction accuracy is recorded.

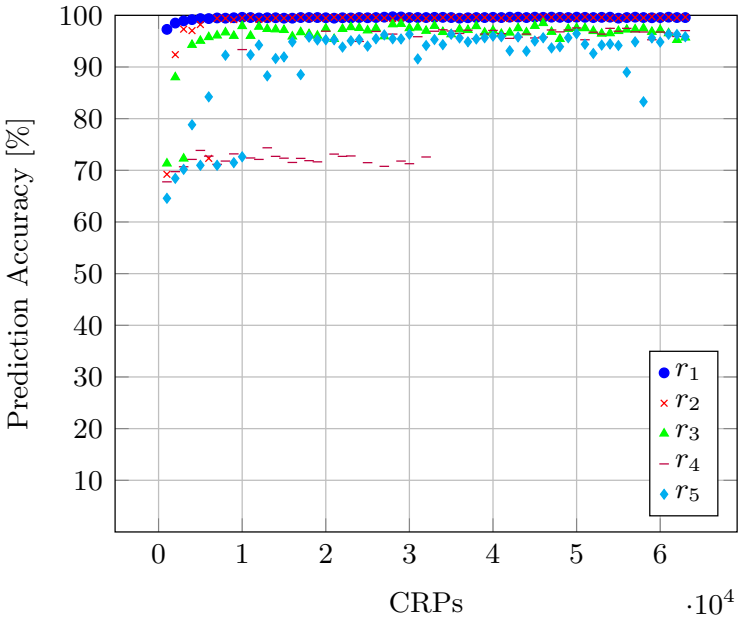


Figure 4.6: ML-attack on 16-bit LFSR plus 16-bit Arbiter-PUF configuration using ANN.

Figure 4.6 shows the prediction accuracy for a 16-bit Arbiter-PUF with $m = 5$. r_1 is the first bit of response which is generated from the first sequence of the LFSR and

it has the highest prediction accuracy of ≈ 99%. As can be seen from Figure 4.6, the prediction accuracy reduces for the next responses of the corresponding subsequent LFSR sequences and the ML starts to show some learning difficulties for a smaller training set size, particularly for r_4 and r_5. However, as the training size increases, the ML achieves better prediction accuracy of about 97.55% and 96.45% respectively, for r_4 and r_5.

All of the above findings show that a non-direct access of internal sub-challenges provided by an LFSR is not good enough to resist an ML-attack. In the next sections, we show that the resilience of the Arbiter-PUF against an ML-attack can be increased with a challenge permutation technique. From the analysis above, the most predictable response is the response corresponding to the first LFSR sequence. As the simulation run time for the 32-bit and 64-bit Arbiter-PUFs is huge, only the first sequence of the LFSR will be evaluated and analysed for subsequent use.

4.4.2 Challenge Permutation Technique

In this section, the challenge permutation technique is introduced and applied on Arbiter-PUFs. For a given size of an Arbiter-PUF, the challenge permutation space is huge. As a starting point, we analyse the permutation as shown in Figure 4.7 and investigate the impact on the output transition probability. k is the bit-length of the challenge and n is the length of the bit challenge that forms an n-bit block. The value of n is a power of two. This permutation scheme is known as an n-block permutation.

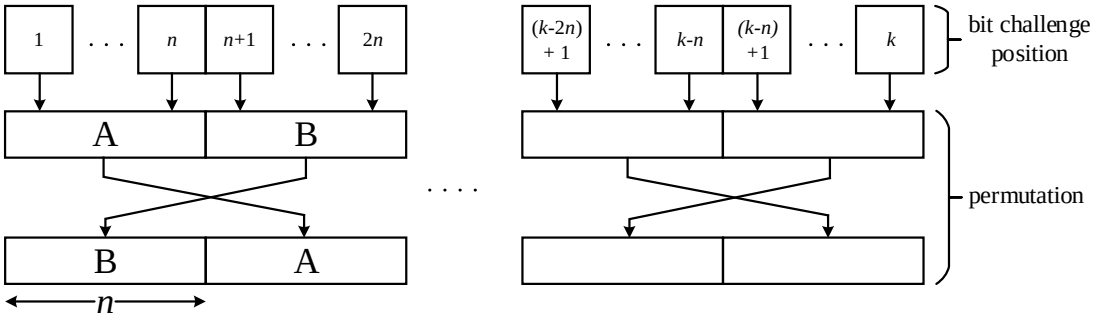
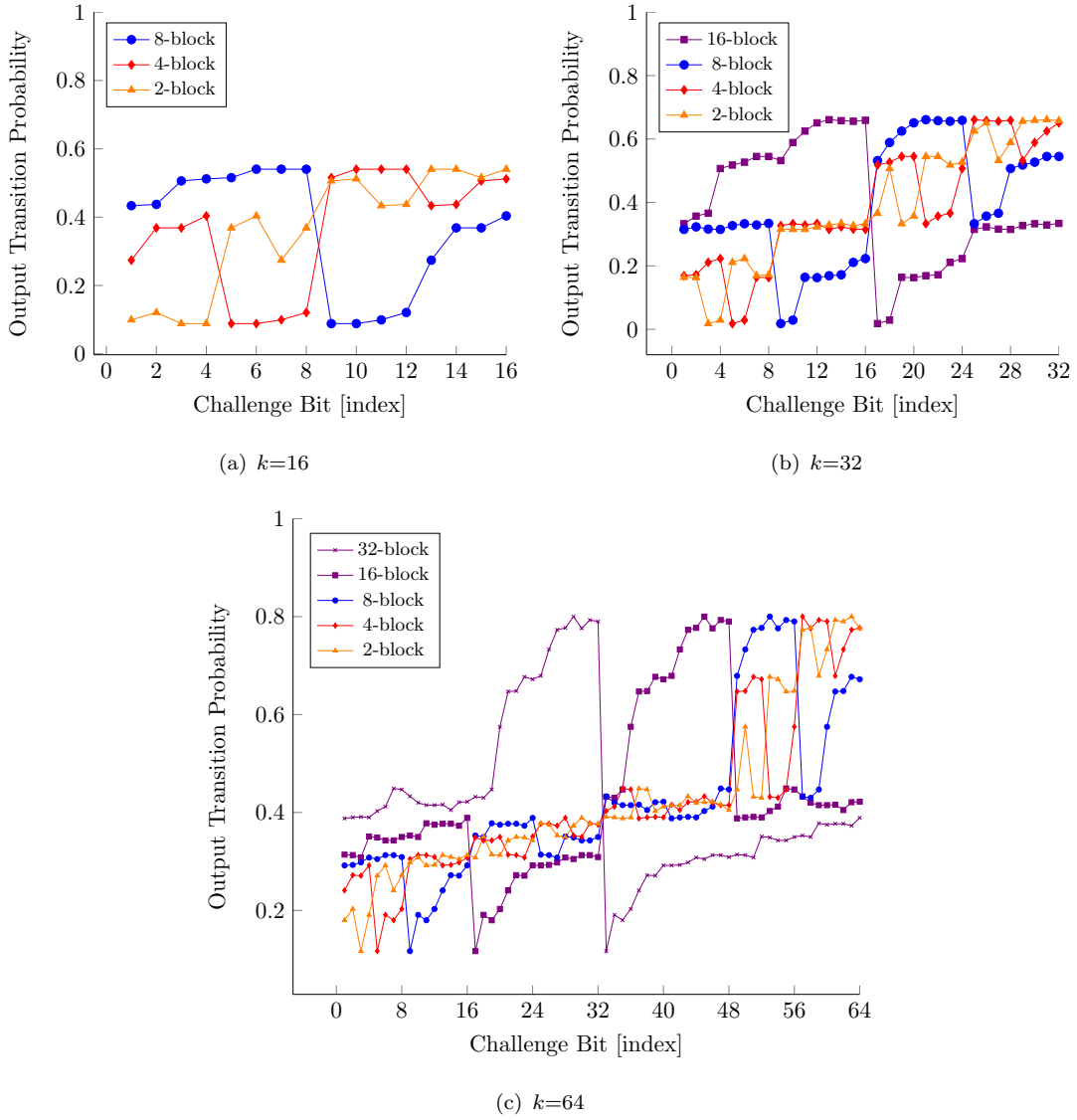


Figure 4.7: n-block permutation scheme.

The n-block permutation is applied to the generated CRPs used to compute the output transition probability in Figure 4.5. Next, the output transition probability based on n-block permutation is computed using Algorithm 1 and the values are illustrated in Figure 4.8[1] for 16-bit, 32-bit, and 64-bit Arbiter-PUFs. As can be seen from Figure 4.8, the probability for n-block where $n = \frac{k}{2}$, is the probability in Figure 4.5 in which

[1]Challenge bit index is a discrete value. Figure 4.8 has been plotted with continuous lines, however for better visibility.

the corresponding probability values of index 1 to $\frac{k}{2}$ and index $(\frac{k}{2}+1)$ to k have been swapped. Similarly, it applies for other n values.



(a) k=16

(b) k=32

(c) k=64

Figure 4.8: Output transition probability for k-bit Arbiter-PUF with n-block permutation.

Next, the n-block permutation is applied to randomly generated challenges given in Section 4.2.1. Subsequently, the permutated challenges are applied to an Arbiter-PUF to generate the corresponding responses. To ensure a fair comparison, only 32000 CRPs are evaluated for the ML-attack analysis. The ML-attack is performed on 32000 CRPs and the prediction accuracy is listed in Table 4.3. For each Arbiter-PUF configuration, as n reduces, the predictability of the response reduces. For small n, 2-block for instance, as the Arbiter-PUF size grows, the prediction accuracy is further reduced. Referring to Figure 4.8, the output transition probability tends to fluctuate as n reduces and as a consequence, the original probability pattern (see Figure 4.5) is not being preserved.

Indeed, this is correlated with the findings in Table 4.3 in which the fluctuation in the output transition probability indicates that the CRP mapping has been disordered and low prediction accuracy could be achieved.

Table 4.3: Prediction accuracy of k-bit Arbiter-PUF with n-block permutation scheme

Permutation	k=16	k=32	k=64
	Prediction Accuracy [%]		
32-block	NA	NA	99.27
16-block	NA	99.35	91.58
8-block	99.46	95.23	87.79
4-block	96.38	91.77	85.31
2-block	94.77	91.58	83.25

4.4.3 Random Challenge Permutation

The challenge permutation can alter the output transition probability of Arbiter-PUFs as described in Section 4.4.2, resulting in an increase in the complexity of the CRP mapping and reduce the susceptibility to an ML-attack. However, with the n-block permutation scheme, the prediction accuracy can only be reduced to 83.25%, for the 64-bit Arbiter-PUF. In this section, the impact of the random permutation is explored, with the aim to maximise the fluctuation in the output transition probability of an Arbiter-PUF and further reduce its predictability.

To start with, we design a procedure to generate a random permutation mapping. The output transition probabilities of Arbiter-PUFs in Figure 4.5 have been disordered through n-block permutation scheme as depicted in Figure 4.8. Nevertheless, some of the challenge bits still preserve the original probability pattern whereby the probabilities of these challenge bits are linearly increased and notably adjacent to each other. To break this pattern and to maximise the fluctuation in the output transition probability, we define two sets of conditions to check the probabilities of three consecutive challenge bits, as follows:

$$\text{Condition 1:} \quad \mid Pr(\text{index-1}) - Pr(\text{index}) \mid \leq \Delta\text{max}_k$$

$$\text{Condition 2:} \quad \mid Pr(\text{index}) - Pr(\text{index+1}) \mid \leq \Delta\text{max}_k$$

where Pr is the output transition probability. The above conditions are not applicable for the very first and last bit challenge positions. A parameter called Δmax_k is introduced to constraint both conditions which can be calculated as follows:

$$\mu_k = \sum_{\text{index}=1}^{k-1} \frac{|Pr(\text{index}+1) - Pr(\text{index})|}{k-1} \tag{4.7}$$

$$\Delta\max_k = \mu_k + \sigma_k \tag{4.8}$$

where μ_k is the average difference between two consecutive output transition probabilities and σ_k is the standard deviation. Both conditions must be avoided to ensure the fluctuation in the output transition probability is maximised.

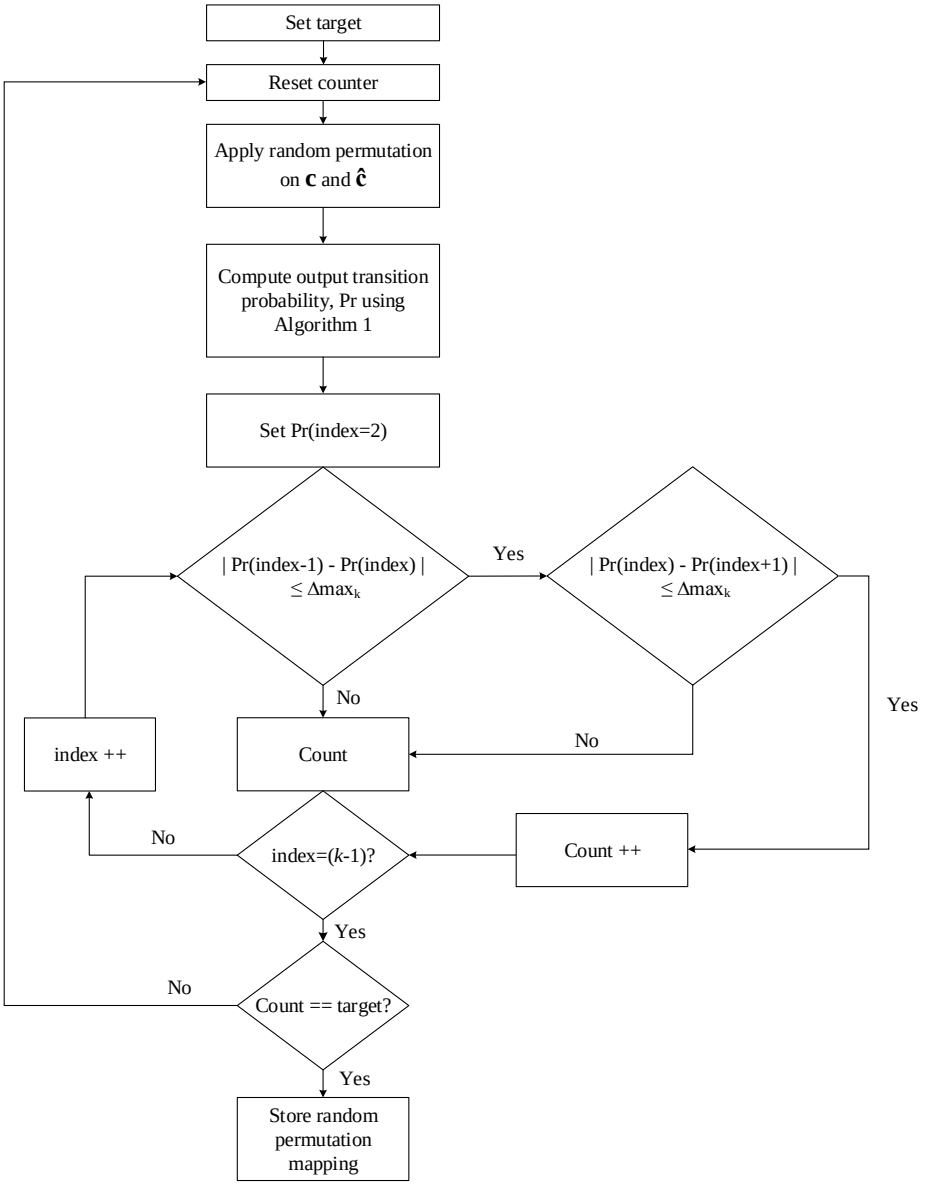


Figure 4.9: Iteratively finding a random challenge permutation mapping.

Figure 4.9 shows the iterative process for finding a good random challenge permutation mapping (see Appendix A.5.2 for the MATLAB script), which maximises the fluctuations in the output transition probability, based on the conditions explained above. The target in Figure 4.9 indicates the number of occurrences of both conditions in the output transition probability. Given an Arbiter-PUF with a k-bit challenge length, the maximum occurrence of both conditions is $k-2$. To demonstrate the correlation between the occurrence of both conditions and the ML prediction, the target is varied within the range of $0 \leq \text{target} \leq k-2$. Once the required challenge permutation mapping is found, it is applied to 32000 randomly generated challenges and their corresponding responses are generated following the methodology described in Section 4.2.1. Subsequently, the ML-attack is performed on 32000 CRPs for 16-bit, 32-bit, and 64-bit Arbiter-PUF. Figure 4.10 illustrates the correlation between the ML prediction and the occurrence of both conditions in the output transition probability for the aforementioned Arbiter-PUFs. As can be seen from Figure 4.10, as the occurrence of condition 1 and 2 reduces, the prediction accuracy reduces. The lowest prediction accuracy achieves for 16-bit, 32-bit, and 64-bit are 86%, 65.13%, and 69.04%, respectively.

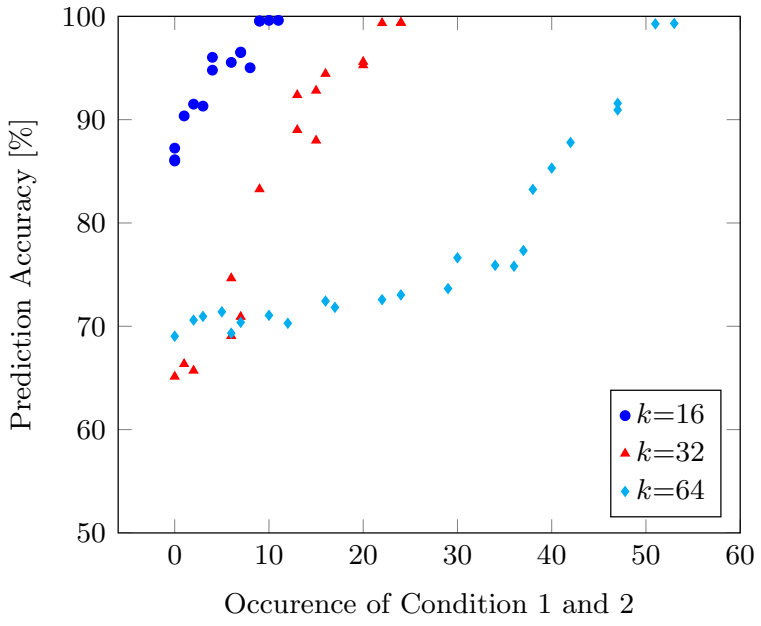


Figure 4.10: Correlation between the ML prediction and the occurrence of condition 1 and 2 for k-bit Arbiter-PUF.

Generally, for an explicit frequency of occurrence (e.g., 10), it can be observed that as k increases, the prediction accuracy reduces. Following this observation, considering the worst case condition where given a $k-$bit challenge length with a total number of 1's is $i = 1$ (i.e., index=1) and $(k-i)$ 0's, a total unique permutation can be computed as $\binom{k}{i} = {}^{k}C_i$. As mentioned in Section 4.3.2, without the challenge permutation technique, by flipping the first index or bit challenge position, another set of CRPs can be derived

easily because the output transition probability is almost zero. However, with a challenge permutation technique, the output transition probability is now $\binom{k}{i} = {}^{k}C_i$ possible values. As k increases, the possible values of output transition probability increases accordingly and increase the resilience of the Arbiter-PUF against an ML-attack.

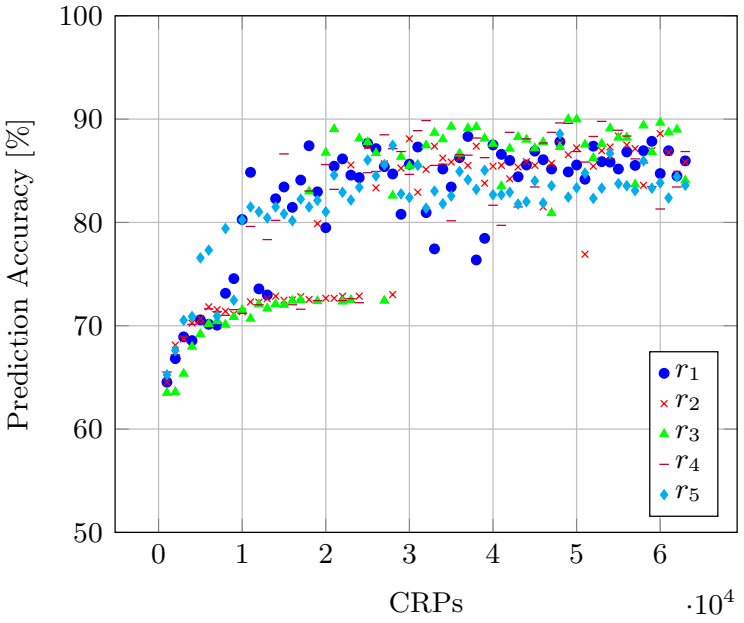


Figure 4.11: ML-attack on 16-bit LFSR plus 16-bit Arbiter-PUF configuration with permutated challenge using ANN.

For a fair comparison of the ML-attack before and after the challenge permutation is applied, a challenge permutation mapping that produces the lowest prediction accuracy for the 16-bit Arbiter-PUF in Figure 4.10 is applied to the CRPs that were used to generate the results in Figure 4.6. With a challenge permutation technique, the prediction accuracy of the 16-bit Arbiter-PUF in Figure 4.6 is reduced on average to within 80% to 90% as illustrated in Figure 4.11.

4.4.4 Hardware Implementation

In this section, we present a hardware implementation of the PUF configuration as described in Section 4.2.2. Figure 4.12 shows the top level architecture of a k-bit Arbiter-PUF to generate an m-bit response, which consists of a controller, k-bit LFSR, k-bit Arbiter-PUF, and m-bit serial-in and parallel-out (SIPO) register. From Figure 4.12, there is no functional unit block to implement the challenge permutation technique. This technique can be implemented by routing obfuscation between the challenge interface and internal ports of LFSR unit. Hence, a challenge permutation introduces no additional logic gates. However, the random challenge permutation mapping shown in Figure 4.9 must be conducted as part of the design.

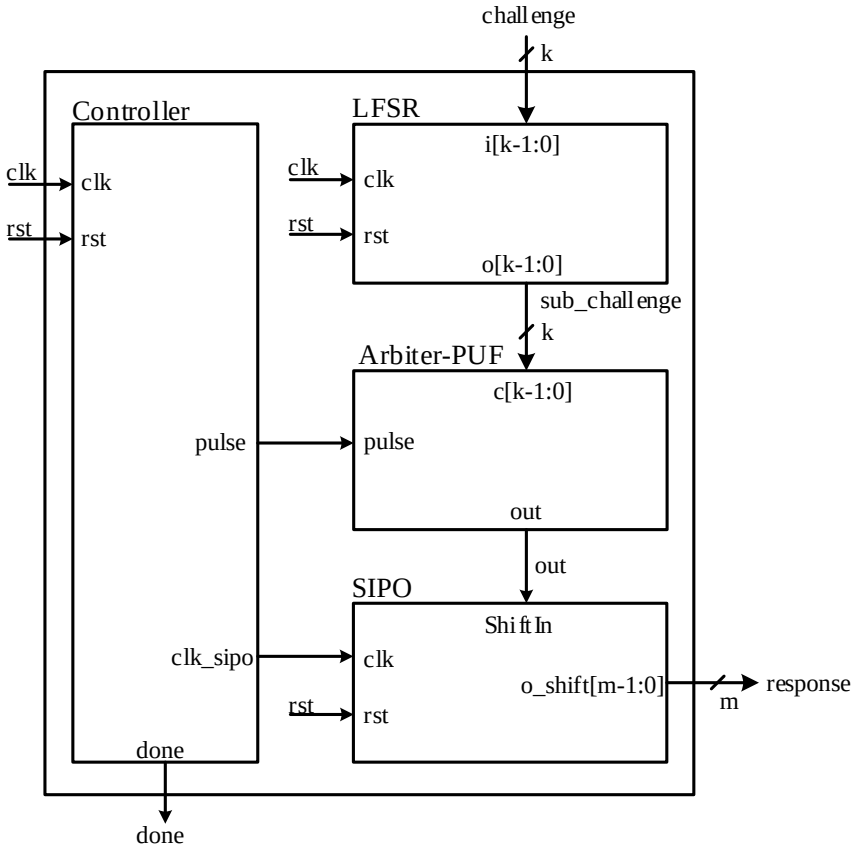


Figure 4.12: Top level architecture of k-bit Arbiter-PUF.

For an LFSR, it is desirable to be able to vary the challenge (seed) value since a different challenge will be sent to the prover every time an authentication is initiated (see Figure 2.17). To achieve this, a 2-to-1 multiplexer is placed at the input of each register in LFSR. As an example, Figure 4.13 illustrates the configuration for 4-bit LFSR which is also applicable for 16-bit, 32-bit, and 64-bit LFSRs used in our study. Based on this configuration, when rst=1, the sel=0 and the o[3:0] is reset to zero at the positive edge clock. Simultaneously, the challenge that asserted at the i[3:0] is loaded to the input of registers through the multiplexers, and the $o\_temp[3] \oplus o\_temp[0]$ is assigned to the $feedback$ signal. Subsequently, when rst=0, the next sequence of LFSR state or sub-challenge is generated at every clock cycle.

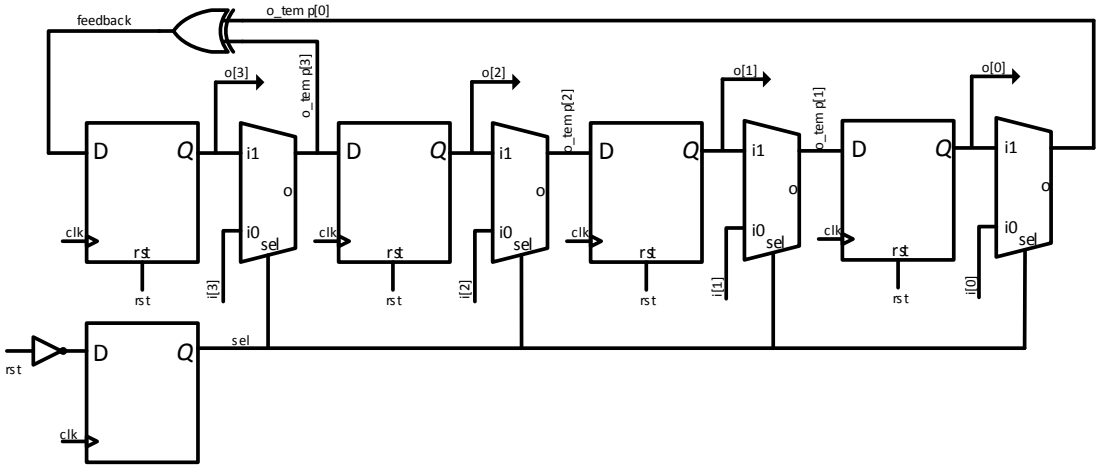


Figure 4.13: Example: 4-bit Fibonacci LFSR with 2-to-1 multiplexer.

All the functional unit blocks in Figure 4.12 have been implemented using Verilog (see Appendix B.1). To estimate the area and power, the Verilog design is compiled using Synopsys Design Compiler. Table 4.4 listed the area and power for different k and m values which have been estimated based on the configuration in Figure 4.12. As discussed in Section 4.4.3, the 32-bit and 64-bit Arbiter-PUFs exhibit better resilience against an ML-attack as compared to the 16-bit Arbiter-PUF. Hence, the 16-bit Arbiter-PUF is excluded from Table 4.4.

Table 4.4: Area and power estimation

k-bit Arbiter-PUF	Area [GE]		Power [mW]	
	m=64	m=128	m=64	m=128
k=32	807	1130	0.2018	0.2492
k=64	1193	1512	0.3368	0.3949

4.4.5 Predictability Comparison

Our findings in Figure 4.10 are summarised in Table 4.5 and compared to the previously reported data. With a permutation challenge technique, the resistance of the Arbiter-PUF to an ML-attack increases, and is better than the XOR Arbiter-PUF, the Lightweight-PUF and the VTC-PUF. On the other hand, with this technique, the susceptibility of the Arbiter-PUF to an ML-attack is approximately the same as the Current Mirror-PUF and the Lightweight OB-PUF. As mentioned in Section 4.1, a Controlled PUF which uses a one-way hash function makes the input-output relations of a PUF

unpredictable and with the attacker model, as given in Section 4.2.4, a Controlled PUF disables the ML-attack [28]. For area and power comparisons, a Controlled PUF configuration, [27], is considered which uses two hash functions at the pre-processing and post-processing stages. Without considering the ECC, and using SHA-1 as a hash function, a Controlled PUF already used 19134 gate equivalents (GEs) in area and 2.512 mW of power. Clearly, the challenge permutation technique has much less area and power - at least 12 times and 6 times smaller (for k=64 and m=128), respectively, as compared to the Controlled PUF. The area and power comparison for other PUFs in Table 4.5 were not reported. Nevertheless, since the implementation of a challenge permutation technique introduces no additional logic gates, it is fair to say that the Arbiter-PUF combined with a challenge permutation technique has a relatively low area and power compared to the other Arbiter-based PUFs in Table 4.5. In another perspective, without a challenge permutation technique, increasing further the size (i.e., $k > 64$) of an Arbiter-PUF does not increase the unpredictability of its response. A study, [73] shows that the 128-bit Arbiter-PUF can be predicted with 99.9% prediction accuracy at about 39000 CRPs.

Table 4.5: Comparison of prediction accuracy

Type	CRPs	Prediction Accuracy [%]
32-bit Arbiter-PUF + proposed permutation	3×10^4	65.13
64-bit Arbiter-PUF + proposed permutation	3×10^4	69.04
4-XOR 64-bit Arbiter-PUF [28]	12×10^4	99
4-XOR 64-bit Lightweight-PUF [28]	12×10^4	99
80-bit Current-Mirror PUF [35]	3×10^4	≈68
Lightweight OB-PUF [8]	2×10^4	63.27
64-bit RPUF [91]	2×10^2	69.1
64-bit VTC-PUF [34]	3×10^4	≈75
Controlled PUF [27]	NA	unpredictable [28]

4.5 Summary

This chapter proposed a challenge permutation technique to improve the resilience of Strong PUFs against an ML-attack, particularly on Arbiter-PUFs. The ML-attack resistance of a particular PUF is reflected by the complexity of the challenge to response mapping. In order to understand the impact of challenge permutations on the CRP mapping of the Arbiter-PUF, the output transition probability when a flip occurs in the challenge bits is computed and used for analysing the vulnerability of the Arbiter-PUF against an ML-attack. The output transition probability of the Arbiter-PUF shows that

it is indeed predictable as the probability increases with the challenge bit position (i.e., the bit that gets flipped) increases from 1 to k. Furthermore, a flip in the first challenge bit position produces an output transition probability that is very close to zero, which further increases the vulnerability of an Arbiter-PUF even to a simple guessing attack.

The proposed technique has been evaluated on 16-bit, 32-bit, and 64-bit Arbiter-PUFs. Those PUFs have been implemented using a TSMC 65-nm technology. The evaluation results show that a challenge permutation technique alters the output transition probability of the Arbiter-PUF and disorders its CRPs mapping. As a consequence, the resilience of the Arbiter-PUF against an ML-attack is increased. Moreover, a certain level of unpredictability can be obtained by controlling the behaviour of the output transition probability through a random challenge permutation. Hence, this approach can be used to find a good and random challenge permutation which produces the highest unpredictability, given a certain configuration of the Arbiter-PUF. A challenge permutation technique requires no additional hardware. Instead, it can be implemented by routing obfuscation between the challenge interface and the internal ports of the LFSR unit.

Chapter 5

A Reliable PUF in a Dual Function SRAM

Chapter 3 and 4 focus on the reliability and security issues of Strong PUFs used for low-cost identification and authentication. In this chapter, the reliability issues of Weak PUFs used for cryptographic key generation is explored. Though a wide range of Weak PUF techniques exists, the most investigated solutions are based on the use of random SUVs of SRAM-PUFs to generate cryptographic keys [23, 24, 98, 99]. Hence, an SRAM-PUF which has been proposed in [23], is used as a case study in this chapter. A dedicated SRAM circuitry to enable security introduces an area overhead and hence makes the cost unacceptable and can hinder the widespread adoption of PUFs. For cost efficiency, several papers [65, 98, 64, 100] suggested reusing the existing on-chip SRAM as a PUF to reduce the area overhead, hence, boosting the potential of an SRAM-PUF for lightweight secure key generation and making it particularly suitable for resource-constrained security devices. However, a dual function SRAM used as memory and PUF has reliability limitations. When the SRAM is used as memory to store an application's instructions and data, the prolonged storage of the same bit patterns can cause asymmetric degradation or stress of bit cells because of NBTI. For SRAM-PUFs, this means that the SUVs may become unreliable [24, 101]. In order to use an SRAM-PUF for cryptographic key generation, it must be able to generate error-free keys by using ECC as discussed in Section 2.7.2. However, the area of the ECC increases as the errors in the SUVs increase which is a disadvantage for SRAM-PUFs used in resource-constrained security devices. Preprocessing the PUF's response is one technique to reduce the area of the ECC [102]. Such technique has been exploited by Maiti *et al.*, [48] in which a pair of ROs is selected from a group of ROs such that they have the maximum difference in frequency, in order to increase the robustness of the RO-PUF against ageing. Motivated by the reliability issues of SRAM in dual use as memory and PUF, and the cost of preprocessing techniques, we have analysed the effect of asymmetric stress on the SUVs in both the instruction and data caches. We used a 32-bit ARMv8

architecture as a case study. We show that we can predict the signal probability pattern in a 32-bit ARM instruction cache (i-cache) and hence determine the appropriate bits to use for reliable SUVs. Therefore, we suggest using the i-cache as both memory and a PUF to save silicon area, and we propose bit selection as a preprocessing technique to increase the reliability of the PUF. The main contributions of this chapter are:

1. We show that the effect of NBTI on the SUVs of the bits of an SRAM-PUF is not uniform, but is dependent on the distribution patterns of the data stored in the memory. The analysis of a 32-bit ARM i-cache shows that the distribution pattern is predictable, as in the NBTI effect on the SUVs of SRAM-PUF.

2. We propose a bit selection technique to select only bit cells that have close to a 50% probability of storing a value of '1'. We show that this technique can significantly reduce the NBTI effect on SRAM-PUFs and yet still preserve the PUF performance.

This chapter is organised as follows: Section 5.1 describes the motivation to explore a preprocessing technique to increase the reliability of SRAM-PUFs in a dual functional mode. An overview of preprocessing approaches is discussed in Section 5.2. Section 5.3 describes the signal probability patterns found in a 32-bit ARM i-cache and data cache (d-cache). Section 5.4 investigates the data dependence of NBTI on SUVs based on the signal probability patterns in i-cache. The bit selection technique is discussed in detail in Section 5.5, including the PUF quality metrics. Section 5.6 describes the area overhead of the bit selection technique and ECC. Finally, this chapter is concluded in Section 5.7.

5.1 Motivation

5.1.1 A Dual Function SRAM

As mentioned earlier in this chapter, dual use of SRAM as memory and a PUF has been suggested to reduce the area overhead and achieve cost efficiency for resource-constrained devices. Schaller *et al.*, [64] proposed reusing the available on-chip SRAM in an ARM-based system-on-a-chip (SoC) as a PUF to protect the platform's firmware. The proof of concept has been successfully implemented on Level 3 on-chip RAM in the OMAP 4 SoC platform which consists of two ARM Cortex-M3 processors. The implementation of SRAM-PUF requires a modification to the boot-loader to extract a device-specific cryptographic key to decrypt the firmware when the device is powered-up. Elsewhere, Kohnhäuser *et al.*, [65] implemented PUF-based software protection by reusing the available on-chip SRAM resources in an ARM-based low-end embedded system, the Stellaris EK-LM4F120XL micro-controller. A similar idea to protect the

firmware authenticity and integrity have been proposed, in which the Level 1 cache (L1-cache) is reused as a PUF [98]. While all the above focused on cryptographic key generation, Bacha *et al.*, [100] reused the Level 2 cache (L2-cache) in an Intel Itanium II processor as a PUF for the challenge-response authentication protocol. The CRPs which are used for server-client authentication process are generated by exploiting the errors that occur in the cache line when the nominal supply voltage of L2-cache is reduced to a few mV. All of the above studies show that a dual function SRAM as a memory and PUF is feasible and most of them have been successfully implemented on the available embedded systems and SoC platforms. However, the impact of NBTI on SUVs in a dual function SRAM has not been discussed.

5.1.2 Ageing Mitigation in SRAM-PUFs

As reviewed in Section 2.6.2, the reliability of SUVs due to NBTI has been mainly focused on SRAM as a PUF [47, 39]. In this section, an ageing mitigation technique for SRAM-PUF is described. Figure 5.1 shows the data dependence of NBTI on a 6-T SRAM cell, [101]. As a '1' is stored in the SRAM cell ($Q = 1$ and $QB = 0$), the transistor MP1 is subjected to NBTI stress resulting in a slow increase in the V_{th}. Under the prolonged storage of '1', the V_{th} of MP1 increases significantly, whereas the V_{th} of MP2 remains the same (i.e., asymmetric degradation or stress). As a consequence, after the power-up process, the cell is less likely to power-up to a '1' than it was before the NBTI effect. Hence, the SUV becomes unreliable. Nevertheless, the SUV becomes reliable when its opposite value is stored in the cell. Following this principle, Bhargava *et al.*, [101] propose direct accelerated ageing (DAA) to increase the reliability of an SRAM-PUF. An SRAM cell would be written with the opposite value of its SUV and reinforced through post-manufacturing burn-in stress (i.e., high temperature and high voltage conditions). However, such a long burn-in is prohibitively expensive and time-consuming in the manufacturing flow. Furthermore, in the field of application, if the SRAM is used as memory and PUF, it will be subject to regular silicon ageing and may deteriorate over time.

One study, [103], analysed 16 possible (anti-)ageing configurations. Of these, reinforcing the power-up state using the long-term ageing data (the inverse of the corrected enrolment state using ECC) was found to be the best solution. However, to use the inverse of the corrected enrolment state as anti-ageing strategy is not really practical as the SRAM is not solely dedicated to the PUF function. The spatial correlation of reliable bit cells under random environmental conditions (temperature and voltage) has also been studied, [104], and it was proposed to only select those fully-biased bit cells, eliminating the partially-skewed and neutrally-skewed cells. The proposed bit select algorithm is expected to increase the robustness of an SRAM-PUF against environmental variations and ageing. However, it is still susceptible to reliability issues due to NBTI

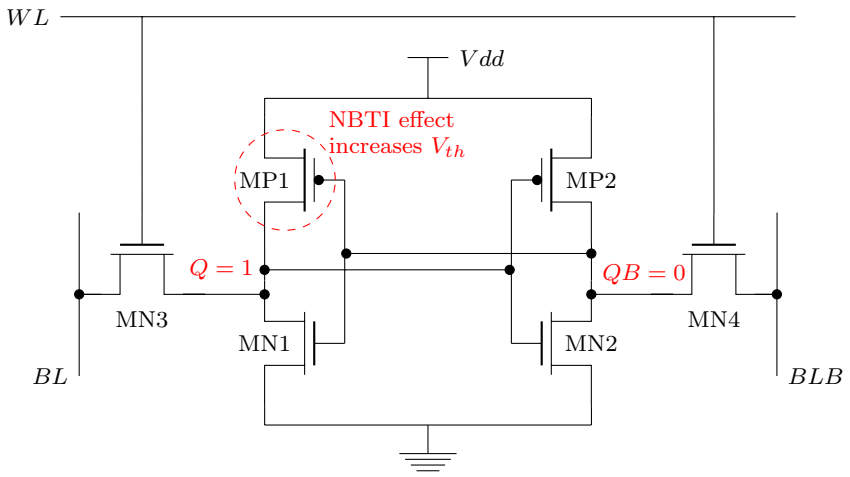


Figure 5.1: NBTI impact on a 6-T SRAM cell circuit.

when used in the field as memory and PUF. It is notable that all of the above studies are dedicated to the use of SRAM for the PUF only and not for regular memory.

5.1.3 Area Estimation of ECC

As discussed in Section 5.1.2, the SUVs may become unreliable as a result of NBTI. Therefore, an ECC is required to generate error-free cryptographic keys from the noisy SUVs of an SRAM-PUF (see Section 2.7.2). Bose-Chaudhuri-Hocquenghem (BCH) is an ECC scheme that has been widely used for correcting errors of PUFs [23, 46, 62, 102]. The BCH scheme is given as $[n, k, t]$ where n raw bits (codeword) are required for a k bit message and the scheme can correct up to t bits in error. The GE area of the BCH scheme has been calculated, [62], for $5 \leq m \leq 7$ and $1 \leq t \leq 7$, where the codeword $n = 2^m - 1$. Using this data, the lin-log plot of the area (y-axis) versus the codeword n (x-axis) can be derived and extrapolated to estimate the area for $8 \leq m \leq 11$. Figure 5.2 shows the calculated area (solid lines), [62], and the estimated area (dashed lines) by extrapolation.

For the BCH scheme, each codeword has a maximum capability to fix a certain number of errors (see Appendix C.2 for BCH code). For example, using a codeword $n = 127$, a BCH scheme can fix up to $t = 31$ error bits for a message $k = 8$. For $t > 31$, one has to choose the next available codeword that has the capability to fix more errors. Figure 5.2 only shows the area for $1 \leq t \leq 7$. To estimate the area for $t > 7$, the relationship between the area and the number of errors has to be found. Based on the data in Figure 5.2, the area increases linearly as the number of errors increases, which is also consistent with the reported results, [105]. Therefore, the area for $5 \leq m \leq 11$ can be estimated for $t > 7$ by linear extrapolation. Although an ECC is very useful to improve the PUF

reliability, it introduces significant extra overhead that increases with the number of potential errors.

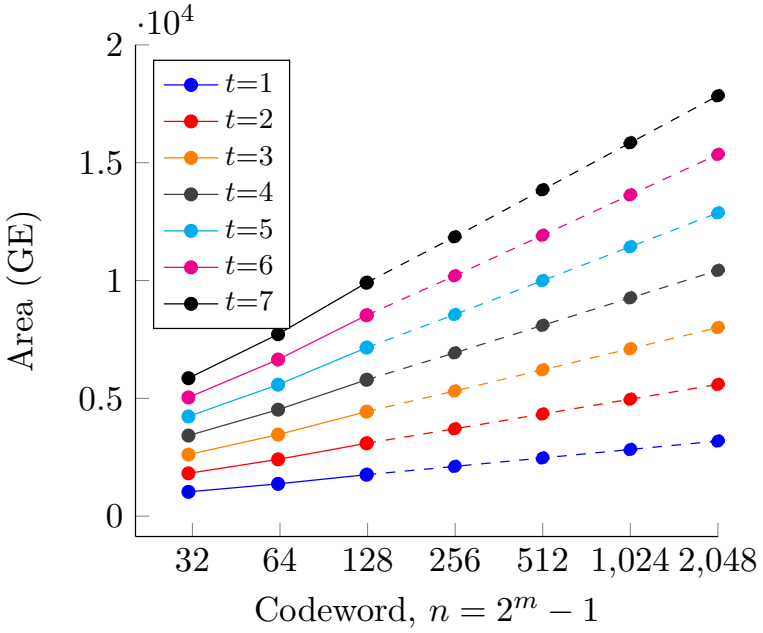


Figure 5.2: Area (GE) of the BCH scheme.

.

In this section, we have shown that a dual function SRAM for memory and PUF is feasible and implementable on embedded systems and SoCs, such as the Stellaris and the OMAP 4 platforms. However, the impact of ageing on SUVs of SRAM has been mainly studied considering only a PUF. The SUVs may suffer a significant reliability issue due to data dependence of ageing in a dual function SRAM. Moreover, the area overhead of ECC increases as the number of unreliable SUVs increases. The aforementioned issues motivate us to study the data dependence on ageing in a dual function SRAM and to propose a bit selection technique to increase the reliability of SUVs and significantly reduce the area overhead of an ECC.

5.2 Pre-processing Approaches

Reusing the on-chip SRAM in a system as a PUF, as discussed in Section 5.1.1, could be cost-efficient, especially for resource-constrained devices. However, dual use of SRAM as a memory and a PUF is not straightforward because NBTI causes asymmetric degradation of memory bit cells and reduces the SUV reliability over time. In our work, the on-chip memory, particularly, the i-cache in an ARMv8 architecture has been used as a case study. The NBTI analysis, based on the signal probability pattern of the i-cache, suggests that some of the bits experience nearly symmetric stress. Further, bit selection

is used as a preprocessing technique to increase the SUV reliability for PUF usage. The bit selection technique is most suitable for a system that runs a specific application such as an embedded system. Figure 5.3 shows an overview of this technique which is divided into three parts. These parts will be discussed in Section 5.3, Section 5.4 and Section 5.5, respectively.

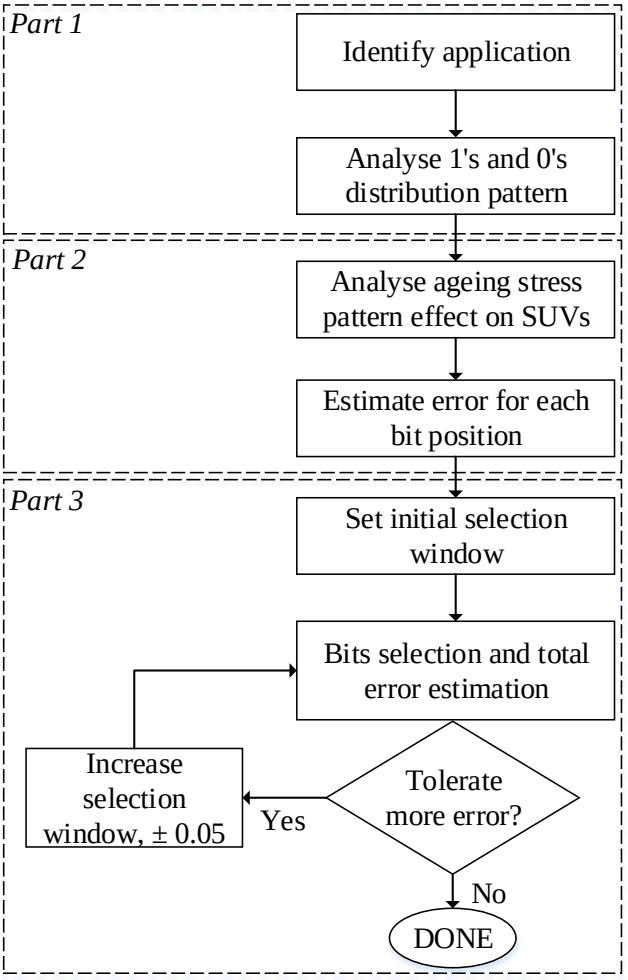


Figure 5.3: Overview of a bit selection technique.

5.3 Signal Probability Pattern in SRAM Caches

In this section, the distribution of a signal probability for a 32-bit ARM i-cache and d-cache are explored [1]. As discussed previously, [106, 107], the instruction set usage indicates only a small subset of instructions is executed by most applications. For an i-cache, it is expected that the overall bit probabilities will be distributed unevenly across all 32 bits. Figure 5.4 shows one of the 32-bit ARM instruction set formats for multiply and multiply-accumulate instructions. The condition field (bits 31:28) determines the

[1] Data in Figure 5.5 and 5.6 were collected by Shengyu Duan.

circumstances under which an instruction is to be executed. In the absence of a suffix, the condition field of most instructions is set to "Always" (suffix AL). Therefore, it is expected that bits 31:29 have a high probability of storing '1' and bit 28 has a low probability of storing '1'.

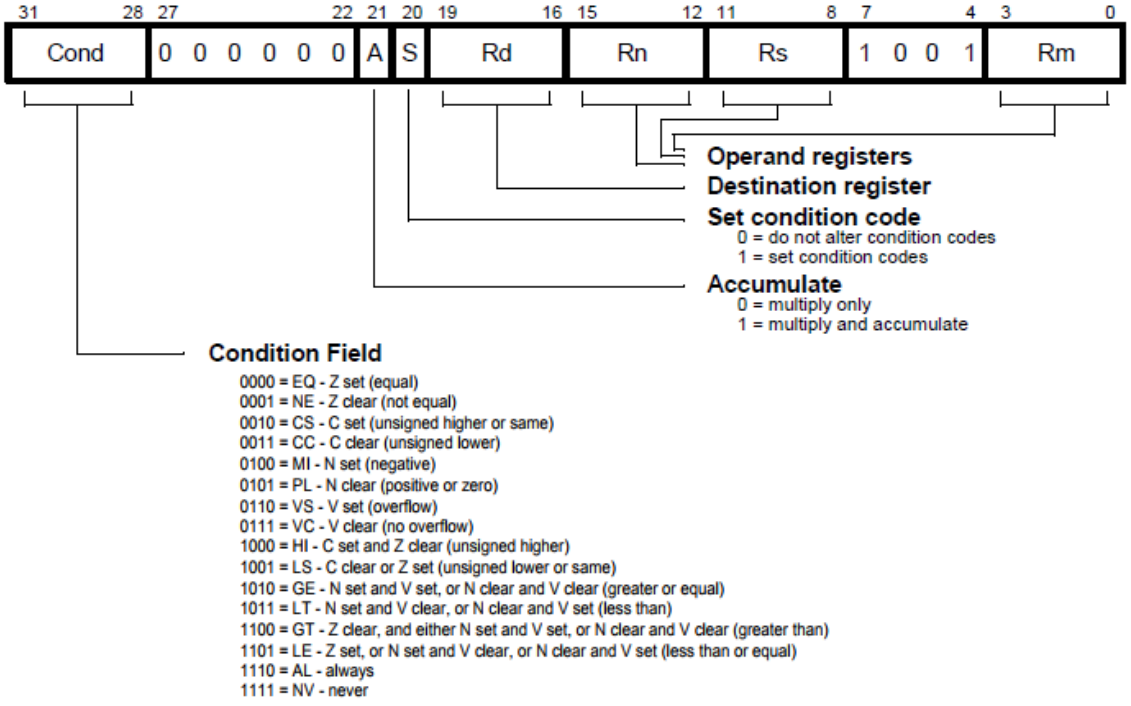
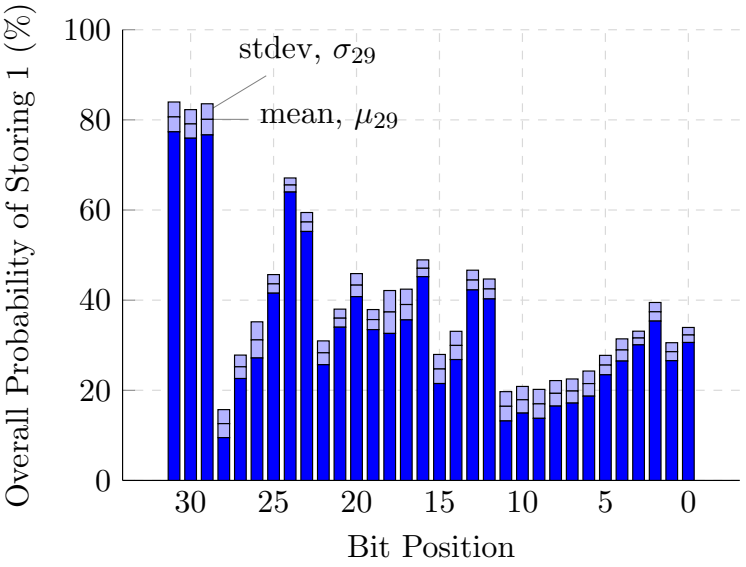


Figure 5.4: Multiply and multiply-accumulate instruction format for 32-bit ARM [108].

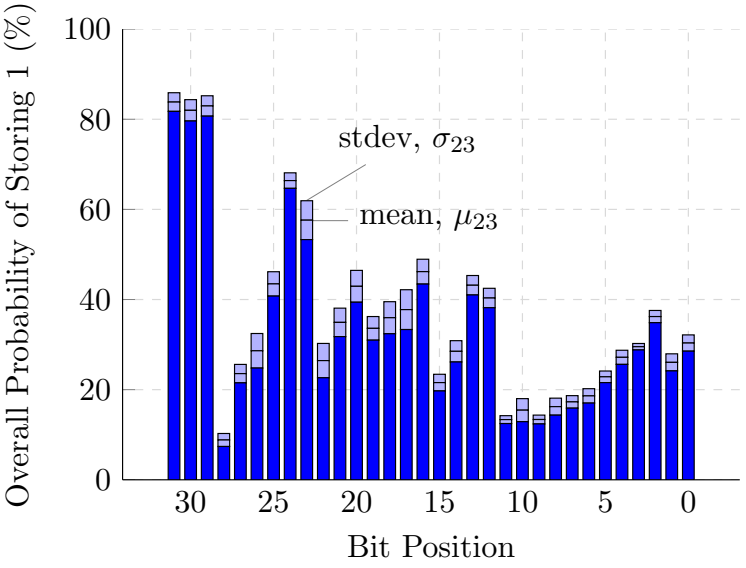
To investigate the signal probability pattern of an i-cache in ARMv8 architecture, a simulator, gem5, has been used, [109]. Table 5.1 gives the cache configurations. 16 benchmark programs were chosen from the MiBench [110] and llvm [111] benchmark suites, covering a number of different functions. In our study, the gcc compiler has been used and each cache access was dynamically traced during the program execution, such that the bit probabilities for each cache line were updated each time that it was replaced. The overall distribution was produced by averaging the final bit probabilities over the whole cache. The results are shown in Figure 5.5, for two extreme cases of cache policy namely the direct mapped cache and the fully associative cache, which give the least and the most balanced block usage, respectively. In each case, some bits preserve the same values in certain locations which also indicates that the probability of storing '1' for bits 31:28 is as expected. As can be seen, the patterns are generally independent of the programs and of associativity, and therefore the patterns are predictable.

Table 5.1: Cache configuration

Cache size	8kB
Block size	16-Byte
Number of block	512
Addressing scheme	Word (4-byte) addressing
Computer architecture	32-bit ARM



(a) Direct mapped cache



(b) Fully associative cache (replacement policy:most recently used)

Figure 5.5: Mean and standard deviation values for the probability of storing a '1' in i-cache over 16 benchmarks.

A similar experiment using the gem5 simulator has been conducted for the d-cache by running four benchmark programs. The overall probability for each bit is observed by tracing each cache access. Figure 5.6 depicts the probability patterns in a direct mapped d-cache for four benchmarks such as bubblesort, basicmath, queens and susan. Ideally, one would expect that d-cache stores random values, such that each bit has an overall 50% probability of storing '1'. Therefore, this would make d-caches more suitable as PUFs, compared to i-caches. However, as can be seen from Figure 5.6, the distribution is highly dependent on applications. Although the distributions are more balanced than in the i-cache, the probabilities of storing a '1' are less than 50% for all these benchmarks, which indicate asymmetric degradation in the d-cache as well. It is possible that other applications may produce different distributions (with bit probabilities equal to or higher than 50%), but neither the reliability of a d-cache nor its performance as a PUF can be easily predicted.

It is also worth to mention that previous studies, [112, 113] proposed a bit flipping technique to mitigate the NBTI impact on SRAM cells or caches. This technique could help to make the signal probabilities of the stored value in the SRAM cells close to a 50%, hence, leads to a symmetric NBTI stress. Kumar *et al.*, [112] proposed a periodic flipping of SRAM cells to the entire memory block which interrupts the normal operation (i.e., read or write operation) by thousands of clock cycles. To avoid the interruption of cache normal operation, Gebregiorgis *et al.*, [113] proposed a flipping of SRAM cells during write operations or "flip-on-access" scheme. Nevertheless, the "flip-on-access" scheme does not guarantee that the signal probabilities of SRAM cells close to a 50% as the stored data are not regularly flipped. Thus, a "flip-on-access" scheme is less effective to balance the NBTI stress and the reliability of a cache or PUF would still degrade.

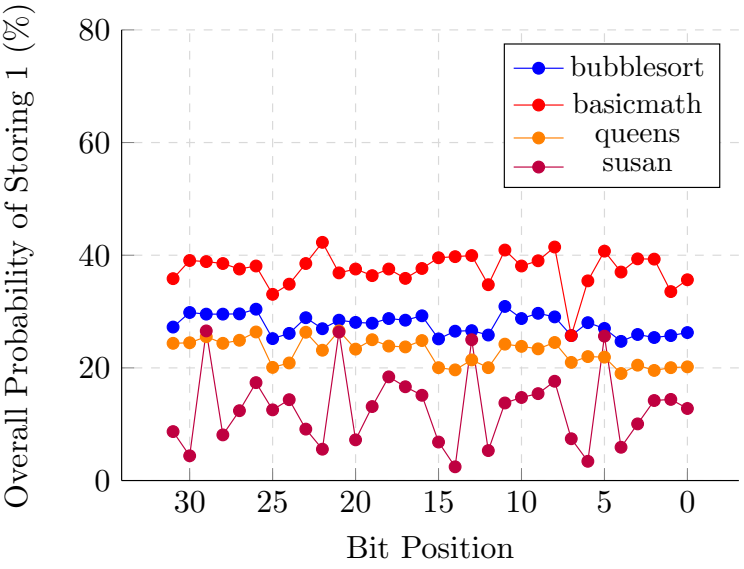


Figure 5.6: Mean values for probability of storing '1' in d-cache running four benchmarks.

From the analysis of the signal probability for a 32-bit ARM i-cache, Figure 5.5, some of the bits have close to a 50% probability of storing '1'. For example, bits 12, 13, 16, 20, 23, and 25. This indicates that those bits will have a symmetric stress, and thus are more reliable over time. On the other hand, bits 31:29 have a high probability of storing '1' while bit 28, as well as bits 11:9, have a high probability of storing '0'. As explained earlier, prolonged storage of the same bit value over time causes asymmetric stress, thus those bits are likely to have reliability issues.

5.4 NBTI Impact on SUVs

5.4.1 Simulation Setup

Without the effect of NBTI, the randomness of SUVs in SRAM cells is determined by the process-dependent variations in the cross-coupled inverters. To simulate the process variations, an SRAM i-cache has been modelled in a low-k TSMC 65-nm technology. To get a good distribution, a Monte Carlo simulation was performed for 100 SRAM instances. However, to run a Monte Carlo simulation of the complete circuit of an 8kB SRAM, as shown in Table 5.1, is time-consuming and therefore, for a proof of concept, only 32 bits x 64 rows were simulated and analysed. The simulation was performed with the BSIM4 (V4.5) transistor model, using the 3σ variation in the technology design kit. Unless otherwise stated, the ramp-up time of the supply voltage is fixed at 1ms throughout the simulation. To determine the ageing due to NBTI, the duty cycle for the 32-bit SRAM cells was defined according to the signal probabilities in Figure 5.5(a). As a proof of concept, only a direct-mapped i-cache configuration is considered for analysis as both i-cache configurations in Figure 5.5 have a quite similar signal probabilities. Then, the V_{th} degradation for every bit cell was simulated following the methodology as described in Section 3.6.1, for 5 years with 100% activity factor.

5.4.2 Bit Error Analysis Under NBTI Effect

Based on the Monte Carlo simulation of 100 SRAM instances, the randomness of SUVs in SRAM cells is shown in Figure 5.7. Prior to the NBTI effect as shown in Figure 5.7(a), SRAM cells have an even distribution of '1' and '0' SUVs for all bit cells and therefore, they are random. In other words, the HW for each bit is about 0.5. However, when they undergo NBTI ageing with the signal probabilities given in Figure 5.5(a), the distribution of '1's and '0's become uneven. Bits 31:29 have a high probability of storing '1's. As a result of the prolonged storage of the same bit value, they are likely to power-up to '0' after 5 years as illustrated in Figure 5.7(b). In the opposite scenario, bits 11:9 and bit 28 are likely to power-up to '1'. The bits that have close to a 50% probability of

storing '1', such as bits 12, 13, 16, 20, 23 and 25, still show an even distribution of '1's and '0's. Our ageing results in Figure 5.7(b) agree with the discussion in Section 5.1.2.

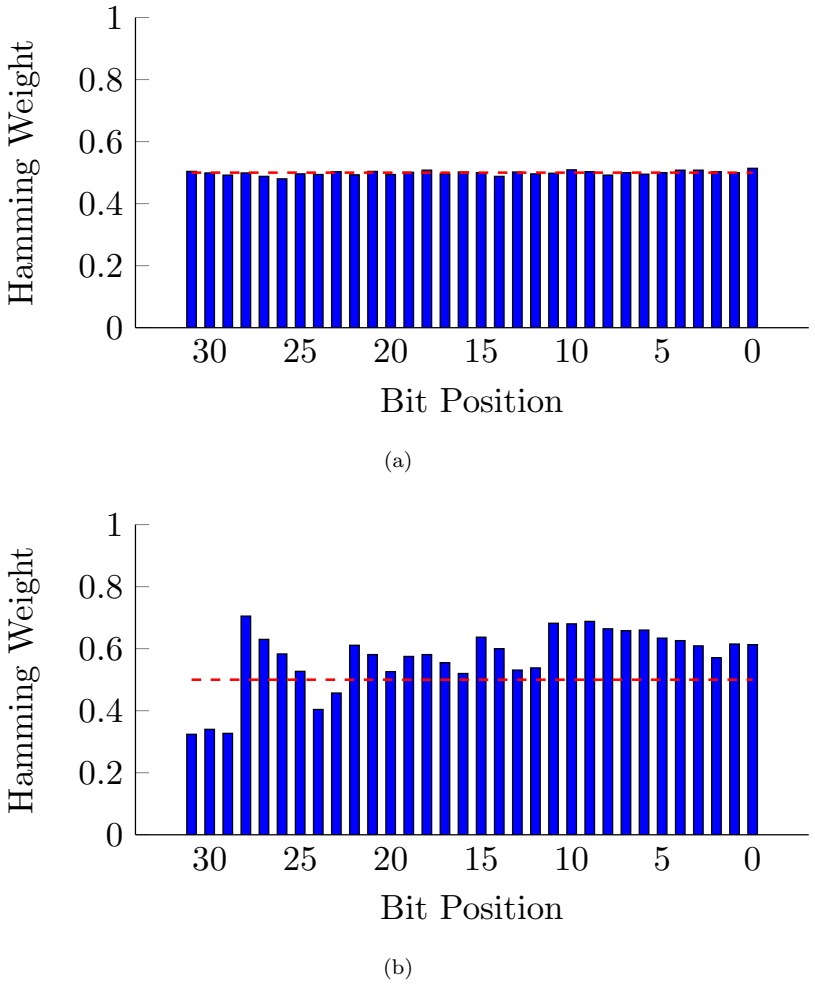


Figure 5.7: Distribution of '1' and '0' (a) fresh (b) 5 years ageing based on the mean probability of storing '1'.

Based on the distribution of SUVs at fabrication time and after 5 years, the bit error for every bit cell is calculated using the Intra-HD_{NBTI}, Eq. (2.5), as shown in Figure 5.8. As expected, because of the symmetric stress experienced by bits 12, 13, 16, 20, 23, and 25, the bit error count in those positions is much less than for the other bits. One might argue that NBTI can be used to increase the reliability of SRAM-PUFs by storing the inverse value of SUVs as in [101, 103]. However, that only works if the SRAM is used as a PUF only. Here, we use the SRAM as both an i-cache and a PUF. Because of the random SUVs and difficulties to control the data being stored in the i-cache, choosing the bit cells that experience a balanced stress can retain the intrinsic mismatch of the inverters in the cell. Nevertheless, these bit cells are exposed to environmental variations such as temperature and/or supply voltage variations as will be discussed in Section 5.5.1. At this point, we have discussed the first and second part of the preprocessing procedures

(Figure 5.3). The remaining procedure for the preprocessing technique will be discussed next.

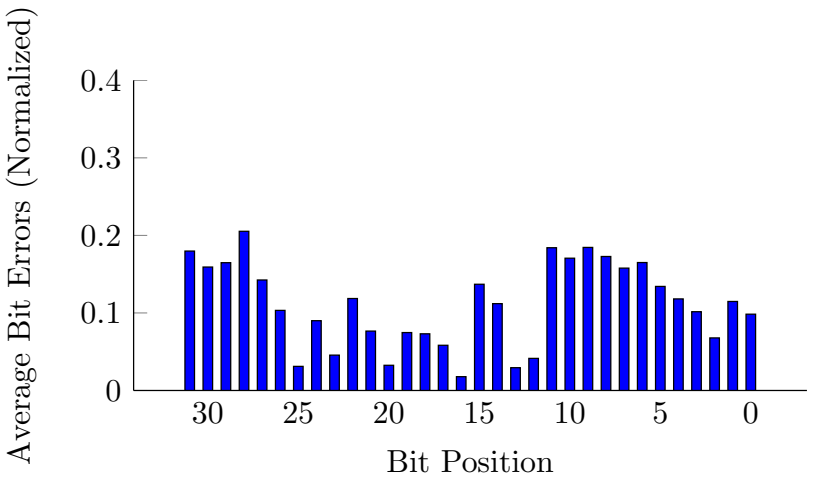


Figure 5.8: Average bit errors based on the mean probability of storing '1'.

5.5 Bit Selections

As discussed in Section 2.7.2, in order to extract a key from a PUF, we need a certain number of bits. Using all of an SRAM gives us the maximum number of bits to choose from, but as we have seen from Section 5.4.2, the SUV of many bits becomes unreliable over time. Therefore, the purpose of bit selection is to find as many reliable bits as possible within an SRAM.

There are two important observations from the previous analysis: (i) the SUV reliability of a cell depends on the evenness of the stress in the cross-coupled inverters; (ii) 6 of the bit cells in this example have the fewest bit errors as the system ages. These are bits 16, 13, 25, 20, 12, and 23 (ranked in ascending order of their bit error count). The idea of a bit selection technique is to only select bits for the PUF that have close to a 50% probability of storing '1' values. Ultimately, the selection of these bits reduces the overall bit error rates. In Section 5.4, the second part of the preprocessing procedure has been discussed in which the individual bit error rates have been estimated. To perform the next part of the preprocessing procedure, which is the bit selection, the relationship between the HW and the mean probability of storing '1' has to be found. By taking the HW from Figure 5.7(b) and the mean probability of storing '1' from Figure 5.5(a), the relationship between these two parameters for all 32 bits can be seen, Figure 5.9. The ideal case is when a bit cell has a 50% probability of storing '1', i.e., HW=0.5. In Figure 5.9, as the mean probability of storing '1' reduces, the HW increases and vice versa. In order to evaluate the increase/decrease in the bit error rate for a selection of bits as the HW reduces or increases, first, we chose the lower boundary (LB) and upper boundary

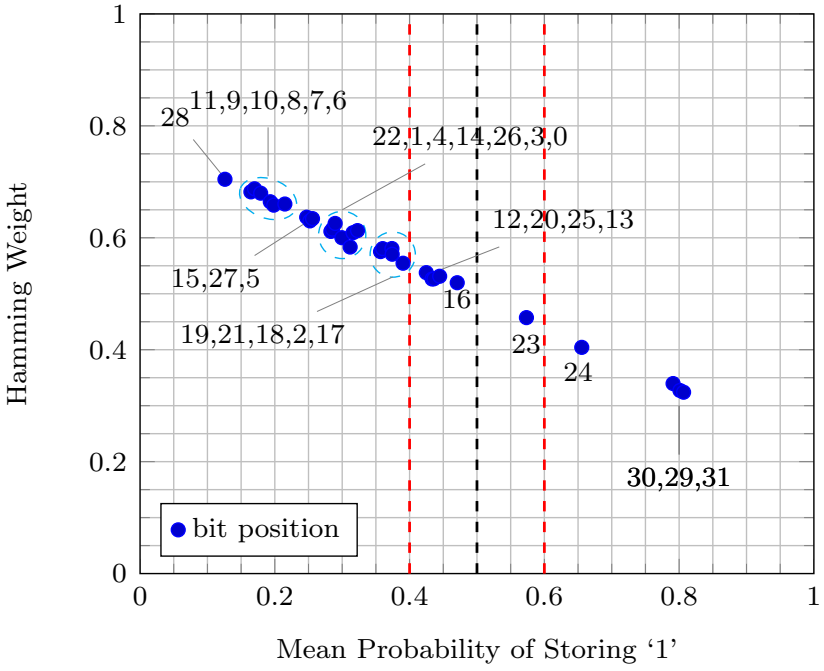


Figure 5.9: The relationship between the HW and the mean probability of storing '1' for all 32 bits.

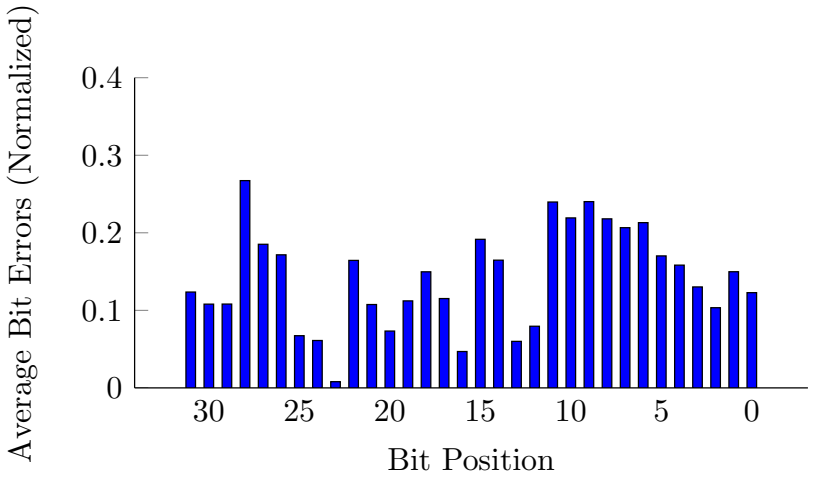
(UB) at 0.4 and 0.6 (i.e., initial selection window), respectively, as highlighted by the dashed lines in Figure 5.9.

Note that the six bits that have the fewest bit error rates after ageing are within these two boundaries. These selected bits are represented as S1 and their corresponding bit error rate is shown in Table 5.2. The bit error rate in Table 5.2 is calculated by taking the sums of the average individual bit errors in Figure 5.8 and dividing by the total number of bit cells. Next, the margin of the selection window is widened by ±0.05. The aim is to maximise the number of bits while keeping the error below the maximum error tolerance value. As can be seen from Table 5.2, as the margin of bit selection increases, the total bit error rate increases. The selection set S7 represents all 32 bits (i.e., without a bit selection technique) and has the highest bit error rate of 11.13%.

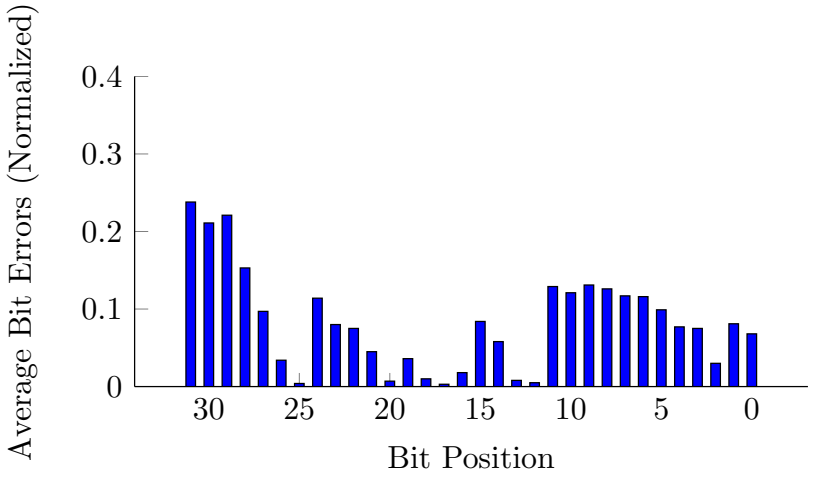
To determine which selection set is the most suitable for NBTI mitigation while fulfilling the maximum error tolerance value, it is important to take into consideration the spread of the probability of storing a '1'. Therefore, $\pm 3\sigma$ is used to give the worst spread of the probability of storing '1' and to select the best selection set that fulfils the criterion. The $\pm 3\sigma$ probability of storing '1' is calculated based on the probability of storing '1' from Figure 5.5(a). Further, the procedure given in Section 5.4 is re-applied to calculate the average bit error for every bit position at $\pm 3\sigma$. Figure 5.10 depicts an average bit error rates computed based on $\pm 3\sigma$ probability of storing '1'. Then, the total bit error rates for selection sets S1-S7 are calculated as listed in Table 5.3.

Table 5.2: Bit error (%) of bit selection combination based on mean probability of storing '1'

Selection Set	LB	UB	Selected Bits	Intra-HD_{NBTI} (%)
S1	0.40	0.60	12,13,16,20,23,25	3.30
S2	0.35	0.65	S1 + 2,17,18,19,21	4.98
S3	0.30	0.70	S2 + 0,3,24,26	6.28
S4	0.25	0.75	S3 + 1,4,5,14,22,27	8.01
S5	0.20	0.80	S4 + 6,15,30	8.93
S6	0.15	0.85	S5 + 7,8,9,10,11,29,31	10.83
S7	0.10	0.90	S6 + 28	11.13



(a) -3σ



(b) $+3\sigma$

Figure 5.10: Average bit errors based on the $\pm 3\sigma$ probability of storing '1'.

From Table 5.3, the overall bit error rate is worst at -3σ. In effect, all the bits would be shifted to the left in Figure 5.9, away from the 50% probability of storing '1'. At $+3\sigma$, all the bits are shifted to the right, closer to the 50% probability of storing '1', making the overall bit error rate better than the mean condition. A similar effect can also be seen in Figure 5.10. Notice from Table 5.3 that the bit error rates for all 32 bits (S7) are within the range of 8.35% to 14.18% after 5 years. Our ageing results are quite close to the experimental analysis reported elsewhere, [39, 103], where their analyses show that an SRAM-PUF suffers up to 8% bit error due to NBTI after 4.5 years using the same 65-nm technology. Elsewhere, NBTI causes up to 14% bit error on an SRAM-PUF after 4.7 years using a 90-nm technology [47]. We do not neglect the fact that the differences might arise from the data dependency of NBTI stress and the ageing model differences. Nevertheless, the effect of NBTI on SRAM cells and the overall trend still hold true.

Table 5.3: Bit error (%) of bit selection combination based on mean and $\pm 3\sigma$ probability of storing '1' after 5 years

Selection Set	Intra-HD_{NBTI} (%)		
	Mean	-3σ	+3σ
S1	3.30	5.58	2.02
S2	4.98	8.39	2.23
S3	6.28	9.39	3.58
S4	8.01	11.43	4.88
S5	8.93	12.14	5.98
S6	10.83	13.77	8.12
S7	11.13	14.18	8.35

With a bit selection technique, the bit error rate varies depending on the selection window. As mentioned earlier, the aim is to maximise the number of selected bits while keeping the error well below the maximum error tolerance value. For a proof of concept in our study, we set the maximum error tolerance due to NBTI at 6%. Therefore, we have chosen selection set S1 since the bit error rate is $< 6\%$, considering the probability of storing '1' for the mean and $\pm 3\sigma$ conditions. As discussed in Section 5.1.3, a reduction in the area of the ECC can be achieved if the error is minimised. Hence, it is expected that S1 could reduce the area overhead of the ECC since the bit error rate is much smaller than without a bit selection technique (i.e., using all 32 bits).

The above analysis shows that by selecting only those bits that experience balanced stress when the SRAM is being used as a memory, we can reduce the overall bit error when the SRAM is used as a PUF. However, there are other sources of variations that could cause errors in the PUF's response, in particular, temperature and voltage fluctuations, [48]. The size of the ECC has to be determined by the maximum error considering ageing,

temperature and voltage variations. Therefore, the effect of environmental variations on set S1 and on all 32 bits, S7, will be discussed next.

5.5.1 Temperature and Voltage Variations

The setup described in Section 5.4.1 is used to simulate the temperature and a supply voltage variations. A measured response at a nominal supply voltage of 1.2V and a temperature of 25°C has been used as a reference. Equation (2.2) is used to calculate the bit error due to the temperature and supply voltage fluctuations. Table 5.4 shows the bit error percentage for the temperature ranging from −40°C to 85°C and a supply voltage range of 1.2V± 10%. The maximum bit error caused by the temperature and supply voltage fluctuations is 6.14%. The bit error value in our findings is comparable to the experimental value reported elsewhere, [39, 114], where the maximum error rate was about 8% with the same temperature range and at a nominal supply voltage of 1.2V for a 65-nm technology node.

Table 5.4: Bit error (%) comparison at different temperatures and supply voltages

Conditions	Selection Set	
	S7	S1
$1.2V, -40°C$	5.91	5.90
$1.2V, 85°C$	1.31	1.32
$1.32V, -40°C$	6.14	6.12
$1.32V, 85°C$	1.31	1.32
$1.08V, -40°C$	5.65	5.67
$1.08V, 85°C$	1.31	1.32

From our analysis, we observe that the variations (1.2V ± 10% at nominal temperature of 25^oC) in the final voltage to which the supply voltage is ramped have almost no impact on the reliability of the SUVs; this is consistent with that seen elsewhere, [47, 104]. Instead, the reliability of SUVs is influenced by the ramp-up time of the supply voltage [47, 101]. Figure 5.11 shows the bit error rates at different ramp-up times from 100μs to 100ms with a nominal supply voltage of 1.2V at 25°C. A measured response with a ramp-up time of 50μs has been used as a reference, to which all other measured responses have been compared. The bit error rates for both S7 and S1 configurations are about the same for all ramp rates. As suggested elsewhere, [101], this analysis is useful when the PUF is characterised at a particular ramp rate but evaluated at a different ramp rate in the field.

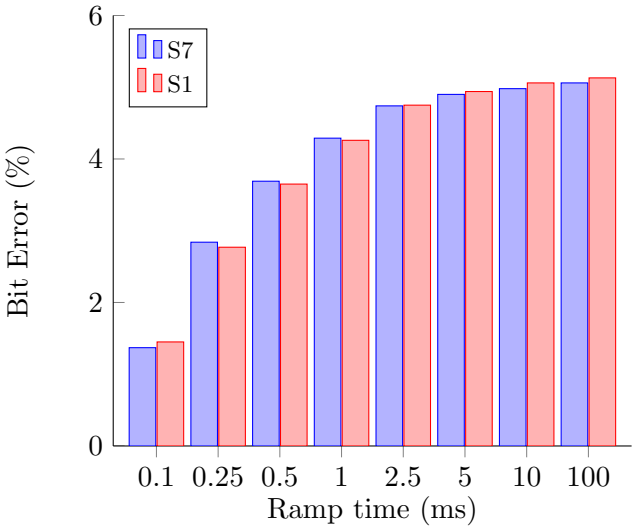


Figure 5.11: Bit error rate at different ramp-up time.

5.5.2 Uniqueness and Uniformity

As discussed earlier in Section 5.1.2, under prolonged storage of the same bit pattern, NBTI causes unreliable SUVs in which the value is less likely to be preserved but flips to the opposite value. Hence, the impact of unreliable SUVs on the uniqueness and uniformity of an SRAM-PUF due to NBTI is discussed in this section. Based on Table 5.3, the highest bit error rate due to NBTI occurs at the condition of -3σ. Therefore, the uniqueness and uniformity are compared between fresh SUVs and aged SUVs for the -3σ probability of storing '1'. Uniqueness is measured by the Inter-HD using Eq. (2.1). Figure 5.12 shows the uniqueness for the S1 set fresh and after 5 years ageing. It can be seen from the value of the mean (μ) and standard deviation (σ) for both conditions, that the Inter-HD values are distributed around 0.5. This shows that the flipping in the SUVs due to NBTI is random. Our findings are consistent with [48] that ageing has a negligible effect on the uniqueness of the PUF.

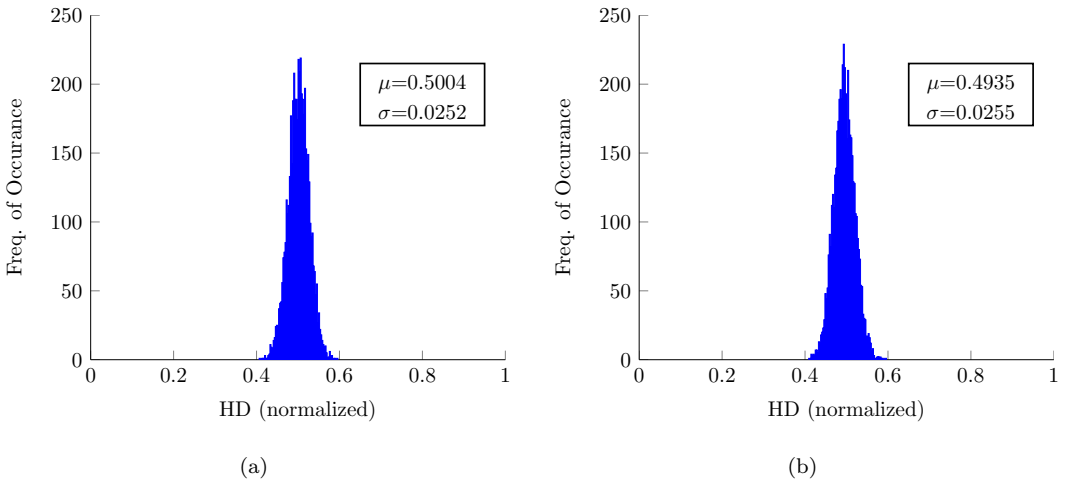


Figure 5.12: Uniqueness for S1 set (a) fresh (b) 5 years ageing at the -3σ probability.

Figure 5.13 shows the mean and standard deviation for uniformity of the S1 set fresh and after 5 years ageing which have been calculated using Eq. (2.4). As indicated, the uniformity of S1 at fabrication time is well distributed around 0.5. However, after 5 years, the uniformity is shifted slightly to the right indicating that the number of 1's in the SUVs distribution has increased. As can be noted from Figure 5.5, all the bits in the S1 set have a mean probability close to but lower than a 50% probability of storing '1'. By considering -3σ probability, means the probability of storing '1' for S1 is further reduced. Therefore, after NBTI, all the bits in an S1 set are likely to power-up to a '1'. Hence, this explains an increase in the number of 1's in the SUVs distribution and its effect on the uniformity. On the other hand, from our analysis, the uniformity for aged SUVs at the $+3\sigma$ probability condition is shifted to the left with the mean of 0.4803 and standard deviation of 0.0266, indicating less 1's in the SUVs distribution. Overall, the uniformity of the S1 set before and after ageing is still close to an ideal value of 50%. A similar trend in the uniqueness and uniformity has been observed for the S7 set fresh and after 5 years ageing.

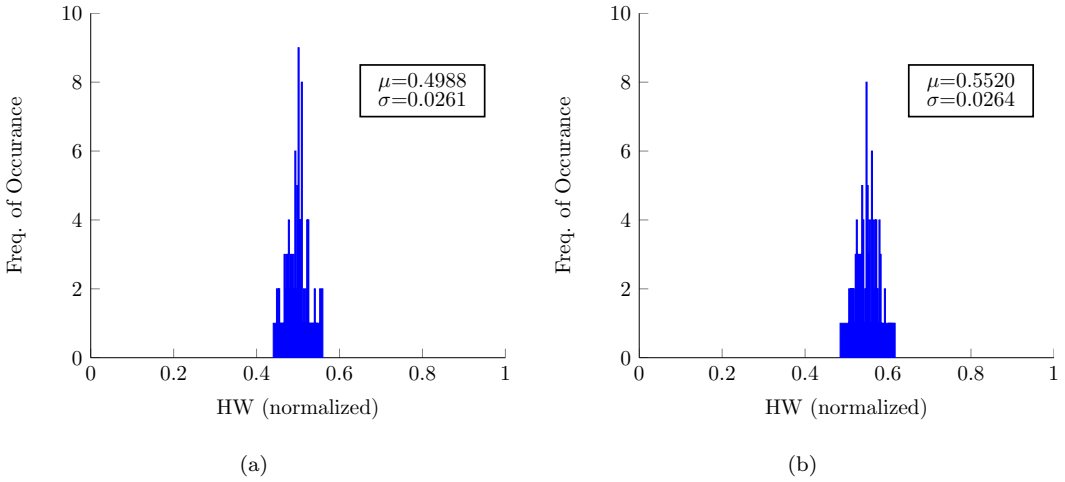


Figure 5.13: Uniformity for S1 set (a) fresh (b) 5 years ageing at the -3σ probability.

All the above analysis leads to the conclusion that only selecting bits from the i-cache that experience symmetric stress improves the overall bit error rates. However, the selected bits are still susceptible to temperature and supply voltage variations. Our analysis shows that generally, the bit errors due to NBTI are significant compared to the bit errors caused by the temperature and supply voltage variations. In addition, the uniqueness and uniformity of a bit selection technique before and after ageing remain close to an ideal value of 50%.

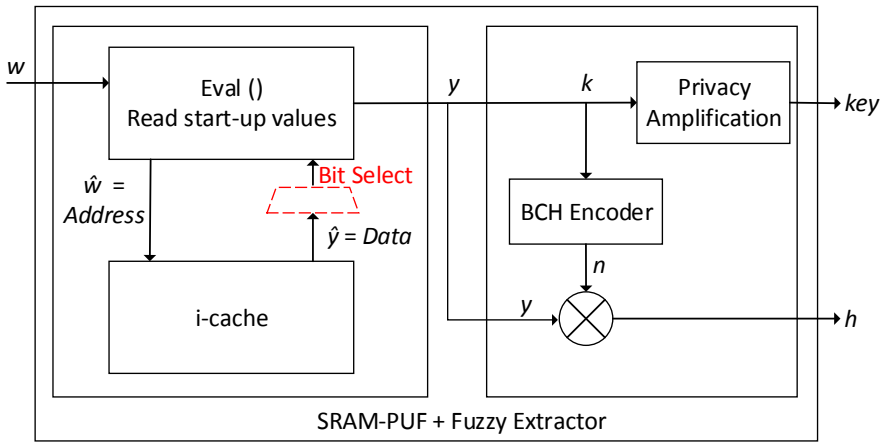
5.6 Hardware Cost and Implementation

We have proposed a bit error reduction technique to overcome unreliable responses due to NBTI, but there will still be a finite error rate. The reliability of SUVs is further influenced by the environmental variations such as temperature and supply voltage as investigated in Section 5.5.1. As mentioned in Section 5.1.3, ECC is one of the ways to achieve error-free cryptographic keys. Bit errors from both NBTI and environmental variations may impact the complexity of the error correction circuit. Therefore, in this section, the area overhead to implement bit select configurations and the overhead of ECC to overcome the bit errors are discussed and compared.

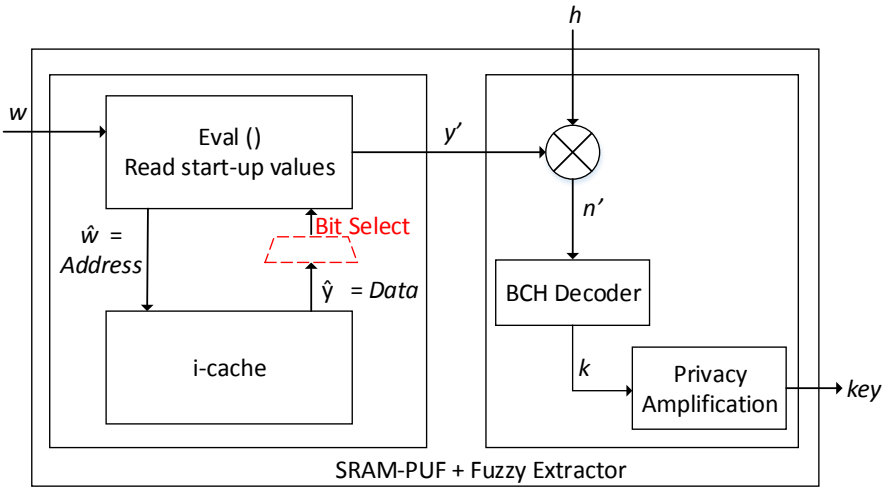
5.6.1 Area Overhead of Bit Select Configurations

The bit selection can be implemented using a multiplexer in which the select signals are pre-configured depending on the target error rate and bit select configuration. A 32-to-1 multiplexer is required to select 1 bit out of 32 bits from the output bus of the i-cache.

Thus, the bit select block can be constructed using a 6×32-to-1 multiplexer for selecting the S1 bits: 16, 13, 25, 20, 12, and 23. In the procedure for cryptographic key generation shown in Figure 5.14, the bit select block is inserted before the SUVs $\widehat{y}$ are input into the evaluation block (or can be inserted within the evaluation block), highlighted in the dashed trapezoid. 280 GEs are required for a 6×32-to-1 multiplexer, estimated using Synopsys Design Compiler.



(a)



(b)

Figure 5.14: Implementation of a bit selection technique.

5.6.2 Area Overhead of ECC

As discussed in Section 5.1.3, a BCH scheme is used as an ECC to generate error-free cryptographic keys. For a $[n, k, t]$-BCH, the probability of failure for a BCH scheme to correct t errors is given as, [23]:

$$Target\_Failure\_Rate = 1 - \sum_{i=0}^{t} \binom{n}{i} p_b^i (1 - p_b)^{n-i} \tag{5.1}$$

where p_b denotes the bit error probability. From the bit error analysis of the S1 set with respect to NBTI, temperature and supply voltage variations (Sections 5.5 and 5.5.1), the maximum bit error of 6.12% is caused by the temperature and supply voltage variations and hence, will be used to determine the number of raw bits, n, required for the BCH scheme. According to [23, 64], to ensure a sufficient entropy, an input to the privacy amplification must be larger than the generated key. Therefore, for this calculation, we refer to [23, 64], where the BCH scheme has to generate at least 171 error-free bits before being used as the input to the privacy amplification to produce a 128-bit cryptographic key (Figure 5.14). We assume a target failure rate of $\leq 10^{-6}$. An optimum size of BCH scheme has been calculated using Eq. (5.1) with a MATLAB script (Appendix A.3). 1524 raw bits are needed for the S1 selection set using a [127,15,27]-BCH. By selecting 6 out of 32 bits for every cache line, the total number of reliable bits that can be extracted from an 8kB SRAM i-cache is 12288. Thus, there are enough bits for an ECC to generate a 128-bit key. The suggested minimum size of an i-cache to be used with the S1 selection is 1kB.

Table 5.5: Area comparison

Technique	BCH (GE)	Bit Selection (GE)	Total (GE)
Without Bit selection	226224	NA	226224
Bit selection	37050	280	37330

Without using a bit selection technique would require more area for the BCH scheme because of an increase in the bit error rates. Considering the bit error rates due to ageing and environmental variations (see Table 5.3 and 5.4), using all 32 bits would yield a maximum error of 14.18%. This requires 4599 raw bits using the BCH scheme of [511,19,119] for a target failure rate of $\leq 10^{-6}$ and a minimum i-cache size of 1kB. The area for both BCH schemes above is estimated based on the extrapolated data of Figure 5.2 and it is listed in Table 5.5, together with the area overhead of a bit selection technique. As can be seen in Table 5.5, by using a bit selection technique, the total area overhead is about 6× smaller compared to that without a bit selection technique. Notice that the area overhead of the multiplexer (3rd column in Table 5.5) to perform a bit selection technique is almost negligible. This comparison proves that the bit selection technique helps in reducing the area overhead of the ECC.

5.7 Summary

An SRAM-PUF is a potential solution for a hardware-based secure key generation. Recent literature suggests reusing the on-chip SRAM in a system as a PUF to achieve better cost efficiency and enables the widespread adoption of SRAM-PUFs. However, dual use of SRAM as a memory and a PUF turns out to be less straightforward than expected due to ageing-induced NBTI. When SRAM is used as a memory, NBTI causes asymmetric V_{th} degradation, which impacts on the reliability of the SUVs. From our analysis, NBTI stress is unevenly distributed in a 32-bit ARM i-cache, but there are similar predictable patterns for different applications that can be exploited to generate reliable SUVs.

In this chapter, we propose a bit selection technique to mitigate the NBTI effect. We select bits that have close to a 50% probability of storing '1' values. Thus, both pMOS transistors in the SRAM cell age at the same rate and so keep the intrinsic mismatch. Our experiments show that selecting 6 bits from the i-cache of an ARM architecture reduced the bit error rate from 14.18% to 5.58% over 5 years. By using a bit selection technique, the bit error rate reduces and the area overhead of the ECC is 6× smaller compared to that without a bit selection technique, with a minimum size requirement of a 1kB i-cache. The area overhead to implement the bit selection is negligible with respect to the reduction in the area overhead of the ECC.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

PUFs have been proposed for lightweight IC identification and authentication through a challenge-and-response protocol, and for cryptographic key generation. Nevertheless, CMOS device ageing, particularly NBTI, presents a challenge for PUFs which has an irreversible effect leading to a permanent reliability issue (i.e., bit errors) of a PUF's response. In a challenge-and-response protocol, an optimum value of the HD threshold, ϵ, is required to balance the trade-off between reliability requirements and vulnerability to an ML-attack. To achieve an optimum value of ϵ, one can either reduce the bit error rates or reduce the vulnerability to an ML-attack. Nevertheless, it is always better to achieve a reduction in both criteria. In cryptographic key generation application, an ECC is required to generate error-free keys. However, the area of the ECC increases as the bit error rates of PUF increases which can be seen as a disadvantage for lightweight applications. Hence, it is desirable to achieve low bit errors for cost-efficient PUF-based key generation. Following the above issues, three research objectives have been established as described in Section 1.4, which focused on a reliable and secure PUFs for lightweight applications.

The first and second objectives are fulfilled by the characterisation of PUF and the analysis of the ageing impact on PUF-based differential architectures, as discussed in Chapter 3. A TCO-PUF which exploits the non-linear current-voltage behaviour in the sub-threshold region is implemented using a TSMC 65-nm technology and its quality metrics are characterised. The simulation results show that the mean value of uniqueness and uniformity are close to 50% but the TCO-PUF suffers from a lack of randomness in its response, resulting in a high standard deviation in the distribution of both metrics. On average, TCO-PUF achieves 92.38% reliability under the temperature variations of -40^{o}C to 85^{o}C and a supply voltage variation of 1.2V $\pm$ 10%. As for the vulnerability to an ML-attack, the TCO-PUF can be predicted with a very high accuracy of about 96%

using an ANN algorithm. Furthermore, we have investigated the impact of NBTI on PUF-based differential architectures such as TCO-PUF and Arbiter-PUF. The results show that the reliability of the TCO-PUF and Arbiter-PUF goes down with age and they experience bit errors of about 4.5% and 2.41%, respectively after 10 years. Ageing has a huge impact on both RO-PUF and SRAM-PUF. The RO-PUF degrades about 12.76% in 10 years and the SRAM-PUF degrades about 7% in 4.5 years. In comparison with the aforementioned PUFs, TCO-PUF and Arbiter-PUF are inherently ageing-resilient as they adopted a differential architecture. Hence, a differential design technique is preferred for the PUF implementation to increase the robustness to CMOS device ageing.

The second objective is also fulfilled in Chapter 5, in which a bit selection technique is proposed to mitigate the ageing impact on SRAM-PUFs. As discussed in Chapter 5, a better cost efficiency in a cryptographic key generation application can be achieved by reusing SRAM as both a memory and PUF. However, dual use of SRAM as memory and PUF is not straightforward because ageing-induced NBTI results in asymmetric degradation of memory bit cells, which impacts on the reliability of the SUVs over time. We have investigated the NBTI stress in a 32-bit ARM i-cache and found that there are similar predictable patterns for different applications that can be exploited to generate reliable SUVs. Hence, we proposed a bit selection technique to mitigate the NBTI effect by selecting bits that have close to a 50% probability of storing '1' values, such that both pMOS transistors in the SRAM cell age at the same rate and the intrinsic mismatch (e.g., V_{th} variations) are preserved. By using a bit selection technique, the bit error rate reduces from 14.18% to 5.58% over 5 years. As a result, the area overhead of the ECC reduces by 6 times compared to that without a bit selection technique, with a minimum size requirement of a 1kB i-cache.

The third objective is achieved by the challenge permutation technique to increase the Strong PUF resistant to an ML-attack, as proposed in Chapter 4. The technique has been implemented on an Arbiter-PUF using a TSMC 65-nm technology. The simulation results show that a challenge permutation technique alters the output transition probability of the Arbiter-PUF. As a result, the CRP mapping complexity increases and reduces the predictability of the Arbiter-PUF responses from ≈99% to ≈65%. For a 128-bit identifier, a challenge permutation technique uses only 1130 and 1512 GEs when implemented with the 32-bit and 64-bit Arbiter-PUFs, respectively. The power consumption of the respective Arbiter-PUFs is 0.2492mW and 0.3949mW. Hence, this technique is suitable for resource-constrained security devices.

6.2 Future Work

Based on the findings presented in this thesis, several directions for future research are identified and described in this section:

1. We have analysed the impact of ageing on the TCO-PUF and the Arbiter-PUF in Chapter 3. The analysis is based on the simulation using HSPICE MOSRA. The simulation results show that a finite bit error still occurs, possibly as a result of second order effect between V_{th} variations (i.e., due to the fabrication process) and ageing phenomena. To support the simulation findings and further understand the second order effect, the TCO-PUF and Arbiter-PUF have to be fabricated on a silicon and an accelerated ageing test has to be performed. Next, both results from simulation and experiment will be compared.

2. In Chapter 4, we have proposed a challenge permutation technique to increase the resilience of an Arbiter-PUF against an ML-attack. Although this technique successfully increases the unpredictability of an Arbiter-PUF, the challenge permutation has to be implemented at the IC design stage. Hence, the integrity of the IC manufacturer is of utmost importance. With no guarantee of the integrity, a technique to detect a software cloned PUF (i.e., through an ML-attack) must be developed. The exploration of this technique poses a great challenge to differentiate a feature(s) between a software cloned and a genuine PUF. An effort along this direction will complete the investigation of an ML-attack on Strong PUFs.

3. Chapter 3 and 4 focused on the reliability and security of Strong PUFs used for lightweight IC identification and authentication. The findings in both chapters can be used to achieve an optimum value of the HD threshold, ϵ, for the challenge-and-response protocol to identify and authenticate an IC. However, as discussed in Section 2.7.1, the scalability problem in the verifier database to record a significantly large number of the CRPs for IC identification and authentication is still an open challenge. It is an advantage for a PUF technology if the concept of public verifiability can be deployed such that no CRP databases are required to verify PUF responses.

Appendix A

MATLAB Code

A.1 Uniqueness

```
% Author: Mohd Syafiq Mispan
% Program: Compute the uniqueness (Inter-HD) for N PUF instances

% clear workspace and command window
clc;
clear;

% read the n-bit response stored in .csv file
M = dlmread('.\Response.csv');

% N PUF instances, each PUF instance has n-bit response
[N,n]=size(M);

% count the combination nCk
C = combnk(1:N,2);
len = length(C);

% convert to string in order to perform HD operation
strData= strtrim(cellstr(num2str(M)));

% calculate HD
count=1;
HD{:}=zeros;
HD_norm{:}=zeros;
HD_percent{:}=zeros;
for i = 1:N-1
    k=i;
    for k = i:N-1
        HD{count}=sum(strData{i}~=strData{k+1});
        HD_norm{count}=HD{count}/n;
        HD_percent{count}=((HD{count})/n)*100;
        count=count+1;
    end
end

% calculate inter-HD
mat_HD_percent=cell2mat(HD_percent);
```

```
inter_HD=(2/(N*(N-1)))*sum(mat_HD_percent);

% calculate mean & std deviation of HD
mat_HD=cell2mat(HD);
mean_HD=mean2(mat_HD);
std_HD=std2(mat_HD);
% plot histogram
histogram(mat_HD);
% add title and axis labels
title('Inter-HD (Uniqueness)')
xlabel('Hamming Distance (HD)')
ylabel('Frequency of Occurances')
% number of occurances
unqdis=unique(mat_HD);
countOCC=histc(mat_HD,unqdis);
data = (vertcat(unqdis,countOCC))'; %plot histogram in excel

% calculate mean & std deviation of NORMALIZE HD
mat_HD_norm=cell2mat(HD_norm);
mean_HD_norm=mean2(mat_HD_norm);
std_HD_norm=std2(mat_HD_norm);
% number of occurances of NORMALIZE HD
unqdis_norm=unique(mat_HD_norm);
countOCC_norm=histc(mat_HD_norm,unqdis_norm);
data_norm = (vertcat(unqdis_norm,countOCC_norm))'; %plot histogram in excel
```

A.2 Uniformity

```
% Author: Mohd Syafiq Mispan
% Program: Compute the uniformity (the distribution of 1's and 0's) for N PUF
    ↪ instances

% clear command window & workspace
clc;
clear;

% read the n-bit response stored in .csv file
M = dlmread('.\Response.csv');

% N PUF instances, each PUF instance has n-bit response
[N,n]=size(M);

% calculate the uniformity - Hamming Weight (HW)
HW{:}=zeros;
HW_norm{:}=zeros;
for i=1: N
    HW{i}=sum(M(i,:));
    HW_norm{i}=HW{i}/n;
end

% calculate mean & std deviation of HW
mat_HW=cell2mat(HW);
mean_HW=mean2(mat_HW);
std_HW=std2(mat_HW);
% calculate number of occurances
unqweight=unique(mat_HW);
```

```
countOCC=histc(mat_HW,unqweight);
data = (vertcat(unqweight,countOCC))'; %plot histogram in excel
% plot histogram
histogram(mat_HW);
% add title and axis labels
title('Uniformity')
xlabel('Hamming Weight (HW)')
ylabel('Frequency of Occurances')

% calculate mean & std deviation of NORMALIZE HW
mat_HW_norm=cell2mat(HW_norm);
mean_HW_norm=mean2(mat_HW_norm);
std_HW_norm=std2(mat_HW_norm);
% calculate number of occurances
unqweight_norm=unique(mat_HW_norm);
countOCC_norm=histc(mat_HW_norm,unqweight_norm);
data_norm = (vertcat(unqweight_norm,countOCC_norm))'; %plot histogram in excel
```

A.3 Optimum Size of BCH scheme

```
% Author: Mohd Syafiq Mispan
% Program:Compute an optimum BCH scheme given the required key, BER and
    ↪ probability error

% clear the command window and the workspace
clc;
clear;

%-----------------Start of User Inputs-----------------

% input to the privacy amplification or a key.
key = 171;
% maximum error considering aging, temp. and voltage variations
max_error = 0.0615;
% target failure rate, the probability that the system fails to generate error-
    ↪ free key.
BER = 1E-6;

% set the file which contains [n,k,t]
M=dlmread('../Simulation of SRAM PUF/BCH_n=127.csv');

%-----------------End of User Inputs-------------------

% extract n,k and t
[row_M,col_M]=size(M);
n=M(:,1);
k=M(:,2);
t=M(:,3);

error=t/n;
error=error(:,1);

power{:}=zeros;
power_raw{:}=zeros;
for i=1:row_M
    if(error(i,:)> max_error)
```

```
        power_raw{i}=key/k(i,:);
        power{i}=ceil(power_raw{i});    %round up the value
    else
        power_raw{i}=0;
        power{i}=0;
    end
end

% calculate the failure rate, max failure rate or BER = 1E-6
failure_rate{:}=zeros;
for j=1:row_M
    failure_rate{j}=1-binocdf(t(j,:),n(j,:),max_error)^power{j};
end

% calculate the # of raw bits
raw_bit{:}=zeros;
for x=1:row_M
    if(failure_rate{x}<BER)
        raw_bit{x}=n(x,:)*power{x};
    else
        raw_bit{x}=0;
    end
end

% combine all the data
power_raw=(cell2mat(power_raw))';
power=(cell2mat(power))';
failure_rate=(cell2mat(failure_rate))';
raw_bit=(cell2mat(raw_bit))';
data=horzcat(power_raw,power,failure_rate,raw_bit);
```

A.4 Linear Feedback Shift Register (LFSR) Fibonacci

```
% Author: Mohd Syafiq Mispan
% Program: 32-bit LFSR Fibonacci

% clear workspace and command window
clc;
clear;

%-------------User Inputs--------------------%
% set the generator polynomial (primitive)
% reference: https://uk.mathworks.com/help/comm/ref/commsrc.pn.html
g = [1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1];

% set the initial mask
% 1 = read value for current state/register
init_mask =[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];

% set the length of output, given maximal length LFSR (2^n)-1
NoOfOutBits = 10000;

% set the size of the seed
% the seed file contains > 10000 seeds
sz_seed=1;
```

```
% read and open the file that contains seed for LFSR
filename='seed.dat';

%------------End of User Inputs--------------%

fid1 = fopen (filename);

seedData = textscan(fid1,'%[^\n\r]');
seedData = seedData{:};

% size of generator polynomial
sz_poly=size(g);

% pre-allocate array
seedMat{:}=zeros;
k{:}=zeros;
temp_k={};

for i=1:sz_seed
seedMat{i} = seedData{i}-'0';
end

% random number generation
for i=1:sz_seed
init = seedMat{i};
curr = init;
mask = init_mask;

 for n=1:(sz_poly(2)-1)
    h = commsrc.pn('GenPoly',g,'InitialStates', init,'CurrentStates', curr,'Mask'
    ↪ ,mask,'NumBitsOut',NoOfOutBits);
    k{n}=h.generate()';
    % shift the masking bit - reading the value for each register
    mask(n)=0;
    mask(n+1)=1;
 end

k_trans=k';
temp_k=[temp_k,k_trans];
end

% cell array to matrix array
challenge=(cell2mat(temp_k))';

% numeric to string
[row_R,col_R]=size(challenge);
str{:}=zeros;
for i=1:row_R
array_str=num2str(challenge(i,:));
array_str(isspace(array_str)) = '';
str{i}=array_str;
end

% test the uniqueness of LFSR's output
unq = unique(str);

% write the LFSR's output into .dat file
fid2=fopen('LFSR_out.dat','w');
for i=1:size(str,2)
```

```
    fprintf(fid2,'%s\r\n',str{i});
end

% write LFSR's output into .csv file
outfile='LFSR_out.csv';
dlmwrite(outfile,challenge);

fclose(fid1);
fclose(fid2);
```

A.5 Output Transition Probability

A.5.1 Non-permutated and Permutated CRPs

```
% Author: Mohd Syafiq Mispan
% Program: Computation of output transition probability for non-permutated
% and permutated CRPs

% clear workspace and command window
clc;
clear;

% n-bit Arbiter-PUF
n=32;

% read file <n-bit challenge><1-bit response>
% challenges start with Cn,Cn-1 . . . C2,C1
% ref. challenges
infile1='..\CRPs.csv';
% HD=1 for every challenge bit index compared to ref. challenges
% 1st 1000 CRPs correspond to HD=1 for Cn
% 2nd 1000 CRPs correspond to HD=1 for Cn-1
% and so on so forth....
infile2='..\CRPs_HD1.csv';

% save the permutated CRPs into a file
outfile='CRPs_perm.csv';

% original mapping
map=sort(randperm(n));

% pre-defined permutation mapping
% map = [5 6 7 8 1 2 3 4 13 14 15 16 9 10 11 12 21 22 23 24 17 18 19 20 29 30 31
    ↪ 32 25 26 27 28];

% OMIT IF ORIGINAL MAPPING
% CRPs based on pre-defined permutation
G_CRPs=CRPs_Permutation_Func(infile1,map);
HD_CRPs=CRPs_Permutation_Func(infile2,map);

% compute output transition probability
prob=HDT_et_APUF_func(G_CRPs,HD_CRPs,map);

% plot the output transition probability
figure;
```

```
index=sort(map);
plot(index,prob,'b--o');

% write permutated CRPs to a file
dlmwrite(outfile,G_CRPs);
```

```
% Author: Mohd Syafiq Mispan
% Program: CRPs generation for challenge permutation experiment

function [CRPs_perm] = CRPs_Permutation_Func(infile,map)

% read file <n-bit challenge><1-bit response>
% total n+1 column
disp('Read non-permutated CRPs...');
M = dlmread(infile);

% define the challange and response
[row_M,col_M]=size(M);
C=M(:,1:col_M-1);
R=M(:,col_M);

% permutate the challenge
disp('Permutate the challenge..');
C_perm = zeros(row_M,col_M-1);
fprintf('----Expected Work done...%%%6.2f', 0);
for i = 1:row_M
    C_perm(i,:) = permutation(C(i,:),map);
    fprintf('\b\b\b\b\b\b%6.2f', i*100/row_M);
end
fprintf('\n');

% permutated CRPs
CRPs_perm=horzcat(C_perm,R);
end
```

```
% Author: Mohd Syafiq Mispan
% Program: Permutate the input vector (vec_org)

function [vec_per] = permutation(vec_org,map)

vec_per = vec_org(map);

end
```

```
% Author: Mohd Syafiq Mispan
% Program: Computation of output transition probability w.r.t a flipping
% in every challenge bit index

function [prob_flip]= HDT_et_APUF_func(G_CRPs,HD_CRPs,map)

% initialize the counter
count=0;

% read golden CRPs
% only the first 1000 CRPs are selected
CG = G_CRPs(1:1000,1:end);

% read the CRPs in which HD=1 for every challenge bit index as compared to golden
    ↪ CRPs
```

```
CHD1=HD_CRPs;

% generate index C1 to Cn
% index 1 refer to Cn, index n refer to C1
index_init = zeros(1,size(CG,2)-1);
for m=1:(size(CG,2)-1) %excluded the response
index_init(:,m)=m;
end

% get the permutation mapping
mapping = permutation(index_init,map);

% compute the output transition probability for each challenge bit index
prob=zeros((size(CG,2)-1),1);
for k=1:size(CG,2)-1      % n-bit Arbiter-PUF

    j=mapping(k);
    CHD1_index = CHD1((1+1000*(j-1)):1000*j,1:end);
    C_COMB=vertcat(CG,CHD1_index);

    % sort ascending for each column/index given k
    index=index_init;
    index(k) = [];
    C_COMB_sorted=sortrows(C_COMB,index);

    %----------------------For debug ONLY-----------------
    %sub=minus(C_COMB_sorted(1,1:end-1),C_COMB_sorted(2,1:end-1));
    %sub(sub==-1)=1;
    %index_HD_debug(k,:) = find(sub);
    %HD=sum(sub,2);
    %----------------------End of debug----------------------

    % check the response
    for i=1:(size(C_COMB_sorted,1)/2)
        if(C_COMB_sorted((i+(i-1)),end)~=C_COMB_sorted(2*i,end))
        count=count+1;
        end
    end

    % output signal transition probability
    prob(k,:)=count/(size(C_COMB_sorted,1)/2);

    % reset the counter after each index k
    count=0;
end

% flipud is used because the challenges in .csv start with Cn
prob_flip=flipud(prob);
```

A.5.2 Iterative Permutation

```
% Author: Mohd Syafiq Mispan
% Program:Finding a good permutation

% clear workspace and command window
clc;
clear;
```

```
% n-bit Arbiter-PUF
n=32;

% read file <n-bit challenge><1-bit response>
% challenges start with Cn,Cn-1 . . . C2,C1
% ref. challenges
infile1='..\CRPs.csv';
% HD=1 for every challenge bit index compared to ref. challenges
% 1st 1000 CRPs correspond to HD=1 for Cn
% 2nd 1000 CRPs correspond to HD=1 for Cn-1
% and so on so forth....
infile2='..\CRPs_HD1.csv';

% save the permutated CRPs into a file
outfile='CRPs_perm.csv';

% save the random permutation mapping into a file
outmap='perm_map';

% initial condition
target=100; %dummy value
G{:}=zeros;
while target~=1

% bB random permutation
index=sort(randperm(n));
b=4;    %set the # of bits to be grouped
if (b>1)
    % group the challenge bit index
    group = n/b;
    for i=1:group
        G{i} = index(:,(1+b*(i-1)):b*i);
    end
    map_group=randperm(group);
    map=cell2mat(G(map_group));
else
    % 1B random permutation
    map=randperm(n);
end

% CRPs based on random permutation
G_CRPs=CRPs_Permutation_Func(infile1,map);
HD_CRPs=CRPs_Permutation_Func(infile2,map);

% compute output transition probability
prob=HDT_et_APUF_func(G_CRPs,HD_CRPs,map);

% reset the counter
count=0;
for i=1:n-2 % exclude 1st and last index
    prev_prob=prob(i,:);
    curr_prob=prob(i+1,:);
    next_prob=prob(i+2,:);

    if((prev_prob<=curr_prob) && (curr_prob<=next_prob))
        delta_CP=curr_prob-prev_prob;
        delta_NC=next_prob-curr_prob;
        if(delta_CP<=0.08 && delta_NC<=0.08)
```

```
        count=count+1;
        end
    elseif((prev_prob>=curr_prob) && (curr_prob>=next_prob))
        delta_PC=prev_prob-curr_prob;
        delta_CN=curr_prob-next_prob;
        if(delta_PC<=0.08 && delta_CN<=0.08)
        count=count+1;
        end
    end
end
target=count;
end

% save the random permutation mapping
save(outmap,'map');

% plot the output transition probability
figure;
index=sort(map);
plot(index,prob,'b--o');

% write permutated CRPs to a file
dlmwrite(outfile,G_CRPs);
```

A.6 ML-attack

```
%Author: Mohd Syafiq Mispan
%Program: ML-attack using FeedForward Neural Network

%Clear the command window and workspace
clc;
clear;

%-------------------------start-of-user-input-----------------%
%Filename that contains challenge-response pairs (CRPs)
%Data in the file must be in the format <challenge><response>
filename='CRPs.csv';

%Training algorithm: resilient backpropagation
training={'trainrp'};

%Size of the training and test set
test_size=2000;
training_size=10000;

%The training set incrementally increases with a pre-defined step_size
step_size = 1000;

% # of bit per challenge
bit_C=16;

% # of bit per response
bit_R=1;

%-------------------------end-of-user-input-----------------%
```

```
%Read the data set
M = dlmread(filename);
[row_M,col_M]=size(M);

%Check the pre-defined training_size, test_size and step_size
if((training_size+test_size)>row_M)
    error('The total of pre-defined training and test sets must be =< %d.',row_M)
    ↪ ;
elseif(mod(training_size,step_size)~=0)
    val=(floor(training_size/step_size))*step_size;
    error('For step_size=%d, please use training_size=%d.',step_size,val);
end

%Define the challenge from data set
M_challenge=M(:,1:bit_C);

%Transfrom the challenge input into feature vector, according to:
%D. Lim, MSc. Thesis, Massachusetts Institute of Technology, 2004
vector_mat=1-2*(M_challenge);
parity_array{:}=zeros;
for i=1:row_M
    for k=1:(bit_C)
        parity_array{i,k}=prod(vector_mat(i,k:bit_C));
    end
end
M_challenge=cell2mat(parity_array);
M_challenge_T=M_challenge';

%Define the response from data set
M_response=M(:,(bit_C+1));
M_response_T=M_response';

%Test set
test_challenge=M_challenge_T(:,(row_M-test_size+1):row_M);
test_response=M_response_T(:,(row_M-test_size+1):row_M);

%Number of iteration
iter=(training_size)/step_size;

%Initialize array
c{:}=zeros;
cm{:}=zeros;

for i=1:size(training,2)

    for j=1:iter

        %Training set
        train_challenge=M_challenge_T(:,1:step_size*j);
        train_response=M_response_T(:,1:step_size*j);

        %For reproducibility - fix the seed
        setdemorandstream(491218382);

        %Feed forward neural networks
        net = feedforwardnet([32],training{i});

        %Training parameters
        net.trainParam.epochs = 1000000;    % Default value is 1000
```

```
        net.trainParam.max_fail = 15;       % Default value is 6
        %Default values
        net.trainParam.goal = 0.0;
        net.trainParam.min_grad = 1e-5;

        %Initializes the weights and biases of the network
        %Can also skip this step and go directly to train the network
        %train command will automatically configure the network
        net = configure(net,train_challenge,train_response);

        %Divide the training set into training and validation
        %to avoid overfitting
        net.divideParam.trainRatio = 85/100;
        net.divideParam.valRatio = 15/100;
        net.divideParam.testRatio = 0/100;

        %Building the feed forward network - use parallel computing
        [net,tr] = train(net,train_challenge,train_response,'useParallel','yes','
    ↪ showResources','yes');

        %Testing the network
        predicted_response = net(test_challenge,'useParallel','yes','
    ↪ showResources','yes');
        testIndices = vec2ind(predicted_response);

        %Measure the performance of the network
        [c{j},cm{j}] = confusion(test_response,predicted_response);
        train_size(:,j)=size(tr.trainInd,2);
        val_size(:,j)=size(tr.valInd,2);
        cc(:,j) = 100*(1-c{j});

    end
end

%Plot the prediction accuracy vs # of training data
plot((train_size+val_size),cc,'c*');
ylim([0 100]);
title('ML-attack on 16-bit Arbiter-PUF');
xlabel('# CRPs ');
ylabel('Prediction Accuracy [%]');
```

Appendix B

Verilog Code

B.1 Hardware Implementation of Figure 4.13

```
//**********************************************************************
// Verilog file: top_level_apuf.v
// Program: Top level hierarchy consists of controller, 32-bit LFSR,
//          32-bit Arbiter-PUF, and 128-bit SIPO
// **********************************************************************
`timescale 1ns/1ps

module top_level_apuf#(
parameter WIDTH_C=32,
parameter WIDTH_R=128,
parameter APUF_SIZE=32
)

(input clk,
 input rst,
 input [WIDTH_C-1:0] challenge,
 output [WIDTH_R-1:0] response,
 output reg done
);

wire [WIDTH_C-1:0] sub_challenge;
wire pulse;
wire clk_sipo;
wire out;
reg en_puf;
reg en_sipo;
reg [7:0] counter;

//instantiate 32-bit LFSR
lfsr_32bit uut_lfsr_32bit(
.rst (rst),
.clk (clk),
.i (challenge),
.o (sub_challenge)
);

//instantiate 64-bit LFSR
```

```
//lfsr_64bit uut_lfsr_64bit(
//.rst (rst),
//.clk (clk),
//.i (challenge),
//.o (sub_challenge)
//);

//instanstiate n-bit Arbiter-PUF
arbiter_puf#(APUF_SIZE) uut_apuf(
.c (sub_challenge),
.pulse0 (pulse),
.pulse1 (pulse),
.out (out)
);

//instantiate SIPO
sipo_async#(WIDTH_R) uut_sipo(
.clk (clk_sipo),
.rst (rst),
.ShiftIn (out),
.o_shift (response)
);

//--------------------Start of Controller----------------
assign pulse = en_puf & clk;
assign clk_sipo = en_sipo & ~clk;

always@(posedge clk)
begin
  if(rst)
    begin
    counter <= 0;
    en_puf <= 0;
    en_sipo <= 0;
    done <= 0;
    end
  else if(!done)
    begin
    counter <= counter + 1'b1;
    en_puf <= 1;
    en_sipo <= 1;
      if(counter==WIDTH_R) //max 128 clock cycles
        begin
        counter <= counter;
        en_puf <= 0;
        en_sipo <= 0;
        done <=1;
        end
    end
end
//-------------------End  of Controller--------------------
endmodule
```

```
//************************************************************
// Verilog file: arbiter_puf.v
// Program: Generate N-bit Arbiter-PUF
// ************************************************************
`timescale 1ns/1ps
```

```
module arbiter_puf #(
parameter N=32
)

(input [N-1:0] c,
 input pulse0,
 input pulse1,
 output out
);

wire [N-1:0] i0;
wire [N-1:0] i1;
wire [N-1:0] o0;
wire [N-1:0] o1;

//Instantiate switch_component
switch_component uut1[N-1:0](
.c(c),
.i0(i0),
.i1(i1),
.o0(o0),
.o1(o1));

//A rising pulse apply at the 1st switch_component
//pulse0 and pulse1 are coming from the same source at top level
assign {i0[0], i1[0]} = {pulse0, pulse1};

//Assign internal signals
assign {i0[N-1:1], i1[N-1:1]} = {o0[N-2:0], o1[N-2:0]};

//Instantiate SR nand latch
SR_nand_latch uut2 (
.S(o0[N-1]),
.R(o1[N-1]),
.Q(out));

endmodule
```

```
//*********************************************************************
// Verilog file: switch_component.v
// Program: Delay component of Arbiter-PUF consists of 2x 2-to-1 muxes
// *********************************************************************
`timescale 1ns/1ns

module switch_component
(input c,
 input i0,
 input i1,
 output o0,
 output o1
);

assign {o0,o1} = c?{i1,i0}:{i0,i1};

endmodule
```

```
//*********************************************************************
// Verilog file: SR_nand_latch.v
// Program: Set-reset latch (nand type)
```

```
// ***********************************************************************
`timescale 1ns/1ps

module SR_nand_latch
(input S,
 input R,
 output Q,
 output Qbar
);

nand (Q, S, Qbar);
nand (Qbar, R, Q);

endmodule
```

```
//***********************************************************************
// Verilog file: lfsr_32bit.v
// Program: Generate 32-bit linear feedback shift register (LFSR)
//***********************************************************************
`timescale 1ns/1ps

module lfsr_32bit
(input clk,
 input rst,
 input [31:0] i,
 output reg [31:0] o
 );

wire feedback;
wire [31:0] o_temp;
reg sel;

//multiplexer
assign o_temp = sel? o:i;

//continous or concurrent assignment followed by a net
//always active and assignments occur whenever the right-hand side operands
    ↪ changes
//tap configuration follows https://uk.mathworks.com/help/comm/ref/commsrc.pn.
    ↪ html
assign feedback = (o_temp[31]^o_temp[30]^o_temp[10]^o_temp[0]);

always @(posedge clk)
 begin
   if(rst) begin
     o <= 0;
     sel <= 0; end
   else begin
     sel <= 1;
     o <= {feedback,o_temp[31:1]}; end
 end
endmodule
```

```
//***********************************************************************
// Verilog file: lfsr_64bit.v
// Program: Generate 64-bit linear feedback shift register (LFSR)
//***********************************************************************
`timescale 1ns/1ps
```

```
module lfsr_64bit
(input clk,
 input rst,
 input [63:0] i,
 output reg [63:0] o
 );

wire feedback;
wire [63:0] o_temp;
reg sel;

//multiplexer
assign o_temp = sel? o:i;

//continous or concurrent assignment followed by a net
//always active and assignments occur whenever the right-hand side operands
    ↪ changes
//tap configuration follows https://uk.mathworks.com/help/comm/ref/commsrc.pn.
    ↪ html
assign feedback = (o_temp[63]^o_temp[61]^o_temp[60]);

always @(posedge clk)
 begin
   if(rst) begin
     o <= 0;
     sel <= 0; end
   else begin
     sel <= 1;
     o <= {feedback,o_temp[63:1]}; end
 end
endmodule
```

```
//////////////////////////////////////////////////////////////
// Verilog file: sipo_async.v
// Program: Asynchronous shift left register;
//          serial in and parallel out
// ////////////////////////////////////////////////////////////
`timescale 1ns/1ns

module sipo_async #(
//Parameterized value
parameter N = 32
)

(input  clk,
 input  rst,
 input  ShiftIn,
 output [N-1:0] o_shift
 );

reg [N-1:0] shift_reg;

always @(posedge clk or posedge rst)
begin
  if (rst)
          shift_reg <= {N{1'b0}};
        else
          shift_reg <= {shift_reg[N-2:0], ShiftIn};
end
```

```
//concurrent assignment.
assign o_shift = shift_reg;

endmodule
```

Appendix C

Miscellaneous

C.1 Reliability of 2-XOR Arbiter-PUF

Table C.1: Simplification of input-output transition probability

A	B	t_{XOR}
$0 \to 0$	$0 \to 0$	$0 \to 0$
$0 \to 0$	$0 \to 1$	$0 \to 1$
$0 \to 0$	$1 \to 0$	$1 \to 0$
$0 \to 0$	$1 \to 1$	$1 \to 1$
$0 \to 1$	$0 \to 0$	$0 \to 1$
$0 \to 1$	$0 \to 1$	$0 \to 0$
$0 \to 1$	$1 \to 0$	$1 \to 1$
$0 \to 1$	$1 \to 1$	$1 \to 0$
$1 \to 0$	$0 \to 0$	$1 \to 0$
$1 \to 0$	$0 \to 1$	$1 \to 1$
$1 \to 0$	$1 \to 0$	$0 \to 0$
$1 \to 0$	$1 \to 1$	$0 \to 1$
$1 \to 1$	$0 \to 0$	$1 \to 1$
$1 \to 1$	$0 \to 1$	$1 \to 0$
$1 \to 1$	$1 \to 0$	$0 \to 1$
$1 \to 1$	$1 \to 1$	$0 \to 0$

$\Longrightarrow$

A	B	t_{XOR}
$0 \to 1$	$0 \to 1$	$0 \to 0$
$0 \to 1$	$1 \to 0$	$1 \to 1$
$1 \to 0$	$0 \to 1$	$1 \to 1$
$1 \to 0$	$1 \to 0$	$0 \to 0$
$0 \to 1$	$0 \to 0$	$0 \to 1$
$1 \to 0$	$0 \to 0$	$1 \to 0$
$0 \to 1$	$1 \to 1$	$1 \to 0$
$1 \to 0$	$1 \to 1$	$0 \to 1$
$0 \to 0$	$0 \to 1$	$0 \to 1$
$0 \to 0$	$1 \to 0$	$1 \to 0$
$1 \to 1$	$0 \to 1$	$1 \to 0$
$1 \to 1$	$1 \to 0$	$0 \to 1$
$0 \to 0$	$0 \to 0$	$0 \to 0$
$0 \to 0$	$1 \to 1$	$1 \to 1$
$1 \to 1$	$0 \to 0$	$1 \to 1$
$1 \to 1$	$1 \to 1$	$0 \to 0$

$\Longrightarrow$

A	B	t_{XOR}
U	U	R
U	R	U
R	U	U
R	R	R

"U" represents unreliable transitions of $0 \to 1$ or $1 \to 0$ and "R" represents reliable transitions of $0 \to 0$ or $1 \to 1$. Given the average reliability of an Arbiter-PUF, the reliability for 2-XOR Arbiter-PUF can be computed as:

$$P(t_{XOR} = reliable) = 1 - P(t_{XOR} = unreliable) \tag{C.1}$$
$$= 1 - (P(A \cup B) - P(A \cap B)) \tag{C.2}$$
$$= 1 - (P(A) + P(B) - P(A)P(B) - P(A)P(B)) \tag{C.3}$$
$$= 1 - (P(A) + P(B) - 2P(A)P(B)) \tag{C.4}$$

C.2 BCH code

Table C.2: Number of correctable errors in the BCH Code for n=127

Index	n	k	t
1	127	120	1
2	127	113	2
3	127	106	3
4	127	99	4
5	127	92	5
6	127	85	6
7	127	78	7
8	127	71	9
9	127	64	10
10	127	57	11
11	127	50	13
12	127	43	14
13	127	36	15
14	127	29	21
15	127	22	23
16	127	15	27
17	127	8	31

Table C.3: Number of correctable errors in the BCH Code for n=255

Index	n	k	t
1	255	247	1
2	255	239	2
3	255	231	3
4	255	223	4
5	255	215	5
6	255	207	6
7	255	199	7
8	255	191	8
9	255	187	9
10	255	179	10
11	255	171	11
12	255	163	12
13	255	155	13
14	255	147	14
15	255	139	15
16	255	131	18
17	255	123	19
18	255	115	21
19	255	107	22
20	255	99	23
21	255	91	25
22	255	87	26
23	255	79	27
24	255	71	29
25	255	63	30
26	255	55	31
27	255	47	42
28	255	45	43
29	255	37	45
30	255	29	47
31	255	21	55
32	255	13	59
33	255	9	63

Table C.4: Number of correctable errors in the BCH Code for n=511

Index	n	k	t	Index	n	k	t
1	511	502	1	30	511	241	36
2	511	493	2	31	511	238	37
3	511	484	3	32	511	229	38
4	511	475	4	33	511	220	39
5	511	466	5	34	511	211	41
6	511	457	6	35	511	202	42
7	511	448	7	36	511	193	43
8	511	439	8	37	511	184	45
9	511	430	9	38	511	175	46
10	511	421	10	39	511	166	47
11	511	412	11	40	511	157	51
12	511	403	12	41	511	148	53
13	511	394	13	42	511	139	54
14	511	385	14	43	511	130	55
15	511	376	15	44	511	121	58
16	511	367	16	45	511	112	59
17	511	358	18	46	511	103	61
18	511	349	19	47	511	94	62
19	511	340	20	48	511	85	63
20	511	331	21	49	511	76	85
21	511	322	22	50	511	67	87
22	511	313	23	51	511	58	91
23	511	304	25	52	511	49	93
24	511	295	26	53	511	40	95
25	511	286	27	54	511	31	109
26	511	277	28	55	511	28	111
27	511	268	29	56	511	19	119
28	511	259	30	57	511	10	121
29	511	250	31				

References

[1] B. Halak, Y. Hu, and M. S. Mispan, "Area efficient configurable physical unclonable functions for FPGAs identification," in *IEEE International Symposium on Circuits and Systems*, 2015, pp. 946–949.

[2] M. S. Mispan, B. Halak, Z. Chen, and M. Zwolinski, "TCO-PUF: A subthreshold physical unclonable function," in *IEEE PRIME*, 2015, pp. 105–108.

[3] M. S. Mispan, B. Halak, and M. Zwolinski, "NBTI aging evaluation of PUF-based differential architectures," in *IEEE International Symposium on On-Line Testing and Robust System Design*, 2016, pp. 103–108.

[4] B. Halak, M. Zwolinski, and M. S. Mispan, "Overview of PUF-based hardware security solutions for the Internet of Things," in *IEEE Midwest Symposium on Circuits and Systems*, 2016, pp. 1–4.

[5] M. S. Mispan, B. Halak, and M. Zwolinski, "Lightweight obfuscation techniques for modeling attacks resistant PUFs," in *IEEE International Verification and Security Workshop*, 2017, pp. 19–24.

[6] M. S. Mispan, H. Su, M. Zwolinski, and B. Halak, "Cost-efficient designs for modeling attacks resistant PUFs," in *Design, Automation & Test in Europe Conference & Exhibition*, 2018, pp. 467–472.

[7] M. S. Mispan, S. Duan, M. Zwolinski, and B. Halak, "A reliable PUF in a dual function SRAM," in *International Symposium on Power and Timing Modeling, Optimization and Simulation*, 2018, pp. 1–6.

[8] Y. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavehei, D. Abbott, and D. C. Ranasinghe, "Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices," in *IEEE International Conference on Pervasive Computing and Communication Workshops*, 2016, pp. 1–6.

[9] V. V. D. Leest, R. Maes, G.-J. S. Pim, and P. Tuyls, "Hardware intrinsic security to protect value in the mobile market," in *Information Security Solutions Europe*, 2014, pp. 188–198.

[10] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A survey on lightweight entity authentication with strong PUFs," *ACM Computing Surveys*, vol. 48, no. 2, pp. 26:1–26:42, 2015.

[11] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "Emerging physical unclonable functions with nanotechnology," *IEEE Access*, vol. 4, no. PP, pp. 61–80, 2016.

[12] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium on VLSI Circuits Digest of Technical Papers*, 2004, pp. 176–179.

[13] Intrinsic ID., "Products," 2016. [Online]. Available: http://www.intrinsicid.com/products/

[14] NXP Semiconductors N.V., "PUF - Physical Unclonable Functions: Protecting next-generation smart card ics with sram-based pufs," 2013. [Online]. Available: http://www.nxp.com/documents/other/75017366.pdf

[15] K. B. Sutaria, S. Member, J. B. Velamala, C. H. Kim, S. Member, T. Sato, and Y. Cao, "Aging statistics based on trapping / detrapping : Compact modeling and silicon validation," *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 2, pp. 607–615, 2014.

[16] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *ACM Conference on Computer and Communications Security*, 2002, pp. 148–160.

[17] A. Maiti, "A systematic approach to design an efficient physical unclonable function," Ph.D. dissertation, Virginia Polytechnic Institute and Stage University, 2012.

[18] K. Verma, B. Kaushik, and R. Singh, "Effects of process variation in VLSI interconnects - a technical review," *Microelectronics International*, vol. 26, no. 3, pp. 49–55, 2009.

[19] T. Siddiqua, S. Gurumurthi, and M. R. Stan, "Modeling and analyzing NBTI in the presence of process variation," in *International Symposium on Quality Electronic Design*, 2011, pp. 28–35.

[20] Y. Ye, F. Liu, M. Chen, S. Nassif, and Y. Cao, "Statistical modeling and simulation of threshold variation under random dopant fluctuations and line-edge roughness," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 6, pp. 987–996, 2011.

[21] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[22] G. E. Suh and S. Devadas, "Physical Unclonable Functions for device authentication and secret key generation," in *ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[23] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2007, pp. 63–80.

[24] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[25] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls, "Extended abstract : The Butterfly PUF protecting IP on every FPGA," in *IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 67–70.

[26] P. Simons, E. Van Der Sluis, and V. Van Der Leest, "Buskeeper PUFs, a promising alternative to D Flip-Flop PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2012, pp. 7–12.

[27] B. Gassend, M. van Dijk, D. Clarke, E. Torlak, and S. Devadas, "Controlled physical random functions and applications," *ACM Transactions on Information and System Security*, vol. 10, no. 4, pp. 15:1 –15:22, 2008.

[28] 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," *IEEE Transactions on Information Forensic and Security*, vol. 8, pp. 1876–1891, 2013.

[29] R. Pappu, "Physical one-way functions," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[30] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. V. Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[31] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 670–673.

[32] V. van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls, "Hardware intrinsic security from d flip-flops," in *ACM Workshop on Scalable Trusted Computing*, 2010, pp. 53–62.

[33] Y. Su, J. Holleman, S. Member, B. P. Otis, and A. Abstract, "A digital 1.6 pJ/bit chip identification circuit using process variations," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 69–77, 2008.

[34] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *Design, Automation & Test in Europe Conference & Exhibition*, 2015, pp. 653–658.

[35] R. Kumar and W. Burleson, "On design of a highly secure PUF based on non-linear current mirrors," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 38–43.

[36] D. Lim, "Extracting secret keys from integrated circuits," MSc. Thesis, Massachusetts Institute of Technology, 2004.

[37] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security: Foundations and Practice*, A.-R. Sadeghi and D. Naccache, Eds. Berlin, Heidelberg: Springer, 2010, pp. 3–37.

[38] L. Lin, S. Srivathsa, D. K. Krishnappa, P. Shabadi, and W. Burleson, "Design and validation of Arbiter-based PUFs for sub-45nm low-power security applications," *IEEE Transactions on Information Forensic and Security*, vol. 7, no. 4, pp. 1394–1403, 2012.

[39] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. V. D. Sluis, and V. Van Der Leest, "Experimental evaluation of physically unclonable functions in 65 nm CMOS," in *Proceedings of the European Solid-State Circuits Conference*, 2012, pp. 486–489.

[40] Y. Cao, L. Zhang, S. S. Zalivaka, C.-h. Chang, and S. Chen, "CMOS image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Transactions on Circuits and Systems*, vol. 62, no. 11, pp. 2629–2640, 2015.

[41] T. Saha and V. Sehwag, "TV-PUF: A fast lightweight analog physical unclonable function," in *IEEE International Symposium on Nanoelectronic and Information Systems*, 2016, pp. 182–186.

[42] S. T. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. F. Wong, "CNPUF: A carbon nanotube-based physically unclonable function for secure low-energy hardware design," in *Asia and South Pacific Design Automation Conference*, 2014, pp. 73–78.

[43] L. Zhang, Z. H. Kong, C.-H. Chang, A. Cabrini, and G. Torelli, "Exploiting process variations and programming sensitivity of phase change memory for reconfigurable physical unclonable functions," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 6, pp. 921–932, 2014.

[44] Y. Gao, D. C. Ranasinghe, S. F. Al-Sarawi, O. Kavehei, and D. Abbott, "mrPUF: A novel memristive device based physical unclonable function," in *International Conference Applied Cryptography Network Security*, 2015, pp. 595–615.

[45] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds. New York: Springer New York, 2013, pp. 245–267.

[46] M. T. Rahman, F. Rahman, D. Forte, and M. Tehranipoor, "An aging-resistant RO-PUF for reliable key generation," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 3, pp. 335–348, 2016.

[47] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. De Groot, V. Van Der Leest, G. J. Schrijen, M. Van Hulst, and P. Tuyls, "Evaluation of 90nm 6T-SRAM as physical unclonable function for secure key generation in wireless sensor nodes," in *IEEE International Symposium on Circuits and Systems*, 2011, pp. 567–570.

[48] A. Maiti and P. Schaumont, "The impact of aging on a physical unclonable function," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1854–1864, 2014.

[49] S. Mukhopadhyay, N. Goel, and S. Mahapatra, "A comparative study of NBTI and PBTI using different experimental techniques," *IEEE Transactions on Electron Devices*, vol. 63, no. 10, pp. 4038–4045, 2016.

[50] D. S. Ang, Z. Q. Teo, T. J. J. Ho, and C. M. Ng, "Reassessing the mechanisms of negative-bias temperature instability by repetitive stress/relaxation experiments," *IEEE Transactions on Device and Materials Reliability*, vol. 11, no. 1, pp. 19–34, 2011.

[51] P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design.* New York: Oxford University Press, 2002.

[52] A. Garg and T. T. Kim, "Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect," in *IEEE International Symposium on Circuits and Systems*, 2014, pp. 1941–1944.

[53] T. Xu and M. Potkonjak, "Stable and secure delay-based physical unclonable functions using device aging," in *IEEE International Symposium on Circuits and Systems*, 2015, pp. 33–36.

[54] J. Kong and F. Koushanfar, "Processor-based strong physical unclonable functions with aging-based response tuning," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, pp. 16–29, 2014.

[55] G. T. Becker, "The gap between promise and reality: On the insecurity of XOR Arbiter PUFs," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2015, pp. 535–555.

[56] G. Hospodar, R. Maes, and I. Verbauwhede, "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *IEEE International Workshop on Information Forensics and Security*, 2012, pp. 37–42.

[57] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Transactions on Emerging Topics in Computing*, vol. 2, pp. 37–49, 2014.

[58] U. Rührmair, F. Sehnke, J. Solter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *ACM Conference on Computer and Communications Security*, 2010, pp. 237–249.

[59] H. Handschuh, "Hardware-anchored security based on SRAM PUFs, Part 1," *IEEE Security and Privacy*, vol. 10, no. 3, pp. 80–83, 2012.

[60] S. Zeitouni, Y. Oren, C. Wachsmann, P. Koeberl, and A.-r. Sadeghi, "Remanence decay side-channel: The PUF case," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1106–1116, 2016.

[61] M. Bhargava and K. Mai, "An efficient reliable PUF-based cryptographic key generator in 65nm CMOS," in *Design, Automation & Test in Europe Conference & Exhibition*, 2014, pp. 1–6.

[62] Y. Lao, B. Yuan, C. H. Kim, and K. K. Parhi, "Reliable PUF-based local authentication with self-correction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 2, pp. 201–213, 2017.

[63] Y. Alkabani, "Remote activation of ICs for piracy prevention and digital right management," in *IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 674–677.

[64] A. Schaller, T. Arul, V. Van Der Leest, and S. Katzenbeisser, "Lightweight anticounterfeiting solution for low-end commodity hardware using inherent PUFs," in *Trust and Trustworthy Computing*. Springer International Publishing, 2014, pp. 83–100.

[65] F. Kohnhäuser, A. Schaller, and S. Katzenbeisser, "PUF-based software protection for low-end embedded devices," in *Trust and Trustworthy Computing*, M. Conti, M. Schunter, and I. Askoxylakis, Eds. Springer International Publishing, 2015, pp. 3–21.

[66] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using PUF-based logic," in *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers*, 2015, pp. 270–277.

[67] NXP Semiconductors N.V., "Step up security and innovation with next generation SmartMX2 products," 2016. [Online]. Available: https://cache.nxp.com/docs/en/brochure/75017695.pdf

[68] C. Helfmeier, C. Boit, D. Nedospasov, and J. P. Seifert, "Cloning physically unclonable functions," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2013, pp. 1–6.

[69] A. Schlösser, D. Nedospasov, J. Krämer, S. Orlic, and J.-P. Seifert, "Simple Photonic Emission Analysis of AES," in *Cryptographic Hardware and Embedded Systems - CHES 2012*, E. Prouff and P. Schaumont, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 41–57.

[70] A. Roelke and M. R. Stan, "Attacking an SRAM-based PUF through Wearout," in *IEEE Computer Society Annual Symposium on VLSI*, 2016, pp. 206–211.

[71] D. Nedospasov, J. P. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF analysis," in *Fault Diagnosis and Tolerance in Cryptography*, 2013, pp. 30–38.

[72] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich, "Physical characterization of Arbiter PUFs," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2014, pp. 493–509.

[73] U. Ruhrmair and J. Solter, "PUF modeling attacks: An introduction and overview," in *Design, Automation & Test in Europe Conference & Exhibition*, 2014, pp. 1–6.

[74] J. Delvaux and I. Verbauwhede, "Fault injection modeling attacks on 65 nm arbiter and RO Sum PUFs via environmental changes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 6, pp. 1701–1713, 2014.

[75] 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 *International Workshop on Cryptographic Hardware and Embedded Systems*, 2014, pp. 476–492.

[76] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong PUF : Possible or a pipe dream?" in *IEEE International Symposium on Hardware Oriented Security and Trust*, 2016, pp. 19–24.

[77] S. Lin, X. Zhao, B. Li, and X. Pan, "An ultra-low power common-source-amplifier-based physical unclonable function," in *IEEE International Conference on Electron Devices and Solid-State Circuits*, 2015, pp. 269–272.

[78] S. Stanzione, D. Puntin, and G. Iannaccone, "CMOS silicon physical unclonable functions based on intrinsic process variability," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1456–1463, 2011.

[79] Y. Cao, L. Zhang, C.-H. Chang, and S. Chen, "A low-power hybrid RO PUF with improved thermal stability for lightweight applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1143–1147, 2015.

[80] M. Kalyanaraman and M. Orshansky, "Novel strong PUF based on nonlinearity of MOSFET subthreshold operation," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2013, pp. 13–18.

[81] Z. Chen, "A sub-threshold physical unclonable function designed from circuit level," MSc. Thesis, University of Southampton, 2014.

[82] V. Vivekraja and L. Nazhandali, "Circuit-level techniques for reliable physically uncloneable functions," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2009, pp. 30–35.

[83] J. Li and M. Seok, "Ultra-compact and robust physically unclonable function based on voltage-compensated proportional-to-absolute-temperature voltage generators," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 9, pp. 2192–2202, 2016.

[84] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[85] A. Wang, B. H. Calhoun, and A. P. Chandrakasan, *Sub-threshold Design for Ultra Low-Power Systems*. Springer, 2006.

[86] Y. Taur and T. Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 2009.

[87] S. Babayan-Mashhadi and R. Lotfi, "Analysis and design of a low-voltage low-power double-tail comparator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 343–352, 2014.

[88] S. Katzenbeisser, Ü. Kocaba, V. Rožić, A.-R. Sadeghi, Ingrid Verbauwhede, and C. Wachsmann, "PUFs : Myth , fact or busted? A security evaluation of Physically Unclonable Functions (PUFs) cast in Silicon," in *Cryptographic Hardware and Embedded Systems – CHES 2012*, E. Prouff and P. Schaumont, Eds. Springer Berlin Heidelberg, 2012, pp. 283–301.

[89] B. Tudor, J. Wang, W. Liu, and Hany Elhak, "MOS Device Aging Analysis with HSPICE and CustomSim," California, USA, pp. 1–5, August 2011.

[90] J. Ye, Y. Hu, and X. Li, "OPUF: Obfuscation logic based physical unclonable function," in *IEEE International On-Line Testing Symposium*, 2015, pp. 156–161.

[91] ——, "RPUF: Physical unclonable function with randomized challenge to resist modeling attack," in *IEEE Asian Hardware Oriented Security and Trust Symposium*, 2016, pp. 1–6.

[92] J. Strömbergson, "sha256," https://github.com/secworks/sha256, 2013.

[93] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full SHA-1," in *International Cryptology Conference*, 2017, pp. 570–596.

[94] J. Heaton, *Introduction to Neural Networks for Java, 2nd Edition*, 2nd ed. Heaton Research, Inc., 2008.

[95] P. H. Nguyen, D. P. Sahoo, R. S. Chakraborty, and D. Mukhopadhyay, "Security analysis of Arbiter PUF and its lightweight compositions under predictibility test," *ACM Transactions on Design Automation of Electronic Systems*, vol. 22, no. 2, pp. 1–28, 2016.

[96] A. Peinado and A. Ortiz, "Prediction of sequences generated by LFSR using back propagation MLP," in *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, 2014, pp. 407–412.

[97] G. Nasierding and A. Z. Kouzani, "Comparative evaluation of multi-label classification methods," in *International Conference on Fuzzy Systems and Knowledge Discovery*, 2012, pp. 679–683.

[98] C. Hoffman, M. Cortes, D. F. Aranha, and G. Araujo, "Computer security by hardware-intrinsic authentication," in *International Conference on Hardware/-Software Codesign and System Synthesis*, 2015, pp. 143–152.

[99] Intrinsic ID, "SRAM PUF: The Secure Silicon Fingerprint," Eindhoven, Netherlands, June 2016.

[100] A. Bacha and R. Teodorescu, "Authenticache: Harnessing cache ECC for system authentication," in *International Symposium on Microarchitecture*, 2015, pp. 128–140.

[101] M. Bhargava, C. Cakir, and K. Mai, "Reliability enhancement of bi-stable PUFs in 65nm bulk CMOS," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2012, pp. 25–30.

[102] M. D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[103] R. Maes and V. van der Leest, "Countering the effects of silicon aging on SRAM PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2014, pp. 148–153.

[104] K. Xiao, T. Rahman, D. Forte, Y. Huang, M. Su, and M. M. Tehranipoor, "Bit selection algorithm suitable for high-volume production of SRAM-PUF," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2014, pp. 101–106.

[105] E. Jamro, "The design of a VHDL based synthesis tool for BCH Codecs," MSc. Thesis, University of Huddersfield, UK, 1997.

[106] C. Mutigwe, J. Kinyua, and F. Aghdasi, "Instruction set usage analysis for application-specific systems design," *International Journal of Information Technology & Computer Science*, vol. 7, no. 2, pp. 99–103, 2013.

[107] A. H. Ibrahim, M. B. Abdelhalim, H. Hussein, and A. Fahmy, "An analysis of x86-64 instruction set for optimization of system softwares," *International Journal of Advanced Computer Science*, vol. 1, no. 4, pp. 152–162, 2011.

[108] ARM Ltd., "Programmers Guide for ARMv8-A," 2015. [Online]. Available: http://infocenter.arm.com

[109] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[110] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *IEEE International Workshop on Workload Characterization*, 2001, pp. 3–14.

[111] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization*, 2004, pp. 75–86.

[112] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negatilve bias temperature instability," in *IEEE/ACM International Conference on Computer-Aided Design*, 2006, pp. 493–496.

[113] A. Gebregiorgis, M. Ebrahimi, S. Kiamehr, F. Oboril, S. Hamdioui, and M. B. Tahoori, "Aging mitigation in memory arrays using self-controlled bit-flipping technique," in *Asia and South Pacific Design Automation Conference*, 2015, pp. 231–236.

[114] M. Cortez, S. Hamdioui, V. Van Der Leest, R. Maes, and G. J. Schrijen, "Adapting voltage ramp-up time for temperature noise reduction on memory-based PUFs," in *IEEE International Symposium on Hardware-Oriented Security and Trust*, 2013, pp. 35–40.