PUFsecurity

Quantum Tunneling PUF

Basics, Circuits, and Security Applications

Kent Kai-Hsin Chuang

Book Series on Hardware Security
Founding Editor: Charles Ching-Hsiang Hsu

Quantum Tunneling PUF

Basics, Circuits, and Security Applications

Kent Kai-Hsin Chuang

Book Series on Hardware Security

Founding Editor: Charles Ching-Hsiang Hsu

Quantum Tunneling PUF: Basics, Circuits, and Security Applications

By Kent Kai-Hsin Chuang

PUF-based Security Solutions and Applications

By Lawrence Liu

Anti-Tampering Designs in Hardware Security

By Meng-Yi Wu, Kent Kai-Hsin Chuang

Random Number: Generation, Emulation, and Practice

By Balance Chun-Heng You, Kent Kai-Hsin Chuang

Contemporary Cryptography: Fundamentals and Algorithms

By Chun-Yuan Yu, Danny Yung Chih Chen, Wayne Wen-Ching Lin

TPM and HSM: Implementation and Applications

By Shih-Li Hsu

Quantum Tunneling PUF

Basics, Circuits, and Security Applications

Written by Kent Kai-Hsin Chuang
Foreword by Charles Ching-Hsiang Hsu
Edited by Ada Ying-Yun Huang, Andrew Irvin, Ann Yi-An Lin,
Evans Ching-Song Yang, Saga Chi-Yi Shao

Notice of Copyright

For further requests, please contact Info@pufsecurity.com**.**

Printed by
PUFsecurity Corporation

Second Edition, 2024

PUFsecurity

About the Author

Kent Kai-Hsin Chuang

Dr. Kent Kai-Hsin Chuang joined PUFsecurity Corporation in 2020, where he currently holds the position of a R&D manager in security technology. His current research interest includes Root of Trust circuits, PUF-based security solutions, and attack/protection techniques for cryptographic implementations.

Dr. Chuang received his Ph.D. degree in electrical engineering from KU Leuven, Belgium, in 2020. During his Ph.D. study, he worked in the COSIC research group of KU Leuven and the reliability research group of IMEC, where he was in charge of developing highly reliable PUFs in CMOS and emerging memory technologies. He holds 14 worldwide issued and pending patents and has more than 15 international publications. Several of his publications include “Highly Reliable Physically Unclonable Functions” and “A Physically Unclonable Function Soft Oxide Breakdown Featuring 0% Native BER and 51.8 fJ/bit in 40-nm CMOS.” He has been invited as a tutorial speaker at two IEEE conferences and also served as a reviewer of IEEE journals more than 20 times.

Foreword

"I was seldom able to see an opportunity until it had ceased to be one."

For many years, this quote from Mark Twain has been imprinted in my mind, reminding me that good ideas, whenever they arrive, must be seized to become genuine opportunities. So when I met Tang Ma (馬騰桂) in 2015, shortly after his retirement from Maxim, his explanation of the possibilities of PUF (Physically Unclonable Function) for semiconductors, and the critical role it could play in security, felt like an opportunity worth pursuing.

As a semiconductor device physicist, I immediately thought that the characteristics associated with a device's physical dimensions could potentially produce unclonable features, due to the natural variances during the fabrication process. The thickness and quality of the film, for example, or the length and breadth of a transistor gate, parameters such as those produce minute differences in a transistor's microstructure. When extracting and comparing the adjacent transistors' electrical characteristics, we can register the difference as 0 or 1 if it is distinct enough within the measurable region.

So, the critical factor is identifying the parameter with a linear difference in its geometry, yet the electrical behavior measured from the terminals is exponential. Drawing on my decades of work on electron transport in thin gate dielectrics, I immediately thought that the tunneling current between gate dielectrics would be significantly different, even if the thickness or quality of the dielectric retains only minimal variations.

To verify this idea, I brought Wei-Jer (翁偉哲), Meng-Yi (吳孟益), Hsin-Ming (陳信銘), and Evans (楊青松) to collaborate together, by setting up an experiment that used our existing NeoFuse (anti-fuse using oxide tunneling) with two oxide capacitors in parallel. The results produced two critical findings:

1. *Only one of the oxide capacitors will have a significant tunneling current when the applied voltage is large enough.*
2. *The occurrence of the tunneling current between the pair of oxide capacitors is random, arriving equally on either the left or right capacitor.*

I knew this technology would be foundational for securing the future of interconnected computing. Since then, these two critical findings have established the silicon fingerprint, NeoPUF, which led to the founding of our company, PUFsecurity, and the development of our integrated suite of security subsystems that play a vital role in the Hardware Security ecosystem. After establishing PUFsecurity, we were encouraged to try and enlighten the broader semiconductor community on the risk of unsecured chips through the education and promotion of Quantum Tunneling PUF, which ultimately led to publishing this book series to introduce the fundamentals of PUF-based Hardware Security.

These books will cover a wide range of topics, including Quantum Tunneling PUF, an overview of PUF-based solutions, tamperproof design, random number generation, and the importance of cryptography to Hardware Security. We will also cover applications like AI and IoT, as well as provide an overview of the latest security standards and regulations. Our hope is that they can play a role in furthering our collective understanding of Hardware Security and its impact on the future of computing.

As the internet enables more and more connected devices, deploying a traditional physically secure boundary for a system is no longer sufficient. We must embrace the principles of “Zero Trust,” explicitly verifying every linked device and decentralizing the secure boundaries of the network to a solution embedded on each device. It is similar to how we biometrically identify each of us through our fingerprints.

A PUF can play the role of a chip fingerprint, uniquely identifying both the device and the chip in which it is embedded. It can then be integrated with an NVM OTP to e stablish a Hardware Root of Trust that can generate, store, and safely manage Keys. Then, we can implement a Security Coprocessor by combining a PUF-based Hardware Root of Trust with Crypto Engines to provide a device with a fully integrated Security Subsystem.

I want to offer my sincerest gratitude to everyone that helped realize this project, in particular, its authors; Dr. Kent Chuang (莊愷莘), Lawrence Liu (劉持志), Dr. Meng-Yi Wu (吳孟益), Balance You (游鈞恆), Danny Chen (陳勇志), Dr. Wayne Lin (林文景), Chun-Yuan Yu (游鈞元), Dr. Li Hsu (徐世理) and Matthew Yu (于立宏). I would also like to thank Ada Huang (黃楹芸), Andrew Irvin

(張安筑), and Ann Lin (林奕安) for their diligent work editing, formatting, and publishing these books. Lastly, I would like to thank Dr. Evans Yang (楊青松) for his tireless dedication in guiding this project.

Thank you all.

Charles Ching-Hsiang Hsu
April, 2023

Preface

I still remember the first time I heard about Physically Unclonable Function (PUF), it was a complete fascination. Even without thinking deeply about how to use it, the fact that it can turn the undesired device variability into good use is already a shock in the head. After spending years working on academic and industrial research projects about PUFs, I am even more convinced that PUFs are indispensable in the future of hardware security. Now it is my turn to convince you to join the exploration in this fascinating field of research.

Through writing this book, I hope to give you a glimpse of what makes PUF so interesting and why it is so important to hardware security. For those who are not yet familiar with hardware security, this book aims to help you begin the journey throughout this field from its foundation. For more proficient readers, the goal is to provide you with in-depth insights about PUF and how it can be used to build various security applications.

This book begins with the basics of hardware security and PUF, including essential PUF properties, popular PUF implementations, and common challenges when designing PUFs. In this part, I want to emphasize the importance of PUF in hardware security, and more specifically, why we urge to seek even better PUF solutions.

Afterward, more advanced topics about how to design highly robust PUFs will be discussed, and in particular, the quantum tunneling PUF technology will be introduced. By showing these examples, I hope you can gain some insight into how to design or choose the right PUF solution for your security systems or applications.

Completing this book is not an easy task, and it would not be possible without the continuous help from many folks at PUFsecurity and eMemory. I would like to take the chance to thank all of you for giving valuable feedback and editorial support during the book writing.

For our readers, no matter how deeply you look into this book, I hope you can find these topics interesting and are willing to stay interested or even work further in the security field. Here at PUFsecurity, you are always welcome to look for more PUFacademy training material, online courses, technical consultations, or collaboration opportunities.

Kent Kai-Hsin Chuang
February, 2023

Hardware Security Overview

We are all eagerly embracing the technology revolution underway with the Internet of Things (IoT), artificial intelligence (AI), e-finance, and electric vehicles. However, we need to safeguard against the security risks they create. While these technologies are rapidly evolving, the number of connected devices will soar in the near future. Securing increasingly sophisticated device networks has become a critical security topic. To accomplish the security goals, it is important to make connected devices secure by design, which requires intensive effort to make hardware security a fundamental part of the conceptual stage for electronic devices and systems.

The primary consideration for hardware security is the use of dedicated components as gatekeepers in a system. This leads to three important research topics in this field: hardware acceleration, countermeasures against attacks, and Hardware Roots of Trust. Implementing cryptographic algorithms in hardware is typically faster than in software. Especially for public-key encryption algorithms and signature schemes, performing operations in software may be unacceptably slow within some platforms. Hardware implementations for cryptographic algorithms are inherently more efficient and elegant.

For edge devices deployed anywhere in the world, the risks of hijacking by malicious parties and vulnerability to attack are high. Under such circumstances, a device and the system on top of it, could be vulnerable even if standardized cryptographic algorithms and security protocols are applied. This weakness can be exploited by advanced attack techniques, including side-channel attacks [1], fault attacks [2], and invasive physical attacks [3]. Consequently, we need to understand possible vulnerabilities caused by new attack techniques and design relevant hardware countermeasures.

A Hardware Root of Trust (HRoT) should be the first element in the Chain of Trust of a security system to adequately protect the physical layer. It should provide an unpredictable and tamperproof secret that enables the required hardware security features, which, historically raises security issues around generation and storage.

Hardware Security Book Series Overview

PUFsecurity Corporation published a Hardware Security Book Series covering a wide range of topics on hardware security. The book series starts with introducing the essential knowledge about Physically Unclonable Functions (PUFs) and PUF-based Root of Trust (RoT) solutions. The next milestone is understanding how security applications can benefit from high-quality PUF and RoT. The next book in the series introduces several integrated PUF-based IPs as a fully comprehensive security solution. The discussions of these solutions will cover hardware architecture, functionality, specifications, and important use cases in Artificial Intelligence (AI) and the Internet of Things (IoT).

More details about generic RoT solutions using NeoPUF, a technology developed by eMemory Technology, are covered in two following books. One focuses on anti-tampering features of

NeoPUF and RoT solutions, including design techniques applied to mitigate common security threats. The other book describes true random number generators, their essential concepts, and demonstrations of their design integration within RoT solutions.

Three types of cryptography are introduced given their theory and design methods in circuit implementation, which include Cryptographic Hash Functions, Symmetric-Key Ciphers, and Public-Key Cryptography. It is crucial to understand the significant differences between algorithms executed in software compared to those in hardware. Finally, two modules, Trusted Platform Module (TPM) and Hardware Security Module (HSM), are illustrated in the last book. Through the specification explanation, we will learn the types of TPMs and HSMs that are secure enough to protect keys during the lifecycle and learn methods to improve the relevant designs of the modules.

This book series comprehensively addresses the methodologies, applications, and market insights related to the core technology in hardware security. It serves as a practical and handy tool for readers at all levels, from beginners to experts. Readers can acquire better and deeper understandings of PUF, RoT, PUF-based solutions, and hardware security, which will further assist them in pursuing excellence in academia and industry.

Contents

List of Figures

List of Tables

1

An overview of PUF concepts, including a definition of PUFs in integrated circuits, the main use cases for PUFs, and the required PUF properties for generic applications. The chapter concludes with key concepts for designing PUF circuits, accompanied by discussions on several well-known PUF implementations.

1. Physically Unclonable Functions

Since its invention two decades ago [4], the physically unclonable function (PUF) has offered an elegant new solution for tackling problems in hardware security. Consequently, PUFs have become increasingly important hardware primitives for various security applications. Providing an inborn and unique hardware signature for every chip, a PUF can significantly improve on-chip security and lower the cost of mass-production.

1.1. PUFs in Integrated Circuits

A PUF within a semiconductor hides secrets from all except those who have direct access. As a result, the data can be accessed on-demand anytime and re-evaluated by those with privileged access. In addition, these secrets are not directly stored in typical non-volatile memory (NVM), which is susceptible to tampering. In summary, a PUF can provide all the essential features to serve as an anchor of HRoT. While there are PUFs of different types, this book will only focus on those that are implemented in integrated circuits, or “silicon PUFs.” This section offers a general definition and the fundamental design concept behind PUFs.

1.1.1. Definition

A PUF is an object which cannot be physically cloned; i.e., no two identical PUFs are possible. This definition is too broad for PUFs used in hardware security applications. PUFs implemented in integrated circuits have more specific definitions proposed in the literature [5] [6]. Rather than joining arguments on what kind of circuit is unclonable, this book offers a functional definition.

> *“A PUF is a circuit which consists of several pre-defined properties, namely the PUF properties, and in most common cases, these are physical unclonability, evaluability, uniqueness, and reliability.”*

1.1.2. Device Variations

Regardless of how a PUF circuit is constructed, its uniqueness and physical unclonability must come from its unique physical characteristics within semiconductors. When discrete integrated circuits are made, each one has unique characteristics due to variations in the fabrication process. This is known as process variation. The effects caused by process variation are uncontrollable, resulting in a random and unique physical pattern on each chip that can be used to generate data. These generated data patterns are unrepeatable, meaning that such a data pattern cannot be physically cloned.

Besides the process variations during chip fabrication, there are time-dependent phenomena that can change the performance of a chip during various stages of its product life. These are known as aging effects. In semiconductors, including PUFs, aging effects can impact system operation. Still, under some circumstances, these aging effects can be exploited to implement a PUF because of their indeterministic nature.

1.2. Generic PUF-based Security Application

Serving as an essential HRoT building block, PUFs enable many security applications. The fundamental security applications include key generation and entity authentication.

1.2.1. Key Generation

Because each PUF has a unique pattern resulting from device variation, the multi-bit pattern can be exploited as the entropy source to derive a cryptographic key for encryption or signature purposes. In a cryptographic system, the same key needs to be accessed multiple times for encrypting, decrypting, or signing. To ensure retrieval of the same key when it is required, the conventional solution is storing the key in an embedded Non-Volatile Memory (NVM) block.

This key can be either injected from external sources or generated by an on-chip TRNG (true random number generator). The main security concerns for this scheme are the quality of the key and the security of the NVM-based key storage.

While there are some good practices for generating and storing keys, researchers still aim to find better solutions, which is where a PUF stands out. Since a PUF can produce unique data patterns from a chip, it provides the required unpredictability for cryptographic keys. Therefore, this unique data pattern, namely the PUF secret, is the basis of a key derivation function in standard cryptographic algorithms.

The PUF secret outputs a key of a specified length that can serve for cryptographic usages, as illustrated in *Figure 1-1*. Using a PUF to derive keys has the advantages of not requiring external key injection or on-chip random number generation, and not requiring NVM-based key storage.

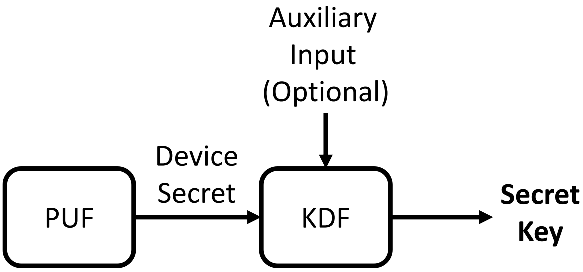


Figure 1-1 Simplified block diagram of a PUF-based key generation function.

1.2.2. Entity Authentication

Amid this IoT era, there is an urgent need for small, portable methods to securely authenticate internet-linked systems. For example, a challenge-response-based entity-authentication scheme can be enabled by a specific type of PUF, namely a strong PUF.

As illustrated in *Figure 1-2*, a server can authenticate a device by checking the device's response to a given challenge. This method allows internet entities to be authenticated through their unique PUF responses. However, this method is considered less practical because such a strong PUF that would meet all the requirements for entity authentication does not yet exist.

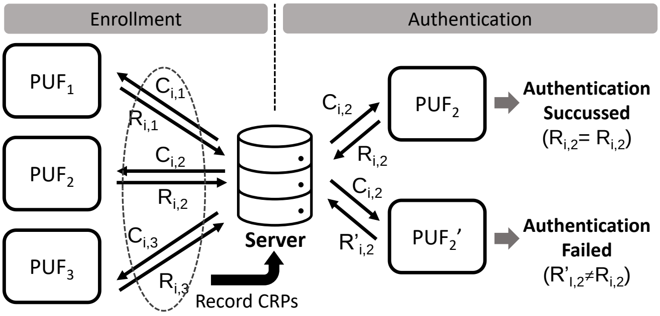


Figure 1-2 Enrollment and authentication flow using PUF.

1.3. PUF Properties

To properly characterize a PUF, all the required PUF properties must be evaluated. In addition to the four fundamental properties, there are a few additional ones that are usually considered essential. The PUF properties discussed in this subsection are uniqueness, randomness, reliability, physical unclonability, mathematical unclonability, tamper resistance, and tamper evidence.

1.3.1. Preliminaries

A set of definitions will aid the understanding of these PUF properties, including Min-entropy,Hamming Weight, and Hamming Distance.

Min-entropy

The min-entropy [7] of a random variable X is defined in the equation below, where x represents a sample of X, in the form of a binary vector with length N, and χ represents the set of all the possible outcomes. The min-entropy of the random variable X is determined by the maximum probability of guessing an outcome x correctly the first time. The numerical value of min-entropy is defined by taking a logarithmic scale on this maximum probability.

$$H_\infty = -\log\left\{\max_{x\in\chi}[\Pr(x = X)]\right\} \text{ (Min-Entropy)}$$

As illustrated in *Figure 1-3*, given a random variable's probability distribution, the min-entropy can be derived from the maximum value of the distribution. Following the definition and this example, min-entropy is the worst-case estimation of how a secret random variable can be predicted. A secret key with a length λ is required to have full entropy, i.e., $H_\infty = \lambda$, to fulfill the requirements for a cryptographic algorithm. The random variable X must be uniformly distributed to achieve full entropy.

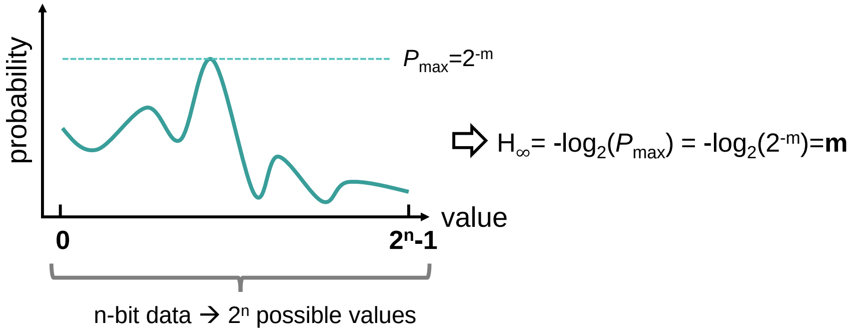


Figure 1-3 Example probability distribution and its subsequent min-entropy.

Hamming Weight

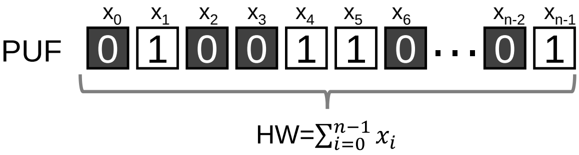

PUF
x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_{n-2} x_{n-1}
0 1 0 0 1 1 0 ··· 0 1
HW=$\sum_{i=0}^{n-1} x_i$

Figure 1-4 Example illustration of hamming weight calculation.

Hamming Weight (HW) is a commonly used representation of the number of ones in a binary vector. As illustrated in *Figure 1-4*, for a binary vector x of length N, $x = x_1, x_2, \ldots, x_N$, where $x_i \in \{0, 1\}$, Hamming Weight is defined as the following equation. It is also common to use the normalized Hamming Weight, which is defined as $HW_{norm}(x) = HW(x)/N$.

$$\mathrm{HW}(x) = \sum_{i=1}^{N} x_i \text{ (Hamming Weight)}$$

Hamming Distance

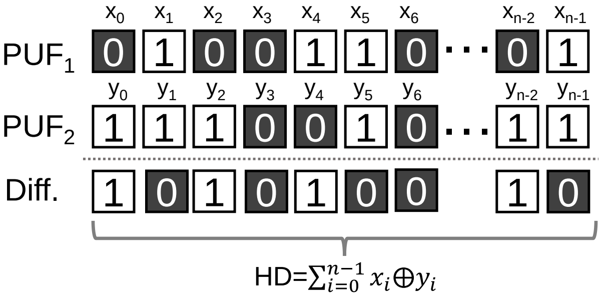

x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_{n-2} x_{n-1}
PUF$_1$
0 1 0 0 1 1 0 ··· 0 1
y_0 y_1 y_2 y_3 y_4 y_5 y_6 y_{n-2} y_{n-1}
PUF$_2$
1 1 1 0 0 1 0 ··· 1 1
Diff.
1 0 1 0 1 0 0 1 0
HD=$\sum_{i=0}^{n-1} x_i \oplus y_i$

Figure 1-5 Example illustration of hamming distance calculation.

Hamming Distance (HD) is a commonly used index to represent the number of bitwise differences between two binary vectors. As illustrated in *Figure 1-5*, for two binary vectors x and y both with the same length of N, the bitwise differences are checked by comparing x_i and y_i, $\forall i \in [1, N]$. By comparing x and y for all indices, one can define another binary vector d with length N, where $d_i = 1$ if $x_i \neq y_i$ and $d_i = 0$ if $x_i = y_i$. Following this definition, it turns out that d_i is equivalent to $x_i \oplus y_i$, where $\oplus$ is the binary exclusive-or (XOR) operator. Consequently, Hamming Distance can be defined as below, and the normalized Hamming Distance is defined as $HD_{norm}\,(x) = HD(x)/N$.

$$\mathrm{HD}(x) = \sum_{i=1}^{N} x_i \oplus y_i \text{ (Hamming Distance)}$$

1.3.2. Uniqueness

Uniqueness is the most fundamental property to enable PUF-based security applications. A PUF that is embedded on a chip consists of self-derivative secret information, which can act as a chip “fingerprint.” The definition for the uniqueness of PUFs and the subsequent chip fingerprints can be understood as the probability of finding an identical fingerprint in different chips. A PUF that has an outcome of N bit can meet the requirements for ideal uniqueness if the probability of finding another PUF chip that has an identical N bit outcome is the minimum value, which is equal to 2^{-N}.

For uniqueness evaluation, the inter-chip (or inter-PUF) Hamming Distance is the most common index used for uniqueness characterization and comparison. If a PUF design can provide implemented PUFs the ideal uniqueness, the resulting inter-chip Hamming Distance will follow a binomial distribution with a mean value equal to $\frac{N}{2}$ and standard deviation equal to $\frac{\sqrt{N}}{2}$.

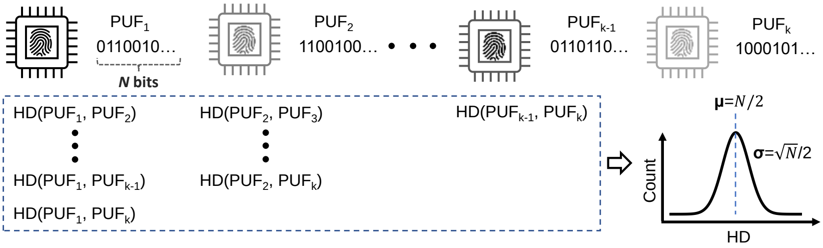


Figure 1-6 Example on how to evaluate the uniqueness of a PUF using the hamming distance distribution.

The typical procedure for a given set of PUF chips subjected to evaluation is first to compute the inter-chip Hamming Distance for all possible combinations, as illustrated in *Figure 1-6*. Then, check if the resulting Hamming Distance follows the ideal binomial distribution. If the distribution shows a noticeable deviation from the ideal case, it can be concluded that uniqueness is not ideal. On the other hand, if the resulting Hamming Distance is distributed close to the ideal case, there is high confidence that the PUF has good uniqueness.

1.3.3. Randomness

Randomness is another frequently considered PUF property that defines the entropy quality of the generated PUF secret.

Randomness is closely related to uniqueness, as good randomness is given by nature if each PUF provides a unique data pattern.

When evaluating randomness, the scope is usually on individual chips rather than comparing the data obtained from multiple chips. Even though good uniqueness can imply good randomness, performing randomness evaluation is still meaningful because the number of PUFs subjected to a thorough evaluation is usually limited. If the number is small, the statistical error for uniqueness evaluation can be significant, and hence testing the randomness of individual PUFs can be helpful under such circumstances.

Unlike uniqueness evaluation, which commonly uses inter-chip Hamming Distance, randomness evaluation is less standardized. In practice, there are three popular approaches, namely Hamming Weight checks, bit-wise correlation checks, and standard statistical tests (AIS31 [8], NIST SP800-22 [9]). Most of these methods are jointly applied for randomness evaluation to provide better confidence in the result.

In most cases, the randomness evaluation procedure starts with a check of the Hamming Weight, since it is the simplest index of all. Hamming Weight shows the ratio of “0”s and “1”s within a given set of PUF-bits, which can be a good index to check if the probability of having a “0” and a “1” is both equal to the ideal case of 0.5. If this equal probability assumption holds, the resulting Hamming Weight will be very close to 0.5. In other words, if the resulting Hamming Weight is far from the ideal value of 0.5, it is very likely that the equal probability assumption does not hold and therefore the PUF does not have ideal randomness. Once the Hamming Weight check does not fail, one can proceed to more sophisticated checks such as performing statistical tests.

1.3.4. Robustness

When a PUF is used in security applications, either for key generation or authentication, the output value shall not change over time or due to environmental fluctuations. Robustness defines the PUFs ability to produce reliable results when being continually queried, no matter how the environment changes within a given boundary. As illustrated in Figure 1-7, the PUF is queried at different times, and there are bit-errors found in some of the queries, indicating that there are concerns about the robustness of this PUF.

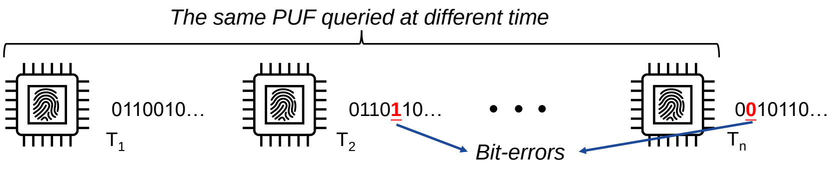


Figure 1-7 Example illustration on the robustness and bit-errors of PUFs.

The most common index for checking the robustness is the bit-error-rate (BER), which counts the number of error bits over the number of evaluated PUF-bits. For instance, if a PUF-bit is evaluated 100 times, and five errors are found, the BER can be computed by five divided by 100, which equals 5%. In the ideal case, the BER should be zero under all possible operating conditions.

When considering the effect of the operating conditions, it should be noted that a PUF can be affected by internal and external factors. Internal factors include noise, crosstalk, clock instability, and more; while external factors include supply voltage, ambient temperature, electromagnetic interferences, radiation, etc. For robustness evaluation, one should first define the range of operating conditions, which can vary for different applications. For a PUF with ideal robustness within the pre-defined range, it should not show any error when being evaluated under any combination of voltage, temperature, etc.

1.3.5. Reliability

While the robustness of a PUF is necessary amid environmental changes, reliability is required against long-term aging effects. Specifically, reliability is related to the effects that are not recoverable under regular circuit operations, such as reset or restart, and such effects will not vanish by themselves within a short period. Aging effects usually accumulate gradually within electronic devices, taking as many as several years to become apparent.

Of several known mechanisms that cause aging effects on transistors and interconnects, the most researched ones are the Negative/Positive Biased-Temperature Instability (nBTI/pBTI) [10], hot-carrier injection (HCI) [11], time-dependent dielectric breakdown (TDDB) [12], and electromigration [13]. These aging effects can cause the degradation of device quality or further create permanent damages that destroy the correct circuit functionality. As a consequence, these effects are usually considered detrimental and need to be avoided in most circuits, including PUFs.

Aging effects caused by circuit degradation are also considered a time-dependent variability, which is in contrast to time-zero variability, due to imperfect processing steps. An effective PUF relies on time-zero variability, yet the earliest silicon PUFs have been vulnerable to overwriting given their added time-dependent variability, which risks exposing PUF-based secret keys or chip identity.

Regarding reliability evaluation, a PUF with ideal reliability must satisfy two requirements. First, the generated data should be identical before and after aging. Second, the robustness of this PUF is not affected by aging effects; i.e., if the BER is 0 before aging, the BER will remain 0 after aging. To observe these long-term effects, the testing procedure requires accelerations, which are the same as other accelerated reliability tests applied for all kinds of circuits.

1.3.6. Unclonability

By definition, a physically unclonable function, a PUF, cannot be cloned physically. For PUFs implemented on chips, a physical clone of a PUF is technically impossible since there is no production process for a chip that allows this to happen. In other words, every chip that is made is unique. Moreover, if cloning were possible, assuming there is no fabrication process variation, a silicon PUF would not exist. While physical unclonability is a result of nature, another type of unclonability, namely mathematical unclonability, should also be considered.

A PUF with mathematical unclonability should be impossible to describe using a mathematical model accurately. Mathematical unclonability can be therefore referred to as imperviousness against modeling attacks, such as machine learning attacks. *Figure 1-8* shows the difference between physical and mathematical unclonability.

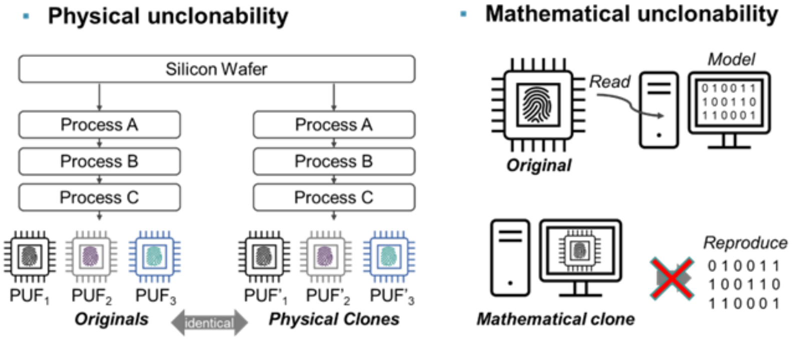


Figure 1-8 Types of Unclonability.

For PUFs that have an enormous number of outcomes that are nearly impossible to be fully queried; e.g., a PUF with 2^{64} challenge-response pairs (CRPs), mathematical unclonability is evaluated by checking how many known challenge-response pairs are needed to predict the unknown pairs with high accuracy. For example, a PUF would be considered to have poor mathematical unclonability if it can be accurately predicted or mimicked: e.g., a machine learning tool that achieves a 90% success rate after only

1000 challenge-response pairs would demonstrate poor mathematical unclonability in that PUF.

This term is, however, not meaningful for all kinds of PUFs. For a PUF with a limited number of outcomes, e.g., a PUF that can output 1024-bits of data, an adversary is strictly forbidden from obtaining the prior knowledge required to mathematically clone a PUF. In such cases, it would be more meaningful to check if the PUF data is safe from leaking out of the chip, other than considering its mathematical unclonability.

1.3.7. Tamper Resistance

In addition to the quality of generated chip fingerprints, protecting these important secrets from prying eyes is an equally important consideration. Since a HRoT provides a foundation for secure hardware, it should be resilient against hardware tampering, including information theft, manipulation, or forging. These attacks are typically physically-oriented, requiring the chip to be disassembled, polished, de-layered, or re-wired. Hence, these techniques are sometimes categorized as physical attacks. With a PUF as an anchor for an HRoT, it is also essential to consider its resistance against tampering.

1.3.8. Tamper Evidence

While providing protection against all possible types of tampering is almost impossible, detection of tampering is the bottom line for assurance against physical attacks. Tamper evidence is, therefore, one of the desired properties, the ability of a PUF to detect tampering and provide an alarm to the system for further actions. A PUF should respond differently if the chip is tampered with, or after the tampering event, to fulfill this requirement.

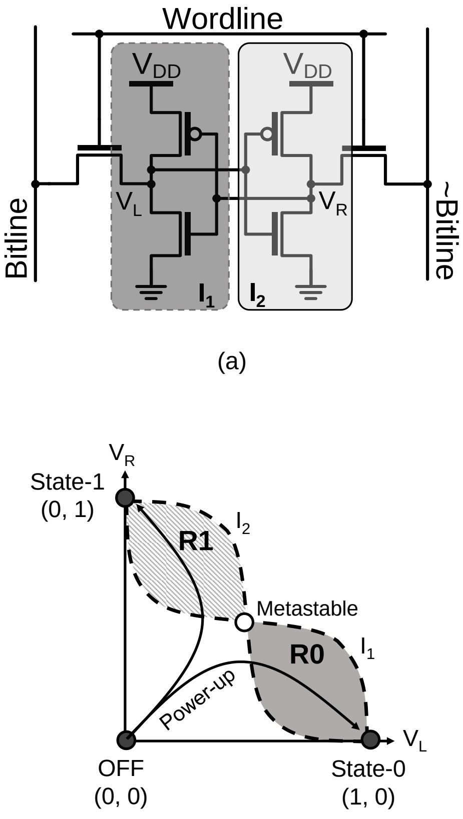
1.4. PUF Implementations

Due to increasing research into hardware security, many PUF designs have been created, and new designs continue to be proposed across different research domains. This section will discuss a few well-known PUFs, including SRAM PUF, Inverter PUF, and Arbiter PUF. The design concepts and operating principles of these PUFs will be described in details, and practical examples will also be provided to help readers have a better understanding of PUFs.

1.4.1. SRAM PUF

SRAM PUFs were proposed during the early stages of PUF development, and they have become a popular PUF design since that time. A typical SRAM PUF consists of an array of static random-access memory (SRAM) cells with a regular cell structure. In a conventional CMOS (complementary metal-oxide-semiconductor) technology, an SRAM cell typically has the structure shown in *Figure 1-9 (a)*.

In an SRAM cell, there are two inverters in a cross-coupled structure, forming a positive feedback loop that can be used to store data. As V_L is the input of the inverter I_2 and the output of the inverter I_1; V_R is the input of the inverter I_1 and the output of the inverter I_2; the relation of V_L and V_R must follow the input-output relation of both I_1 and I_2.



(a)

(b)

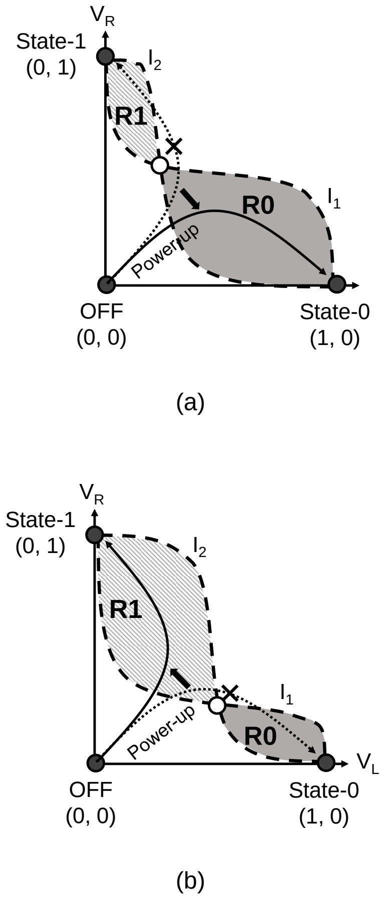
Figure 1-9 (a) Conventional SRAM cell consisting of six CMOS transistors. (b) Butterfly transfer curve of an ideally symmetric SRAM cell.

As illustrated in *Figure 1-9 (b),* there are three pairs of V_L and V_R that satisfy this criterion, resulting in two stable states and one metastable state of the SRAM cell. When being used as a memory cell, the data bit is stored as the voltages at the internal nodes V_L and V_R, the stored bit is "1" when $V_L = 0$ and $V_R = V_{DD}$ (state-1); conversely, the data is "0" when $V_L= V_{DD}$ and $V_R = 0$ (state-0). It should be noted that even if the two input-output relations could both hold in the metastable state, the SRAM cell cannot stay in this state for a long time because any tiny fluctuation in V_L or V_R can break this equivalence and eventually make the SRAM cell end up at either state-1 or state-0.

In order to understand how an SRAM PUF is operated, we must first study its power-up behavior. As illustrated in *Figure 1-9,* the SRAM is at the OFF state when it is not powered; i.e., both nodes are at zero voltage. Once the SRAM is powered-up, the voltage of the two internal nodes will rise as the supply voltage increases. Depending on the impact of the noise on two different nodes, the power-up curve will deviate for other cases, as shown in *Figure 1-9 (b).* Finally, the SRAM cell will end up in either state-0 or state-1, depending on whether the trajectory enters R0 or R1. The resulting state is called the power-up state of an SRAM.

In a conventional SRAM cell, the two inverters are designed to have the same dimension, resulting in identical pull-up and pull-down strengths. The SRAM cells, after fabrication, however, will not have identical inverters due to the mismatches caused by process variations. Consequently, the two inverters will have different pull-up and pull-down strengths, resulting in skewed voltage transfer curves as the examples illustrated in *Figure 1-10*.

For the case in *Figure 1-10 (a)*, the NMOS (n-type metal-oxide-semiconductor transistor) of I_1 is stronger than its PMOS (p-type metal-oxide-semiconductor transistor), resulting in a left-skewed transfer curve, which makes the SRAM power-up state end up at state-0 for the two initial trajectories. On the other hand, if the NMOS of I_2 is stronger, resulting in the transfer curve in *Figure 1-10 (b),* the power-up state of the SRAM will be state-1 instead.



(a)

(b)

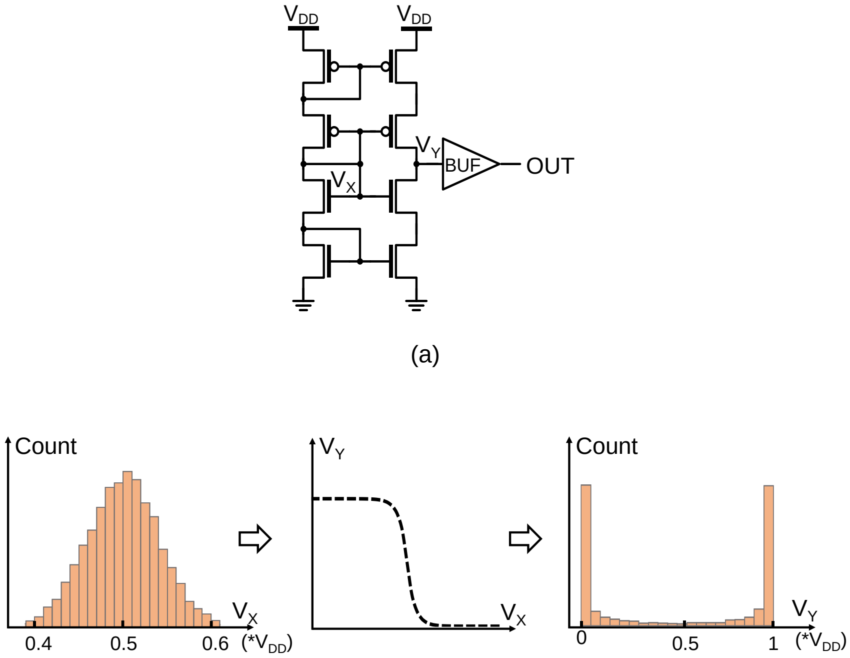
Figure 1-10 Illustration of how the skew of transfer curve results in different power-up states. (a) Inverter I2 skewed, powered-up to state-0. (b) Inverter I1 skewed, powered-up to state-1.

It should be noted that these two examples have only considered the process variation on a single transistor, while in an actual case, the resulting transfer curves will be determined by a combination of variations in all the transistors. In summary, for any fabricated SRAM cell, its transfer curves may be skewed in an indeterministic fashion, which will give a preference for the power-up state to end up at either state-0 or state-1. Consequently, an SRAM PUF array can generate a set of binary data where each bit is determined by the power-up state of an SRAM cell.

1.4.2. Inverter PUF

While SRAM PUFs and other PUFs based on a similar concept rely on a positive feedback loop to generate two different output states, these bi-stable circuit cells are, at times, highly sensitive to noise when the circuit is operating close to its metastable point. Another type of PUF has been proposed to mitigate the metastable behavior that does not require a positive feedback loop to derive its output state. These are called monostable PUFs.

One popular approach is the inverter PUF proposed in [14]. The operating principle of this circuit is to amplify a small voltage shift induced by process variations. While the circuit is designed to achieve a high amplification gain, it can amplify the small mismatch to a clearly distinguishable logic “0” and “1” without incorporating a positive feedback scheme. With the core structure shown in *Figure 1-11 (a),* the input voltage of the high-gain inverter stage is generated using a replica-biasing scheme. The bias voltage V_X is designed to be close to $V_{DD}/2$, but the resulting V_X will be shifted because of the imbalance between NMOS transistors and PMOS transistors, as illustrated in *Figure 1-11 (b).*



(a)

(b)

Figure 1-11 (a) Circuit schematic. (b) Operating concept of the inverter PUF.

The small signal will be amplified by the inverter with a high gain, i.e., having a very steep slope in its voltage transfer curve. The output V_Y will be close to V_{DD} or 0V, depending on how V_X is shifted. Consequently, the input voltage distribution with a Gaussian shape will be transformed into a binary-like output distribution. The input-output relation is determined by a single voltage transfer curve, resulting in only one stable state for a particular PUF-cell, and the inverter PUF is hence categorized as a monostable PUF.

Since there is no metastable state in such designs, this circuit does not exhibit a noise-sensitive period during power-up, and therefore, the native stability of such PUFs is in general better than bi-stable PUFs. However, given the fact that the gain of the inverter stage cannot be infinitely high, there will always be some PUF-cells that do not have an output voltage close to V_{DD} or 0V if their bias voltage is too close to V_{DD}/2. Moreover, a drawback of this type of PUF is that the data can be affected by transient fluctuations during its operation, potentially causing data to flip from one read cycle to another. The data in a bi-stable PUF will remain unchanged until the power is turned off.

1.4.3. Arbiter PUF

Arbiter PUF [15] is one of the most popular implementations of a strong PUF, and it has been widely discussed in many publications. A strong PUF has a number of challenge-response pairs (CRPs) exponentially dependent on the number of its circuit elements. SRAM PUFs and inverter PUFs are, on the other hand, categorized as weak PUFs, since the number of their output bits is linearly dependent on the number of its circuit elements.

As shown in *Figure 1-12*, the circuit core of an arbiter PUF consists of cascading delay stages formed by configurable delay elements. Within each delay element constructed by two multiplexers, the two different sets of delay paths can be selected by changing the control bit. An arbiter placed after the last delay element will determine whether the signal on the upper path or the lower path arrives first, resulting in an output of logic “0” or “1” according to their order.

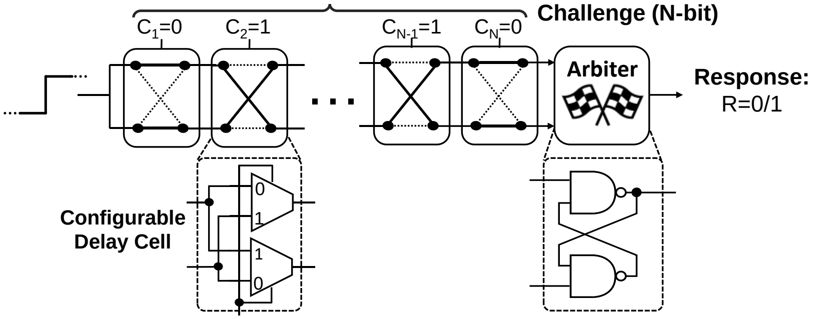


Figure 1-12 Schematic of a basic arbiter PUF.

Depending on the control bit provided to a delay element, the timing signals can stay either on the straight paths or on the crossed paths. Since the delay of each path segment is affected by process variations, providing a set of different control bits; i.e., a different challenge, the summed-up delay of the upper and lower path may change and result in a different output bit, or a different response.

For an arbiter PUF of N stages, the challenge has N-bits, and there are 2^N possible challenges. Since each challenge may result in a different response, there are therefore 2^N CRPs, which meet the criteria for a strong PUF. However, an arbiter PUF is not seen as a good example of a strong PUF. Its response is determined by a linear combination of delays, which causes strong correlations that make the CRPs highly predictable by modeling techniques such as machine learning [16]. Consequently, while an arbiter PUF has a sufficiently large number of CRPs to be categorized as a strong PUF, it does not provide strong security strength when used in real applications.

1.4.4. Limitations of Conventional PUFs

Three popular PUF implementations are introduced in this section. These PUFs all suffer from one major drawback, which is the instability of SRAM during power-up. There are transient fluctuations when operating an inverter PUF and vulnerabilities caused by modeling attacks on an arbiter PUF. In addition to modeling attack vulnerabilities, an arbiter PUF also suffers from poor stability, because the overall delay difference may be minimal.

In summary, for most of the conventional PUF implementations, the inability to generate stable PUF outputs is a significant issue because these PUFs do not have sufficiently good robustness and reliability

1.5. Conclusion

This chapter provided an overview of the PUFs implemented in integrated circuits (ICs). PUFs with different design concepts may support generic security applications, including cryptographic key generation and entity authentication. PUFs on different chips need to have unique responses, and the derived PUF data must be reliable to maintain good security functions. We have also shown that for conventional PUF implementations, it is difficult to achieve the required robustness and reliability. These limitations lead to the main topic of the next chapter: how to design a PUF that is highly robust and reliable.

2

A description of the essential PUF properties, which include robustness and reliability. It demonstrates the importance of having a robust and reliable PUF in any Hardware Root of Trust solution. Next is a description of several improvement techniques, with an analysis of the capabilities and limitations of improving the quality of conventional PUFs. The chapter closes with the benefits of an intrinsically reliable PUF and a few exemplary PUF implementations.

2. Making a PUF Highly Reliable

As mentioned in the previous chapter, one of the essential requirements for a PUF is to generate a consistent output of data across different times and environmental fluctuations. A PUF must have two properties to satisfy this requirement: robustness and reliability. In the rest of this book, the term “reliability” will be used in general to represent both robustness and reliability.

2.1. Why a PUF Must Be Reliable

Before looking for a highly reliable PUF, we should ask why a PUF must be reliable. There are several reasons. The priorities are keeping cryptographic keys consistent and making PUF data resilient to fault injections.

2.1.1. Consistency of Cryptographic Keys

There are two vital elements to encrypting a message. These are a cryptographic algorithm and a cryptographic key. Running the data through the encrypting function together with the key will yield an encrypted data string, or ciphertext, depending on the chosen cryptographic algorithm and secret key. To ensure that this scheme is secure, running the same message through the same encrypting function but with different keys should yield different ciphertext, no matter how unalike the keys are. This is referred as the “avalanche effect” of cryptographic operations. This effect is required to protect keys from being simulated by observing the relationship between input data and output ciphertexts.

On the other hand, the avalanche effect is also effective when decrypting ciphertext back into the original data. Let us suppose data, *D,* is encrypted into ciphertext, *C,* using a key, *K.* The receiving party of this ciphertext should decrypt it back to the data using the same key. But let us also assume there are some bit errors in the decryption key, denoted as *K'*. The decrypted message will be different from the original one, denoted as *D'*. Because of the avalanche effect, *D'* cannot be converted back to *D* even if *K'* has only a one-bit difference. This example shows how catastrophic it is for a cryptographic system to have errors in the cryptographic keys.

Because of the requirements of these keys, it is a clear prerequisite for a PUF to be reliable when used to generate keys. If a PUF needs to be queried when a system has to perform an encryption or decryption, one would expect the PUF to reproduce the same result every time. Otherwise, one result may be that data is encrypted and decrypted using different keys, rendering the whole system unreliable.

2.1.2. Resistance against Fault Injections

Fault injection is an essential security threat. An attacker can try to inject faults at specific times during cryptographic operations and use a fault model to guess the correct key value. By employing this type of fault injection, an attacker may reduce the complexity of predicting a key to a significantly lower level than a brute-force attack. While keys are subject to this type of attack, the PUFs that are used to derive these keys may also be targets of fault injection. Suppose the PUF is sensitive to environmental fluctuations. In that case, an attacker can easily inject faults into the PUF and the derived keys through changes such as power supply glitches or increased ambient temperature. Consequently, PUFs need to be highly reliable to reduce the possibility of successful fault injection attempts.

2.2. Reliability Improvement Techniques

Most PUFs are not intrinsically reliable as generated PUF data, without processing steps, are not consistent with different queries. Therefore, PUFs need to be accompanied by improvement techniques to ensure reliable operation in a cryptographic system.

2.2.1. Helper Data Algorithms

Because processing can improve reliability, one straightforward solution is incorporating error correction codes (ECCs), with a mathematically proven ability to correct errors. Several methods are proposed to include ECC algorithms in PUF-based security applications. These algorithms are often called helper data algorithms. The central concept here is to enroll PUF-data at the first, or a selected query, as a golden reference. It then aims to reconstruct the PUF data through other queries back to this golden reference. The helper data is a specifically encoded bit-sequence incorporating one or more ECC algorithms, containing information that can help reconstruct PUF data. The helper data should be kept in reliable storage, typically a nonvolatile memory block, and it cannot contain any information that could be used by an adversary to obtain real PUF data without directly accessing the PUF.

An example of a helper data algorithm used to derive a key using an SRAM PUF is illustrated in *Figure 2-1*. At the enrollment phase, this algorithm will fetch a random number (RN) and encode it using a selected ECC algorithm, e.g., a 255-bit Bose–Chaudhuri–Hocquenghem (BCH) code [17].

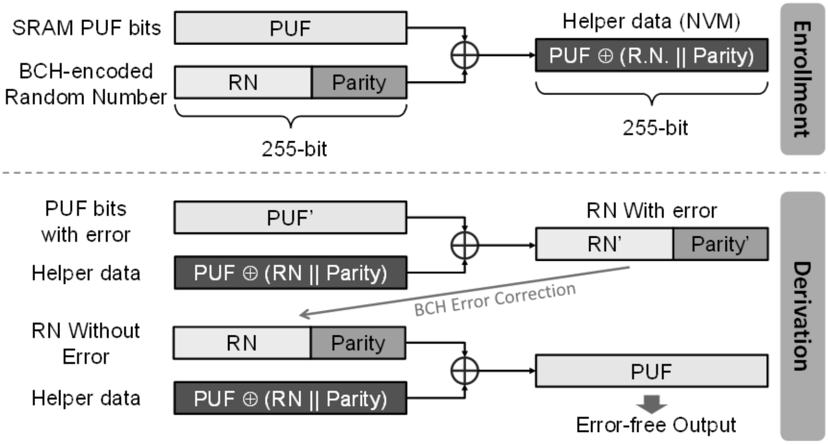


Figure 2-1 Example of reconstructing PUF using ECC and helper data.

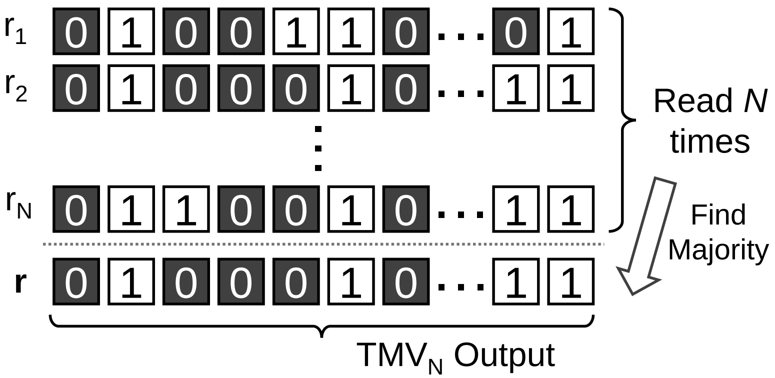
The encoded bits will consist of the original RN and some parity bits, denoted as *RN||Parity* in this example. The procedure continues by querying the PUF circuit to obtain the reference PUF data, designated as *PUF* in this example. The helper data is then received by performing a bitwise XOR operation on two-bit sequences. The result is denoted as *PUF* ⊕ *(RN||Parity)*. The enrollment step ends with storing the resulting helper data in an NVM block.

The PUF is first queried in the reconstruction phase to obtain a set of PUF-bits different from the golden reference due to run-time errors. This queried result is denoted as *PUF’* in this example. The next step is to perform bitwise XOR on PUF and the helper data. While *PUF’* is the original *PUF* with some errors on top, it can be written as $PUF'=PUF\oplus e$, where *e* is the bit errors in the queried PUF-bits. After the XOR operation, the result can be written as: $PUF'\oplus PUF\oplus(RN||Parity)=\ (RN||Parity)\oplus e=(RN'||Parity')$, which turns out to be the BCH-encoded random number with errors on top of it.

Since the error correction capability is typically designed to exceed the maximum number of possible errors, the *RN’||Parity’* bit sequence can be corrected back to the original *RN||Parity* bit sequence without error. Afterward, the original PUF data can be derived by performing an XOR operation on *RN||Parity* and the helper data, which can be used to create a secret key later on. The advantage of using ECC algorithms is that the error-correcting capability is mathematically proven. Hence, it is possible to choose a scheme that provides error-free outcomes if the maximum number of errors is known; i.e., the worst case.

2.2.2. Temporal Majority Voting

Conceptually, performing temporal majority voting (TMV) is to read a PUF multiple times and choose the output with the most occurrences as the final result. As illustrated in *Figure 2-2*, a PUF with some unstable bits is evaluated *N* times for each PUF-bit. The bit which occurs more than N/2 times is chosen to construct the final output. By probability theory, having more incorrect readouts than correct ones will have a lower probability than the opposite case. If the number of reads increases, this probability decreases with an exponential dependence.



r_1 0 1 0 0 1 1 0 ··· 0 1

r_2 0 1 0 0 0 1 0 ··· 1 1

⋮

r_N 0 1 1 0 0 1 0 ··· 1 1

r 0 1 0 0 0 1 0 ··· 1 1

Figure 2-2 Illustration of the TMV algorithm, where r is based on example N = 3.

To provide a quantitative analysis, we first consider a PUF-cell that has an error probability ϵ; i.e., the probability of reading an incorrect PUF-bit from this cell. Each readout event can define a random variable X with an outcome $x \in \{0, 1\}$, where 0 represents the case where the correct PUF-bit is read and 1 represents the case where an error bit is read; i.e., $\Pr(X = 1) = \epsilon$.

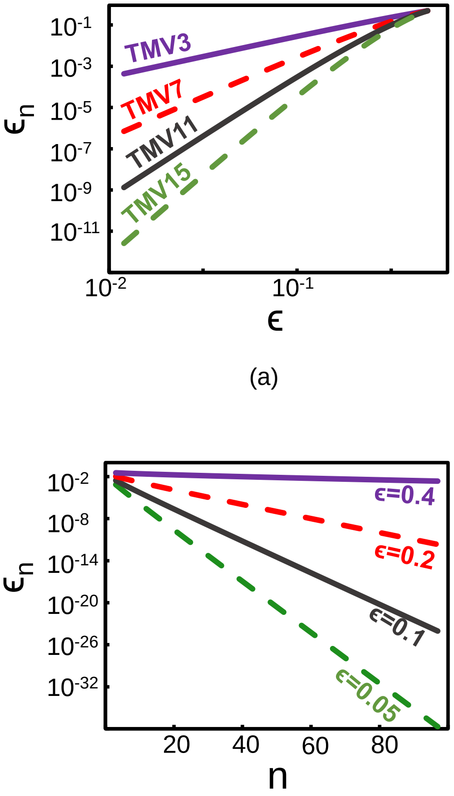
After performing a TMV algorithm of $N = n$, the number of total errors in n readouts can be defined as another random variable $Y = \sum_{i=1}^{n} X_i$, where X_i is a sample of X, representing readout events that are independent and identically distributed (i.i.d.).

Following these assumptions, the error probability after processing by TMV of $N = n$, denoted as ϵ_n, can be derived as $\epsilon_n = \Pr\left(Y > \frac{n}{2}\right)$. By definition, the random variable X is a Bernoulli trial with probability ϵ_n, and hence Y is a binomial random variable. Deriving from the probability mass function of a binomial distribution, the resulting error probability ϵ_n can be computed using the following equation.

$$\epsilon_n = \sum_{k=\overline{n/2}}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$

In this equation, $\overline{n/2}$ is the first integer that is greater than *n*/2, while *n* is usually an odd number in a TMV scheme. *Figure 2-3* shows how the resulting error probability ϵ_n scales with the number of reads in TMV, n, and the error probability of the PUF-cell, ϵ. As shown in *Figure 2-3 (a),* ϵ_n decreases rapidly with ϵ, and all the curves converge at $\epsilon = 0.5$, showing that the TMV works better with lower error probability and fails when error probability reaches the maximum value of 0.5.

One the other hand, *Figure 2-3 (b)* demonstrates that ϵ_n decreases exponentially with the number of reads in the TMV scheme. If the ϵ is high, the resulting ϵ_n reduces with a less steep slope, indicating that TMV does not have good efficiency when the original error probability is high.



(a)

(b)

Figure 2-3 TMV output error probability with respect to (a) Input error probability. (b) Number of repeated reads.

2.2.3. Dark-bit Masking

The dark-bit masking scheme is performed by first identifying the unstable PUF-cells, i.e., cells that produce different output bits at different read events, and then marking these bits as dark bits. When reading out the PUF data, these marked dark bits will be excluded from the final output, as illustrated in *Figure 2-4*. In this example, the bits that have non-zero BER (e.g., 5%, 3%, 11%, …) are considered dark-bits, and will be masked out in the final output. By applying dark-bit masking, the unstable PUF-cells will no longer contribute errors to the final PUF output, and, ideally, the resulting data will be completely stable.

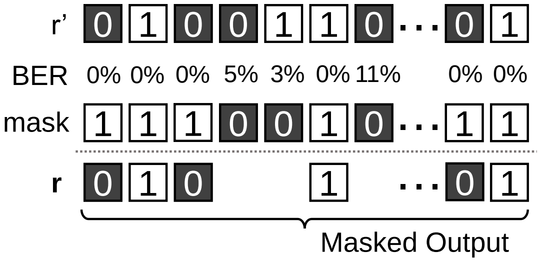


Figure 2-4 Illustration of the dark-bit masking algorithm.

Dark-bit masking can, in theory, eliminate all possible errors if all of the unstable PUF-cells can be correctly identified, but it has several practical limitations. First, identifying dark bits is challenging, particularly for PUF-cells that produce errors with very low probability. For example, if a PUF-cell produces errors with a probability of 0.01, it can easily take more than 100 reads to observe the first error, making the identification procedure quite time-consuming. On the other hand, limiting the number of reads by time consumption may result in unidentified dark bits that can degrade the resulting stability.

A dark-bit masking scheme that uses the data remanence effect in SRAM-based PUF-cells to identify unstable cells was proposed in [18], where the unstable PUF-cells are accurately identified, and it results in fully stable PUF data. Given the solid experimental results, this method is no doubt a practical approach to making PUF output stable, but it still has another disadvantage.

The dark-bit map is required as an auxiliary input while reading PUF data, and hence it must be stored in an embedded or external nonvolatile memory block. This requirement of the dark-bit masking scheme requires additional hardware resources, which are similar to helper data algorithms, making it less practical to replace the conventional helper data algorithm with a dark-bit masking scheme.

One may also consider generating the dark-bit mask on the fly to eliminate the storage requirement, i.e., identifying unstable bits before reading PUF data every time. However, this solution is infeasible in practice, as always identifying the same set of unstable bit cells is impossible. If the remaining bit cells vary at each read event, it will also introduce instability in the final PUF data, making it an impractical solution.

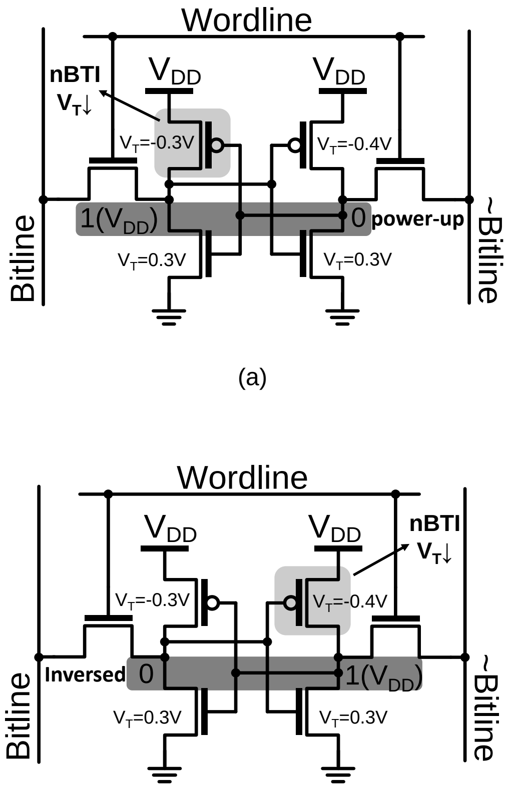
2.2.4. TMV and Dark bit Masking

As discussed, TMV works better for reducing errors in cells that have small error probabilities, while dark-bit masking is more efficient when identifying PUF-cells that are more unstable. One may think of a solution that combines these two methods; i.e., first applying dark bit masking to screen out PUF-cells with high error probability, and then performing TMV when reading the remaining PUF-bits for further error reduction. In prior PUF designs that have included stabilization techniques, these two methods are both applied in most cases.

2.2.5. Burn-in Enhancement

The final stabilization technique discussed in this section is based on the long-term degradation of semiconductor devices, which will normally result in declining circuit performance or even fatal failures. The concept of burn-in is to intentionally apply controlled aging conditions to the PUF circuit, which can be considered as introducing time-dependent device variations on top of pre-existing device variations induced by the chip manufacturing process. Once time-dependent device variation is controlled to increase in the intended direction, it can multiply mismatches within a PUF-cell and hence generate PUF data more consistently.

One famous example is exploiting the BTI (Biased-Temperature Instability) degradation in SRAM PUFs. As discussed in [19], if an SRAM cell keeps storing its power-up state, the transistors will exhibit the stress condition as depicted in *Figure 2-5 (a).* In this case, the highlighted PMOS, which has a higher threshold voltage (VT), will see the nBTI stress, and its VT will decrease gradually if the cell remains under the same stress condition. The decreased VT harms the stability of this PUF-cell since the mismatch of the inverter pair is decreased, making the SRAM cell more sensitive to noise while being powered up. On the other hand, the solution in [19] was proposed to counter BTI effects, and it can even exploit such degradation to enhance the stability of an SRAM PUF. Instead of letting the SRAM cell keep its power-up state, the method will program each SRAM cell to a state opposite from its power-up value. Considering the same SRAM example, *Figure 2-5 (b)* shows the updated bias condition*.* In this case, the highlighted PMOS with a lower V_T is now under nBTI stress, and hence the mismatch between these two PMOS transistors will be increased, resulting in better data stability.



(a)

(b)

Figure 2-5 Simplified illustration of how nBTI degradation mechanism affects an SRAM cell when storing (a) Power-up state. (b) Inverse power-up state.

One drawback of applying this method is that it requires a long time to accomplish. To make BTI effectively improve a PUF’s stability, it can take months or years under nominal operating conditions. In practice, one can accelerate the degradation process by applying a higher voltage and temperature while performing burn-in enhancement. Even under such accelerated stress conditions, the required stress time is still on the order of hours. This additional time consumption is too much for a regular semiconductor testing flow, implying that this solution is infeasible for mass-production.

2.3. Intrinsically Reliable PUFs

As previously discussed, improving stability when reading a PUF is quite challenging, even before considering environmental fluctuation and aging effects. This indicates that a considerable effort must be paid to make PUF data highly reliable. In order to mitigate this cost of data stabilization, the aim, naturally, is for an intrinsically reliable PUF.

Today, only a few PUF implementations have proven to be highly reliable. These implementations demonstrate zero BER under multiple operating conditions, and some well-known examples are PUFs that use Quantum Tunneling behavior in gate oxides [20][21], PUFs based on Resistive Random-Access Memory (RRAM) [22], and PUFs based on contact formations [23]. The following section will first discuss the benefits of these highly reliable implementations, and then briefly introduce and compare some of these PUF examples.

2.3.1. Benefits

The most obvious advantage of a highly reliable PUF at the core of a HRoT is its elimination of the complicated error-correction and stabilization scheme. This brings several benefits from different perspectives.

First, demands on hardware resources are reduced, including the extra processing logic and NVM for storing helper data. Second, a chip design that does not have dedicated hardware for these additional processing steps must use a microprocessor to implement these algorithms, possibly creating long latency when reading PUF data. This introduces security risks because the PUF data has to be transferred outside the secure boundary of the PUF hardware. Third, not requiring helper data, also eliminates the risk of tampering resulting in fatal failures in the final PUF data. Other benefits deriving from this more straightforward approach include improvements in yield, testability, and scalability. These benefits are described in detail in Chapter 5.

2.3.2. Examples

The main principle behind designing a highly reliable PUF is to exploit the nearly permanent effects in a semiconductor device, as has been done earlier with non-volatile memory. One design uses two competing anti-fuse one-time programmable memory (OTP) cells [20], which will store different values, and this difference will continue throughout the entire device’s lifetime. By replacing anti-fuse OTP cells with RRAMs, a PUF with similar performance can be implemented, as discussed in [22].

In addition, PUF using RRAM provides reconfigurability, which, while sometimes is considered an advantage, could also bring the vulnerability of PUF data being maliciously modified. Using RRAM-based PUFs also requires additional processing steps, which are not always available in targeted foundry technology nodes, making it less flexible in terms of availability compared to an OTP-based PUF.

A PUF implementation based on contact formation is introduced in [23], which uses variability while processing contact layers to cause short or open circuits between the active and first metal layers. While the resulting PUF is highly reliable, the major drawback of this method is that it requires design-rule violations to make the contact size smaller than the one defined by the foundry. Since violating design rules might have fatal consequences and is typically not allowed in most technology platforms, implementing this PUF may only be possible in a limited number of foundries and process technologies.

Another disadvantage is that the probability of having a short or an open circuit is highly dependent on processing steps. For example, there may be more short circuits if the processing temperature is slightly increased. Even if the PUF has been pre-characterized and designed with a contact dimension that achieves an equal probability of resulting in “0” and “1”, any random variations or manual adjustments within processing steps may destroy this balance and degrade the randomness and uniqueness of fabricated PUF. As a result, this type of PUF is not a viable solution in terms of mass-production.

2.4. Conclusion

This chapter has given detailed arguments to show the need for a highly reliable PUF along with several illustrative examples. Because conventional PUFs require sophisticated data-processing schemes that are challenging and resource-consuming, it is a natural choice to select PUFs that are intrinsically reliable. This section has introduced and compared several examples of highly reliable PUFs, demonstrating that a PUF based on OTP is the best choice. More details about this type of PUF will be discussed in the next chapter.

3

How a PUF uses the Quantum Tunneling mechanism. The basic structure and operating principles are described with a detailed explanation of the Quantum Tunneling mechanism. Next, an actual Quantum Tunneling PUF circuit implemented as commercial IP, namely NeoPUF, is introduced. The detailed experimental characterization will also show that NeoPUF has state-of-the-art robustness and reliability, while still meeting all the other requirements of a PUF. Finally, the anti-tampering capability of NeoPUF is demonstrated with examples of several unsuccessful attacks on NeoPUF chips, using leading analysis tools.

3. NeoPUF Quantum Tunneling

This chapter will conceptually introduce the Quantum Tunneling mechanism in the gate oxide of MOSFETs to implement highly reliable PUFs. The chapter will then demonstrate a proven Quantum Tunneling PUF implementation and how it achieves state-of-the-art performance.

3.1. Quantum Tunneling Mechanism

Tunneling behaviors in the gate oxide layer of MOSFETs have been studied for decades because they can induce leakage current at the gate terminal of a transistor, which is called gate leakage. If a semiconductor device is constantly stressed, the level of tunneling current might increase throughout the device's lifetime and eventually reach a level that permanently changes the transistor's behavior. Such a phenomenon – usually considered detrimental to the performance of MOSFETs – is called gate oxide breakdown [12] or gate oxide rupture. Even so, this tunneling behavior has been used advantageously to implement one-time programmable memory cells and several promising PUF implementations.

3.1.1. Formation of Oxide Traps

Tunneling behavior in a gate oxide is oriented from traps within the dielectric layer in the form of dangling bonds. The occurrence of traps is typically due to bond breakages within the dielectric material or, more commonly, in the interface of dielectric layers.

These traps will form quantum wells for charged carriers, and if a carrier falls into a trap, it will require higher excitation energy to escape from this trap. In addition, because the quantum well will narrow the barrier formed by dielectric layers, the probability for a carrier to tunnel through the gate oxide will increase.

Once a voltage is applied across the gate oxide, traps will be generated within the dielectric layer with a speed proportional to the magnitude of the applied voltage. The generated traps may occur at random locations within the dielectric layer and typically, more traps will be located near the oxide interface. The speed of trap generation is also proportional to temperature. Because of that, traps will also be more likely to form in areas that are heated, for example, by leakage of current.

3.1.2. Tunneling Current and Tunneling Path

Given this trap-generating mechanism within the oxide layer, it is feasible to efficiently create a tunneling path within the gate oxide by applying high voltage stress across it. When oxide traps are generated at random locations, within a certain time, a set of traps may form a path that allows carriers to tunnel through with a much higher probability compared to the gate oxide before stress. This conduction path is depicted as the tunneling path. The level of tunneling current will increase sharply once a tunneling path is formed. It should be noted that the formation of a tunneling path requires a set of traps separated by a short distance, as illustrated in *Figure 3-1*. Because the traps are generated at random locations, the required time to form a tunneling path is indeterministic. For instance, there might be a large number of traps in an oxide that fail to form a tunneling path; contrarily, there might be a small number of traps through which a tunneling path is formed.

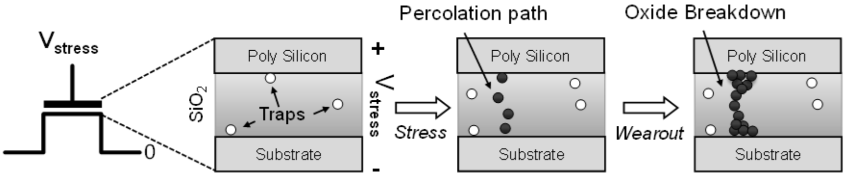


Figure 3-1 Illustration of a gate oxide breakdown event.

Once traps that can form a tunneling path occur in a gate oxide, the gate current will increase rapidly and generate heat around the tunneling path. This will further accelerate the trap-generation process, and many new traps will occur at this location. The occurrence of new traps will further increase the tunneling current and then generate more heat. These two mechanisms will form positive feedback which results in a rapid growth of the tunneling path and later on leads to a resistive-type conduction path; i.e., one with a hard gate oxide breakdown. As part of this process, one can apply a current limiter to define the maximum current level, which will stop the growth of the tunneling path and prevent a complete rupture. In other words, current can still be conducted through the tunneling path instead of being blocked. This controlled situation is called a soft breakdown event, in contrast with the aforementioned hard breakdown event. To ensure reliable circuit operations and improve physical security, using the soft breakdown mechanism is usually preferable to the hard breakdown mechanism.

3.1.3. The entropy of Quantum Tunneling

The indeterministic nature of Quantum Tunneling can be used as a PUF entropy source. One method is based on the first occurrence of a tunneling path within two MOSFETs. As illustrated in *Figure 3-2*, two MOSFETs are placed in parallel to form the core of a PUF-cell. While applying the same stress on the two transistors, traps will be generated in both transistors under the same condition. Since the required time to form a tunneling path is uncertain, there is no way to know which transistor will first have a tunneling path. As soon as this happens, the stress voltage will immediately drop or even halt through a feedback mechanism, and hence there will be only one tunneling path in a PUF-cell.

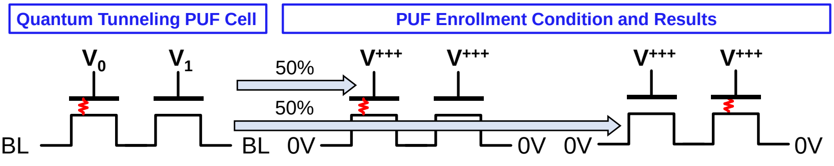


Figure 3-2 Basic Quantum Tunneling PUF-cell structure and a simplified illustration of how PUF-cells are enrolled.

Given this uncertainty, such a PUF-cell can effectively generate a random bit based on where a tunneling path occurs. For example, if the tunneling path occurs at the left transistor, it outputs a “0” bit or a “1” bit in the alternate case. The chance of a tunneling path forming on the left or right is precisely 50/50 because the two transistors are identically manufactured with the same oxide area and thickness. The time it takes to form a tunneling path at either transistor is a random variable that is identically distributed. Even though there will be process variations that introduce mismatches between the two transistors, these will only cause minor differences in the timing distributions. More importantly, the variations in these oxides are also random and cannot be used to predict which transistor will first have a tunneling path. Consequently, PUF-cells using this mechanism are proposed in [20][21] and have been qualified as having excellent properties.

3.2. Circuit Implementation

The Quantum Tunneling PUF to be introduced in this chapter is NeoPUF, owned by eMemory Technology. It was first introduced in a technical paper presented at ISSCC2018 [20] and was awarded as the best article in the Far East region that year. The bit-cell of NeoPUF is shown in *Figure 3-3*, in which the two NMOS transistors connected to the V_0 and V_1 terminals will be subjected to high-voltage stress.

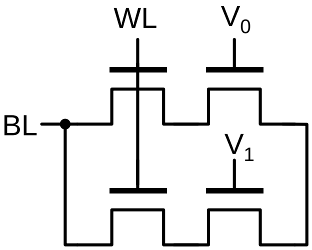


Figure 3-3 Bit-cell structure of NeoPUF.

There are two additional NMOS transistors between the bitline (BL) and the stressed transistors. These additional transistors can act as a current limiter when a tunneling path is generated, enabling a negative feedback control mechanism that have two important effects: (1) limit the stress voltage to ensure only one tunneling path is created in a PUF cell; and (2) limit the current level to prevent hard breakdown events.

3.2.1. NeoPUF Array

The entire PUF is implemented in an array structure, as shown in *Figure 3-4,* in which each cell can be controlled individually, similar to a typical memory block. Before creating the desired PUF behavior, the NeoPUF array is first subjected to an enrollment step, where the core transistor pair within each cell is stressed to form a tunneling path. After enrollment is finished, the PUF data can be read out by a sense amplifier, which will detect the current magnitude to determine the tunneling path location.

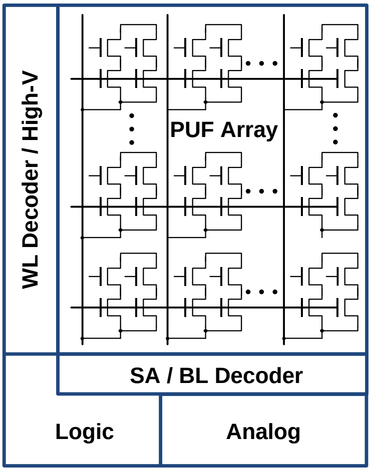


Figure 3-4 Structure of NeoPUF array.

3.2.2. Single-ended Readout Scheme

The data readout scheme used in the NeoPUF array is based on a single-ended technique, where one end of a differential sense amplifier (SA) is connected to a reference current source, and the other end is connected to a bit-line. Once a cell is accessed, and other bit-cells that are sharing the same bit-line are not, there will be either a tunneling current that is higher than the reference, or a tiny level of leakage current that is lower than the reference, flowing into the connected sense amplifier. According to the difference between the sensed current and the reference current, the SA will output a “0” or a “1” depending on the location of the tunneling path.

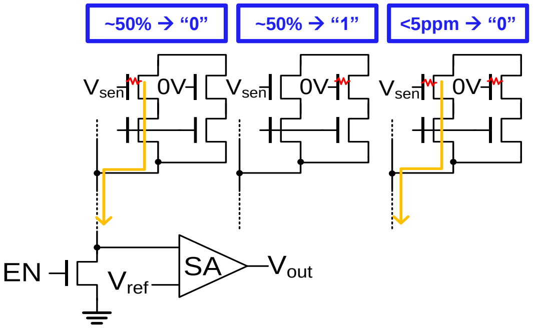


Figure 3-5 Single-ended read mechanism of NeoPUF ensuring zero bit-error even for a non-ideal bit-cell.

As illustrated in *Figure 3-5,* the sensing current is only coming from the cell on the left side, while the behavior of the right transistor is never considered. This sensing scheme has the advantage of better stability, compared to a differential sensing scheme that feeds current on both sides into a differential sense amplifier. The reason for better stability is the absence of ambiguity, even if the two transistors each have a tunneling path. In such a case, the proposed single-ended scheme will easily distinguish a “0”, while in a differential scheme, the result may be flipped between “0” and “1” if the two current levels are close to each other. Consequently, using a single-ended readout scheme can prevent possible instability caused by a non-ideal enrollment result.

It should be noted that if there are many cells that have two tunneling paths, the resulting PUF data will have noticeably more “0”s than “1”s, which is harmful to randomness and uniqueness. Fortunately, in the case of NeoPUF, the probability of having two tunneling paths within a PUF-cell is very unlikely, which, based on testing, is lower than 5ppm. As a result, applying a single-ended readout scheme in NeoPUF will significantly eliminate possible instability with only a negligible impact on the degradation of randomness.

3.3. Reliability and Robustness

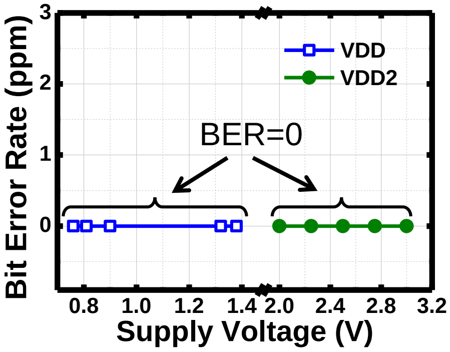
The tunneling path is considered permanent in a semiconductor device’s lifetime under nominal operating conditions. The oxide traps will remain effective unless the oxide is annealed at a very high temperature, which typically requires up to 400°C [24]. The device will not reach such extreme temperatures under normal operations. As a result, NeoPUF is highly reliable, because the location of the tunneling path within a PUF-cell can always be reliably distinguished to derive a PUF-bit. This section will focus on characterizing the reliability and robustness of NeoPUF to provide evidence that NeoPUF is an ideal and highly reliable PUF solution.

3.3.1. Ideal Data Stability across Operating Conditions

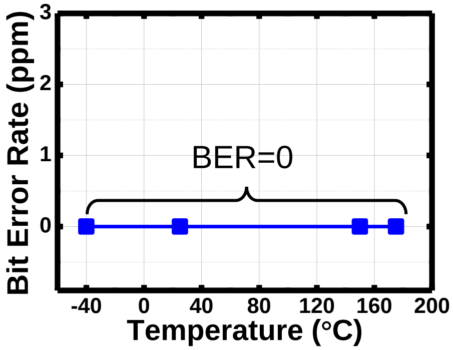
The robustness of NeoPUF is tested by collecting data from a large number of PUF readouts under multiple operating conditions. The bit-error rates across conditions are computed based on the gathered data, and the robustness of NeoPUF is proven by showing BER=0 in a wide operating range.

First, the NeoPUF test chips are tested under different supply voltages, where a conventional specification for supply variation is ±10% of the nominal value. As shown in *Figure 3-6 (a),* the tested condition is set beyond the conventional range, and the resulting BER is zero across these supply voltages, showing that NeoPUF has ideal robustness in terms of supply voltage variations.

Second, the chips are tested under different temperatures, where the tested conditions are over a wide temperature range from -40°C to 175°C. The result, as demonstrated in *Figure 3-6(b),* NeoPUF's behavior is unaffected by varying the temperature, showing a BER=0 across all tested temperatures. This result shows that NeoPUF can be robustly used in security systems that might operate in harsh environments such as automotive electronics. Given the results demonstrated in this subsection, it can be concluded that NeoPUF indeed has very good robustness and can perform well in many different environments.



(a)

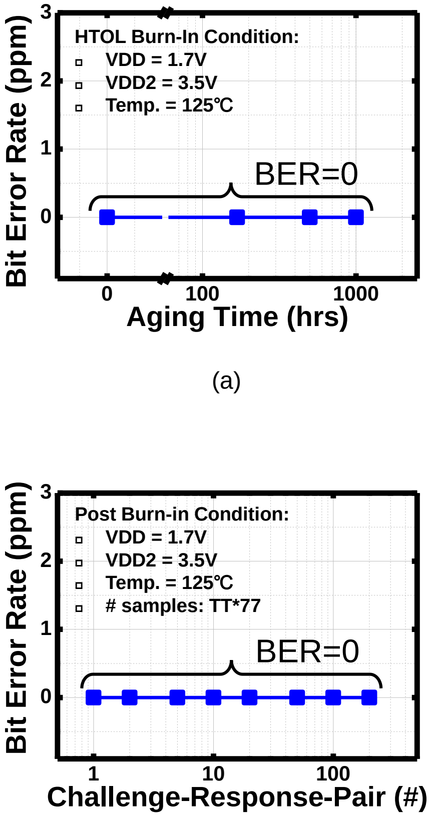


(b)

Figure 3-6 Bit-error rate of NeoPUF v.s. operating (a) Voltage. (b) Temperature.

3.3.2. Immunity Against Aging: HTOL

While robustness across different conditions has been proven, reliability; i.e., resilience against long-term aging, is also an important feature to be characterized. Since long-term effects can usually take several months or years to become noticeable, they need to be accelerated in experiments.



(a)

(b)

Figure 3-7 NeoPUF reliability assessment result using HTOL test. (a) BER along with aging time.(b) Post burn-in BER of different PUF samples.

One commonly used acceleration experiment is the HTOL (High-Temperature Operating Life) test, with the core supply voltage raised to 1.7V and the operating temperature raised to 125°C. As shown in *Figure 3-7 (a),* the BER remains zero from the beginning of the experiment to the end of 1000 hours of experiment time, equivalent to a ten-year device lifetime. Besides having zero BER in the experiment, the NeoPUF chips were continuously measured after the HTOL test, with the result that the BER remained zero, as shown in *Figure 3-7(b),* indicating that long-term aging did not degrade the chips' operation. Consequently, the reliability of NeoPUF is solidly proven.

3.4. Radiation Hardness

While NeoPUF can operate in harsh environments such as aerospace applications, resilience against radiation damage is also an important indicator of robustness. Regarding the effects caused by radiation, there are two main types of damage mechanisms for CMOS circuits, namely SEE (single event effect) and TID (total ionized dose). Radiation particles and electromagnetic waves may come from any angle and penetrate through packages, substrates, and metal layers to hit active devices. Although both types of radiation may affect all devices, devices with different structures or modes of operation may be susceptible only to SEE or TID.

3.4.1. Radiation Hardness Considerations

Single events are mainly caused by radiation particles, including neutrons, protons, electrons, and ions. If a high-energy particle penetrates a silicon substrate and strikes the junction of a transistor, it can induce a current spike. This can result in transient effects such as changing the stored value within a register under an SEU (single event upset). Additionally, a triggered latch-up effect affects CMOS circuits through the SEL (single event latch-up). The SEU, SEL, and other single-event effects are recoverable.

For example, a register can store the next input data correctly after having an SEU, and the latch-up can also be terminated after a power cycle. Since NeoPUF uses a Quantum Tunneling mechanism in gate oxide, no substrate current or bias is involved when the PUF-cell is being read. The PUF-cell is therefore unaffected by single events caused by particle strikes.

On the other hand, because NeoPUF is based on the tunneling mechanism in a gate oxide, it is more susceptible to TID damage. This can result in bond breakages in the gate oxide when a high-energy electromagnetic wave, such as a gamma ray strikes. Once a high dosage of gamma rays radiates the gate oxide, many traps may be generated and increase the level of leakage current or induce shifts in the threshold voltage.

Since the behavior of a NeoPUF-cell is insensitive to its threshold voltage, only the leakage current induced by TID may be harmful to NeoPUF. In the following experiments, the TID effect on NeoPUF is therefore examined to check whether TID-induced leakage can damage the robustness of NeoPUF.

3.4.2. TID Experiment Result

In this experiment, NeoPUF test chips are radiated by gamma rays with a total dosage of 100kGy (SiO_2), which is similar to the exposure that a device in a nuclear power plant would sustain.

This level is much higher than what a typical device in an aerospace environment might encounter throughout the entire system's lifetime. After the gamma ray exposure, the chip is again subjected to multiple readings under different supply voltages. The data after radiation exactly matches the data before radiation. This result shows that NeoPUF is immune to TID effects caused by high-energy radiation sources, making it an ideal solution when a radiation-hardened PUF-cell is needed.

3.5. Uniqueness and Randomness

While good reliability and robustness ensure that a NeoPUF can consistently derive the same key, the quality of the derived key is ensured by having suitable properties of uniqueness and randomness.

3.5.1. Near-ideal Hamming Distance Distribution

The uniqueness of a PUF is typically checked using its inter-chip Hamming Distance distribution. As discussed in section *1.3.2,* if the data bit generated by each PUF-cell is independent and has a 50/50 probability of “0”-bits and “1”-bits, the resulting Hamming Distance will follow an ideal binomial distribution, which can be determined by the number of PUF-bits.

As shown in *Figure 3-8*, the resulting Hamming Distance distribution plotted in blue squares, closely follows the ideal distribution plotted in a black line. This result has shown that NeoPUF has the required uniqueness property and can generate high-quality cryptographic keys.

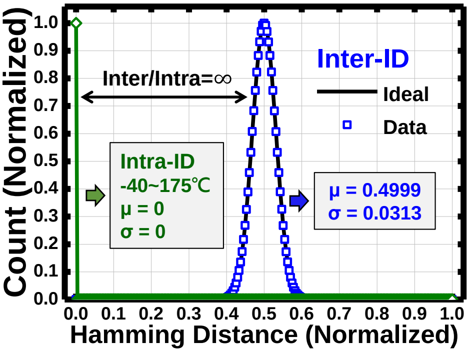
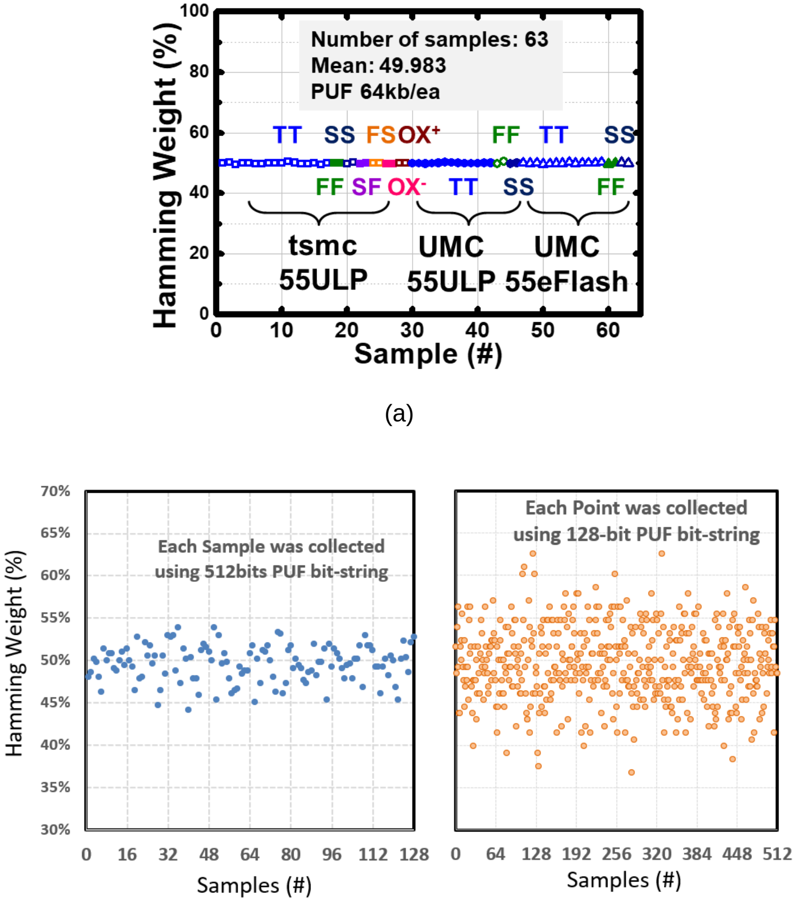


Figure 3-8 Inter-ID and Intra-ID Hamming Distance distribution of NeoPUF from 63 fabricated NeoPUF dies, in which each ID has 256 bits.

3.5.2. Near-ideal Hamming Weight across Platforms

Regarding the randomness property of PUF data, Hamming Weight is the most straightforward indicator. Each PUF-cell should have an equal probability of producing “0”-bits and “1”-bits to achieve ideal randomness, resulting in a normalized Hamming Weight close to 0.5. As shown in *Figure 3-9 (a),* the normalized Hamming Weight is very close to 0.5 across different technology nodes and process corners, indicating good randomness.

In addition, because the PUF array has 64k PUF-cells, there are concerns about the spatial dependence on randomness across the PUF array. As shown in *Figure 3-9 (b),* the Hamming Weight across word-lines and bit-lines are all close to 0.5, where the deviations are larger for the case of the Hamming Weight on each bit-line. This is due to the number of PUF-cells in each bit-line being only 128-bit. For 128-bit random sequences, around 95% of the Hamming Weight will be between 42% and 58%, which is close to the resulting statistics for Hamming Weight of the bit-line data.



(a)

(b)

Figure 3-9 Hamming Weight of each PUF (a) Chip. (b) Each PUF row/ column.

3.5.3. NIST Randomness Tests

In general, randomness cannot be experimentally proven because one can only observe the outcomes of a random variable, but not the random variable itself. Therefore, the evaluation for randomness is based on a “null hypothesis” by first assuming a random variable has ideal properties and then checking if the experiment outcome matches the assumption. If the discrepancy between theory and experiment is large, one can easily conclude that the experimental source of randomness is not ideal. On the other hand, if the experiment matches the hypothesis, one is confident that this randomness source is likely to be ideal.

Following this concept, many randomness tests have been developed to check if a bit sequence generated by a random source is truly random; i.e., each bit has a full entropy of one. These tests examine statistics from generated bit sequences and check if the statistical properties match the ideal random sequence for full entropy. Since many statistical properties can reflect randomness, multiple statistical tests are usually exploited for a more thorough examination. Test suites are a set of statistical tests that are standardized for general applications, and they are useful when determining the randomness of a bit sequence.

One of the commonly used randomness test suites is defined in NIST SP800-22 [9], which consists of statistical tests as listed in *Table 3-1.* The passing criteria for these test suites have p-values greater than 0.01. The table shows that the bit sequence generated by NeoPUF test chips has good randomness.

Table 3-1 Summary table of NIST 800-22 test results

	Test Name	**Stream Length**	**No. of Runs**	**Min. Pass (%)**	**Avg. P-value**	**Pass?**
1	Frequency	40000	75	97.33	0.4999	Pass
2	Block Frequency	40000	75	100	0.5067	Pass
3	Cumulative Sum Forward	40000	75	98.67	0.5084	Pass
4	Cumulative Sums Reverse	40000	75	98.67	0.4946	Pass
5	Runs	40000	75	100	0.5384	Pass
6	Longest Run	40000	75	100	0.4783	Pass
7	Rank	40000	75	97.33	0.4568	Pass
8	FFT	40000	75	97.33	0.5142	Pass
9	Non Overlapping Template	40000 (m=9)	75	94.67	0.5060	Pass
10	Overlapping Template	40000 (m=9)	75	100	0.4498	Pass
11	Universal	1000000	3	100	0.6428	Pass
12	Approximate Entropy	40000 (m=10)	75	100	0.4245	Pass
13	Random Excursions	1000000	3	100	0.5701	Pass
14	Random Excursions Variant	1000000	3	100	0.4801	Pass
15	Serial	40000 (m=16)	75	100	0.5387	Pass
16	Linear Complexity	1000000	3	100	0.7000	Pass

In addition, a more recently introduced test suite, NIST SP800-90 [25], can be applied to evaluate the entropy of a random bit sequence. As defined in the document, one should first have a decent theoretical understanding of the entropy source and build up a stochastic model to provide a theoretical entropy estimation. For NeoPUF, as discussed in *section 3.2,* the entropy of a NeoPUF-cell is equal to one because of the competing mechanism between the two gate oxides. Following this analysis, this NIST suite provides a set of tests that helps to verify if the experimental data matches the theoretical estimation.

Since NeoPUF is known to provide bit sequences under which every bit is IID (independent and identically distributed), the results are evaluated using the IID test set defined in this document. Results show that the NeoPUF-bits have the same statistical properties as IID random bit sequences and match the theoretical understanding. Given that the NeoPUF-bit is IID, the test suites provide a method of evaluating the entropy of the bit sequence. It turns out that the bit sequence has an estimated min-entropy equal to 0.98/bit, which is close to the ideal value of 1/bit. Consequently, NeoPUF is again proven to have near-perfect randomness.

3.6. Anti-tampering Features

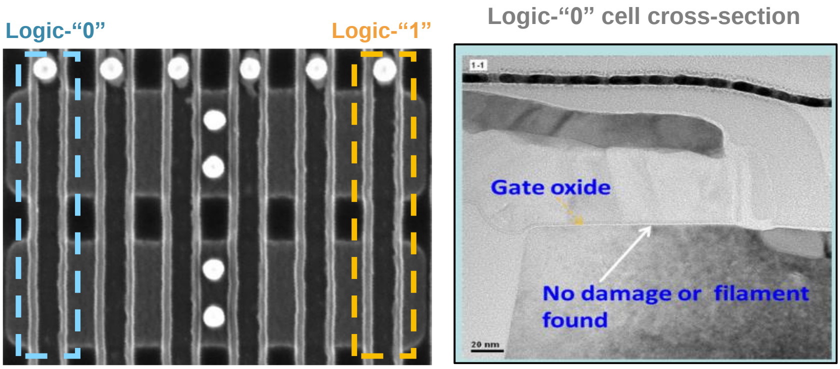
NeoPUF is well-suited for applications that require high security, in which tamper resistance is an important consideration. Any Root of Trust solution designed for security applications should be equipped with adequate anti-tampering features. The same concept applies to a PUF as well, and there are two main perspectives when considering tamper-proof PUF designs. The first is the intrinsic anti-tampering feature that comes with PUF technology. This ensures that the content of a PUF cannot be stolen or overwritten when an attacker has direct access to the PUF array. The second is the additional anti-tampering design added to peripheral circuits to protect the PUF content from being stolen through controlling and observing PUF operations.

3.6.1. Physical Security against SEM and TEM Techniques

SEM (Scanning Electron Microscopy) and TEM (Transmission Electron Microscopy) are powerful analysis techniques for semiconductor devices. While these techniques benefit researchers and manufacturers by revealing structural information supporting failure analysis, device characterization, and theory verification, they can also help malicious parties reveal secrets inside a secure device.

For conventional PUF implementations using process variations like SRAM PUF, vulnerabilities against physical inspection using SEM and TEM are generally overlooked because variations in doping profiles are too small to be observed by visual inspection. On the other hand, if a secret is stored using eFuse, which is made of metal lines that are programmed to form short or open circuits, the stored “0” and “1” values will be visible in an SEM image and, therefore, the secrets stored in an eFuse are not considered physically secure.

When considering NeoPUF technology, which also follows a similar open/short-circuit concept to distinguish bit-values, one may have concerns regarding its vulnerability against visual inspection. Fortunately, as mentioned earlier, NeoPUF-cells are based on the quantum-tunneling mechanism induced by oxide traps, and these traps are formed by atomic-level defects that cannot be individually identified. If the tunneling current is limited and the tunneling path does evolve into a resistive conduction path, the physical difference between the two gate oxides cannot be distinguished through visual inspection using either SEM or TEM.



(a) (b)

Figure 3-10 Microscopic inspection results of NeoPUF using (a) SEM. (b) TEM.

This theoretical claim is verified through the physical analysis of NeoPUF chips. As shown in *Figure 3-10 (a),* the overhead view, obtained using SEM of two NeoPUF-cells that have different logical values, does not show an observable pattern that can be exploited to obtain the PUF secret. In the TEM image shown in *Figure 3-10 (b),* the cross-section view of the gate oxide of a

NeoPUF-cell presents a tunneling path that has been formed. The TEM image shows no evidence that indicates there is a tunneling path occurring in this gate oxide. As a result, it can be concluded that NeoPUF is not vulnerable to microscopic inspections of its physical structure.

3.6.2. InGaAs Results

The previously discussed inspection technique requires destructive sample preparation and can work when the chip is powered off. There are other techniques that are less invasive and can be exploited to attack a chip while it is performing regular operations. One proven method is using the photoemission microscopy (PEM) technique, in which a photodetector is used to sense photoemission events induced by circuit operations within a chip. This technique is very useful when analyzing failures within a chip by discovering faulty points such as leakage spots, stuck-at-zero/one faults, and irregular switching events.

In advanced semiconductors, the operating voltage of core devices is typically below one volt, and the energy of the emitted photons decreases accordingly, making the light spectrum move to near-infrared, which is less likely to be absorbed by the silicon substrate. These emissions can be sensed by photodetectors made of InGaAs (Indium-Gallium-Arsenide) material. A PEM tool equipped with an InGaAs camera is, therefore, a natural choice for a top-tier attacker who wants to analyze the contents of a PUF circuit.

Using InGaAs imaging analysis of NeoPUF, an attacker would want to detect differences in photoemissions in the PUF-cells, which store different values. In the experiment, the PUF chip has its substrate facing up, allowing photons emitted from the substrate to be collected by the InGaAs camera. The PUF chip is connected to a logic periphery implemented on an FPGA, which processes user commands and controls the PUF circuit accordingly. Testing with this peripheral block provides a more realistic circuit characteristic because the PUF will also be equipped with and controlled by a logic wrapper in a commercial chip. A PUF array itself can only respond passively according to the given input conditions. Having a logic wrapper allows a user to define circuit parameters such as response time and access privileges.

Since the InGaAs imaging analysis requires continuous access to the PUF-cells, in order to keep circuits active and collect a sufficient number of photons for logic distinguishment, the response time to a read command is set to be at least 13µs, limiting the maximum PUF read speed. The experiment result of the InGaAs imaging analysis is shown in *Figure 3-11 (a).*

There are no hotspots found that can be used to distinguish the logic values generated by PUF-cells. To verify that the InGaAs image has been correctly captured, a zoomed-out figure is shown in *Figure 3-11 (b),* where clear hotspots are found at the location of the bandgap reference, indicating that the InGaAs imaging technique is capable of sensing photoemissions in this chip, but it does not reveal the PUF data.

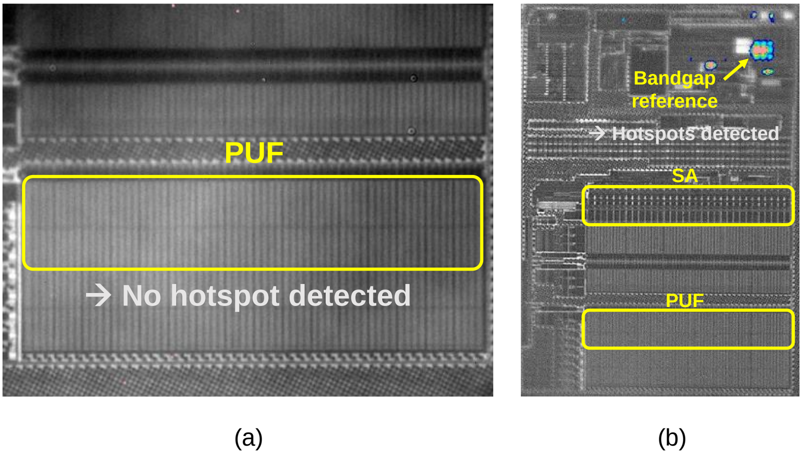


Figure 3-11 InGaAs images of a NeoPUF chip sample show (a) PUF array without a detectable hotspot. (b) Entire chip with hotspots found near the bandgap reference circuit.

3.7. Conclusion

NeoPUF has been proven to have all the required PUF properties through comprehensive experimental characterization. Using the Quantum Tunneling mechanism, NeoPUF demonstrates state-of-the-art reliability and robustness while effectively maintaining other properties, including randomness, uniqueness, and physical security. It can be concluded that NeoPUF is an elegant and highly reliable PUF solution, providing advantages over conventional PUF implementations.

4

A comparison of two aspects of NeoPUF and conventional SRAM PUF: feasibility for mass-production and vulnerability to physical attacks. Basic properties are compared to highlight the differences between these two types of PUFs. That is followed by the issue of mass-production, showing how reliance on an old technology impacts the feasibility of implementing a PUF in a volume-production device. Vulnerability against various attacks is evaluated in detail. The closing demonstrates that NeoPUF is a better solution for HRoT in an electronic system.

4. NeoPUF vs.SRAM PUF

After reviewing the ideal properties of NeoPUF, one may still wonder how it compares to conventional PUF implementations. Therefore, this chapter will provide an in-depth comparative analysis of NeoPUF and SRAM PUF, which are the most frequently discussed PUFs available, covering their basic functionality, mass-production feasibility, and vulnerability against physical attacks.

4.1. Comparison on Functionality

This section focuses on a functional comparison between NeoPUF and SRAM PUF, showing that NeoPUF is a better solution that can provide better functionality when used in security applications.

4.1.1. Data Stability

The most apparent difference between the two solutions is the stability of PUF data. As NeoPUF generates bits based on significant differences caused by the tunneling mechanism, excellent data stability is guaranteed. For SRAM PUF, because of its minimal threshold of voltage differences that determine the PUF-bit, the value of a PUF-bit can frequently vary while it is continually read. The difference between SRAM PUF and NeoPUF can be illustrated using the example shown in *Figure 4-1*. Even though the probability is very low, an SRAM PUF-cell might have transistor pairs without mismatch, making the power-up state uncertain. Hence, the PUF output is unstable. On the other hand, the tunneling path in a NeoPUF-cell will conduct a current level significantly higher than when there is no tunneling path. It shows that the readout of a NeoPUF is always stable, even in the worst case where two tunneling paths exist in one PUF-cell. Consequently, NeoPUF is a more practical solution compared to an SRAM PUF in terms of data stability.

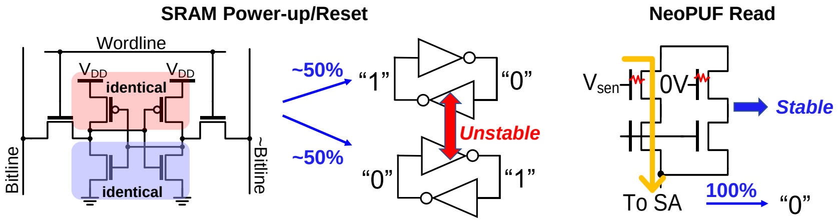


Figure 4-1 Worst-case stability comparison of SRAM PUFand NeoPUF.

4.1.2. Robustness

When a PUF is subjected to environmental changes, the generated PUF-bits are at risk of being flipped to an opposite value, causing the subsequent secret parameters or keys to change, effectively destroying the PUF's operation. As demonstrated in the previous chapter, NeoPUF is immune to two key environmental fluctuations, voltage and temperature, showing state-of-the-art robustness.

On the other hand, an SRAM PUF is prone to voltage and temperature variations because it relies on threshold voltage differences that can be significantly affected by environmental fluctuations. Consequently, NeoPUF is a more robust solution than the SRAM PUF. It is, therefore, more suitable for security applications that require use in a wide range of environments.

4.1.3. Reliability

In order to maintain a required device lifetime, a PUF should be resilient to aging effects that can cause fatal failures within the defined lifetime; e.g., ten years. As mentioned in *section 3.3.2,* NeoPUF has been evaluated under an accelerated aging test environment and has proven resilient. At the same time, the resulting BER remains zero during, and after, the long-term aging measurement.

On the other hand, the threshold voltage difference within an SRAM PUF-cell is sensitive to aging effects, and the value of PUF-bits in an SRAM device can be flipped after long-term usage, as illustrated in *Figure 4-2*. In this example, the threshold voltage V_T of the highlighted PMOS transistor may decrease due to aging effects and make the PUF cell less stable from time to time. The threshold voltage shift induced by the aging effect has two negative impacts on PUF operation.

The first impact is that a PUF-cell may become less stable; i.e., its BER increases. Secondly, the PUF-bit may be permanently changed so that it becomes impossible to derive the same PUF key. Consequently, using an SRAM PUF that is prone to aging effects bears the potential risk of temporary or permanently corrupted keys.

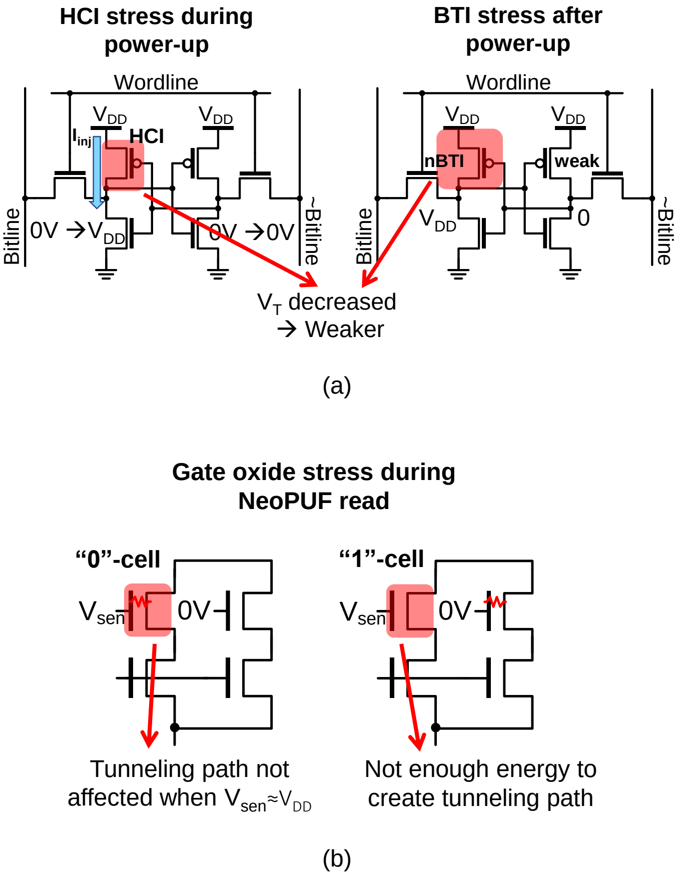

HCI stress during power-up
BTI stress after power-up
Wordline
Wordline
V_{DD}
V_{DD}
I_{inj}
HCI
nBTI
weak
Bitline
~Bitline
0V →V_{DD}
0V →0V
V_{DD}
0
V_T decreased
→ Weaker
(a)
Gate oxide stress during NeoPUF read
"0"-cell
"1"-cell
V_{sen}
0V
V_{sen}
0V
Tunneling path not affected when V_{sen}≈V_{DD}
Not enough energy to create tunneling path
(b)

Figure 4-2 Illustration of how aging effects impact (a) Conventional SRAM PUF. (b) Quantum Tunneling NeoPUF.

4.1.4. Randomness and Uniqueness

In theory, both NeoPUF and SRAM PUF have good randomness and uniqueness because the cells are designed to be fully symmetric, making all PUF-cells have the same probability of generating "0"-bits and "1"-bits, just like flipping a coin. In practice, however, an SRAM cell is not always symmetric, making it possible to have non-ideal randomness and uniqueness. Moreover, an SRAM array might have a dopant gradient across it, resulting in spatial dependence on the resulting PUF value. For NeoPUF, because the two competing gate oxides are placed at a minimum distance, the discrepancy between their oxide thickness is minuscule and will not impact the probability of the resulting bit-value. Using spatial dependence to guess the PUF data is therefore infeasible.

4.1.5. Latency

One often overlooked feature when deriving PUF data is latency. An SRAM PUF requires additional processing steps to stabilize the final data result. In a typical SRAM PUF, the PUF is read multiple times for temporal majority voting. Then the output from voting is processed by an error-correction algorithm to make the final data completely error-free. The entire process can take from microseconds to milliseconds, depending on how the post-processing is implemented. For NeoPUF, because the data is stable, it is instantly ready.

Having an instantly ready PUF is very beneficial because it can be used to secure the system right after power-up. For example, a PUF-based key can be immediately derived after power-up to decrypt firmware and perform an integrity check to secure boot flow. However, in the case of SRAM PUF, if the post-processing algorithm is implemented in software, it will require the firmware to be loaded before a PUF key is derived. In this case, a PUF key cannot secure the initial boot flow.

4.2. Feasibility for Mass-production

This section focuses on analyzing the feasibility of PUFs for mass-production in a chip foundry. The primary considerations are portability across technology platforms, yield, and testability.

4.2.1. Technology Dependence

A PUF can be applied in various applications and therefore implemented on various fabrication process nodes. Moreover, a design in a specific node may include different options such as low power or high performance. The PUF may be fabricated using a slightly different recipe within the same node and the same option because the processing steps are adjusted on the fly in a foundry. For these reasons, a PUF needs to meet the required specifications no matter how the technology changes.

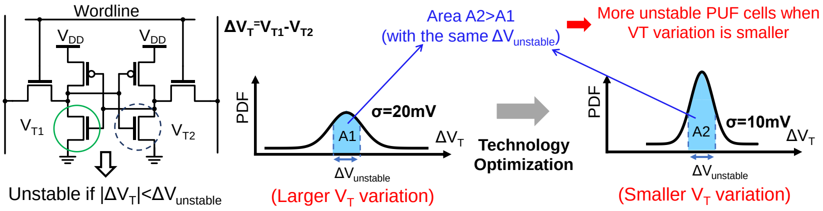

Wordline
V_{DD}
V_{DD}
V_{T1}
V_{T2}
Unstable if $|\Delta V_T|<\Delta V_{unstable}$
$\Delta V_T=V_{T1}-V_{T2}$
PDF
A1
σ=20mV
ΔV_T
$\Delta V_{unstable}$
(Larger V_T variation)
Area A2>A1
(with the same $\Delta V_{unstable}$)
Technology Optimization
More unstable PUF cells when VT variation is smaller
PDF
A2
σ=10mV
ΔV_T
$\Delta V_{unstable}$
(Smaller V_T variation)

Figure 4-3 Illustration of how technology optimization impacts device variation and the behavior of an SRAM PUF.

For SRAM PUF, a mismatch between the inverter pair is caused by process variations, which strongly depend on the process technology. For example, as illustrated in *Figure 4-3*, it is assumed that the difference between V_{T1} and V_{T2}, denoted as ΔVT, is a random variable following the probability distributions as specified in the figure. If an SRAM PUF is implemented in an emerging technology, the standard deviation of V_{T1} and V_{T2}, as well as the ΔV_T, will be more significant than the case that is implemented in a mature technology, because the processing steps for emerging technologies have not been well optimized. As illustrated in the figure, the difference of standard deviation σ will impact the probability of ΔV_T to locate in the unstable region (colored area). Since area A2 is larger than A1, the SRAM PUF implemented in a mature technology node will tend to be less stable.

Another risk of an SRAM PUF is that it must be first characterized to know the worst-case error percentage to design a suitable error-correction scheme. However, this worst-case error percentage can vary with technology options and process recipes. Consequently, one should design an error-correction scheme with a large margin to accommodate such variations; otherwise, a different correction scheme should be designed for a different batch of PUF chips, increasing overall costs. For NeoPUF, Quantum Tunneling technology migrates well to all process technologies and the resulting current differences between the two possible cases. As a result, the performance of NeoPUF is excellent across a wide range of technology platforms and process options, making it a better solution.

4.2.2. Yield and Reliability

PUF and PUF-based security solutions are typically considered security add-ons, providing an additional layer of protection. However, they are an essential part of security in electronic systems, especially as cybersecurity grows in importance. System designers tend to consider using a PUF only if it provides benefits without negative consequences. One concern is that a PUF embedded in an SoC may degrade overall yield at the chip fabrication stage. In other words, the designer will typically sacrifice the benefits of PUF in favor of a simpler path to mass-productio.

For SRAM PUF, randomness, robustness, and reliability are all technology-dependent, and these factors will impact yield quality. Once an error-correction scheme is designed to cover the range of operating conditions for example, with a worst-case estimated error percentage of 15%. This may result in a 100% yield when the first few batches of devices are fabricated. But if the manufacturing recipe is changed, the yield during the ramp-up stage is likely to drop. A big tradeoff comes when designers give too much emphasis to the error correction scheme to prevent yield drop. As aging effects ensue, SRAM PUF without a complicated error correction scheme will also encounter functional failures if the resulting error percentage exceeds the ECC capability.

Consequently, since the reliability of SRAM PUF varies for different process nodes, potential yield problems cannot always be captured at the prototype stage. As a result, the final product may fail to achieve profitability goals. On the contrary, since NeoPUF is highly robust and reliable in all advanced technology nodes, the yield has been qualified at 100%, making NeoPUF suitable for mass-production in all these nodes. Furthermore, since NeoPUF is immune to aging effects, products that are embedded with it will enjoy greater longevity.

4.3. Vulnerability Against Attacks

PUFs must be evaluated for vulnerability against attacks as a fundamental part of system security. In a typical case, PUF data is used to derive secret keys that cannot leave the chip or a specific security boundary within it. This constraint must be followed to make many security claims valid. An attacker may try to break through by applying various physical attack techniques. This section will therefore provide an in-depth analysis of vulnerabilities for both SRAM PUF and NeoPUF, showing that NeoPUF is indeed a more secure PUF solution.

4.3.1. Optical Attacks

An optical attack, one of the most widely used techniques, is typically performed using a laser. For example, a successful scheme for stealing data in a SRAM PUF is outlined in [3], which is done by injecting laser pulses into the SRAM areas of a de-packaged chip from its backside.

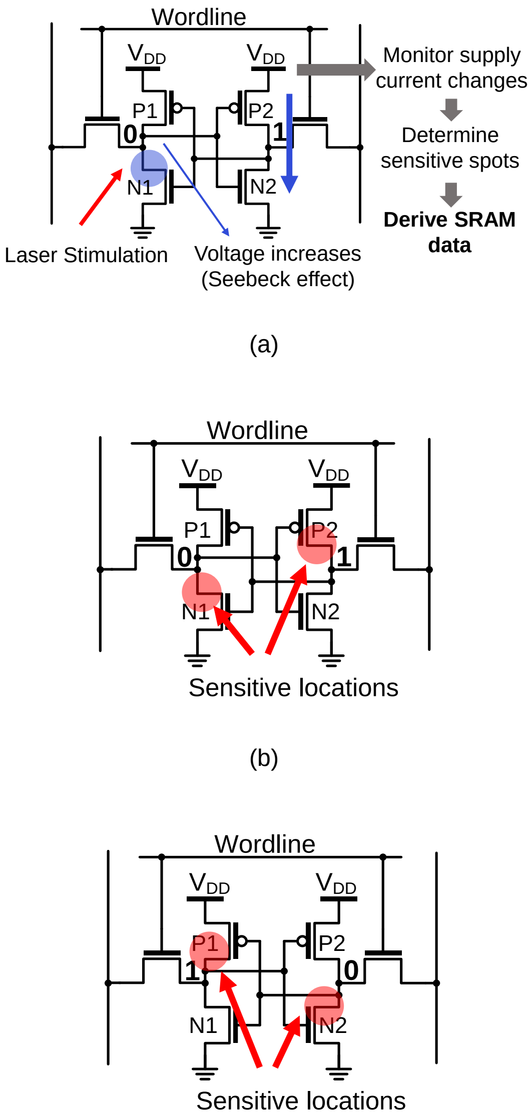
With CMOS logic, the junction bias within the cell will be different depending on the input/output states. In the case of SRAMs or registers, the junction bias will also depend on the stored value. If a laser pulse hits a reverse-biased junction, there will be charge carriers generated in the depletion region, inducing a photocurrent at that location. If the magnitude of this photocurrent is large enough to affect the logic state, a bit-flip could occur. This is a laser fault, which an attacker can exploit. Successful laser attacks can extract key data from SRAM PUFs.

Another type of attack is done by analyzing responses to laser pulses. One method is using a laser beam to induce local heating at a particular device. Different amounts of power will drain out after heating, based on the bit-value stored in the affected SRAM cell.

As illustrated in *Figure 4-4 (a),* if the SRAM cell is currently at the state with 0 at the left node and 1 at the right node, and if the laser pulse has hit the highlighted spot, the voltage of the left node will be increased due to the Seebeck effect. While the voltage of the left node increases, the NMOS transistor N2 will be turned on and start draining current.

As a result, if hitting this spot with laser pulses, and a sudden increase of the supply current is observed, one can determine the state of the targeted SRAM cell. As illustrated in *Figure 4-4 (b)-(c),* an SRAM cell will have different sensitive spots when it is in different states. Hence, shooting laser pulses to these spots can help to distinguish the internal state of an SRAM PUF. By using this type of attack, it is possible for the attackers to steal data from an SRAM PUF.

On the other hand, since the value of NeoPUF is not stored as a logical state, there is no difference in junction biases within NeoPUF-cells containing different bit-values. Shooting a laser at a NeoPUF-cell will not affect its operation, implying that the cell will not produce observable response in the event of laser injections. As a result, NeoPUF can thwart attackers who utilize lasers to steal PUF data.



(a)

(b)

(c)

Figure 4-4 (a) Illustration of obtaining SRAM data through laser stimulation. (b) Sensitive spots when the SRAM cell is at the “1” state. (c) At the “0” state.

4.3.2. Temperature Attacks

Another type of attack is performed by heating or cooling PUFs. If the temperature is manipulated, there can be either information leakage such as changes in current consumption, or faults that attackers can exploit. This behavior occurs because SRAM PUFs are highly sensitive to temperature, and hence the outcomes and the stability of the results can be easily affected.

Moreover, there is an attack technique for an SRAM PUF that exploits the data remanence effect, as noted in [26]. The data remanence effect occurs in latch-type memory cells, including SRAM cells. While these memory cells hold stored data until the SRAM is reprogrammed or powered down, the charges stored within a cell may remain some time after shutdown. This effect is known as data remanence.

As shown in *Figure 4-5*, a data remanence attack could target a non-dedicated SRAM PUF in which the SRAM cells not only generate PUF data but also serve as storage for regular operations. In this scenario, the power-up state of the SRAM array is read once after power-up, and the resulting data is protected from non-privileged users to ensure the confidentiality of the PUF-based key. After the key is derived, the SRAM is entirely reset to clear traces of the PUF data, and then the SRAM can be accessed by other user programs to read or store data.

A data remanence attack on this type of design would freeze the SRAM cells right after power-on, where the initial state is already stored within the SRAM cells, and then cut power to the SRAM to disable the reset procedure. After the reset is disabled and charges remain in the SRAM cell, the initial power-up SRAM data is not cleared. The remaining SRAM PUF data will then be accessible by user programs because it is no longer protected by access policies.

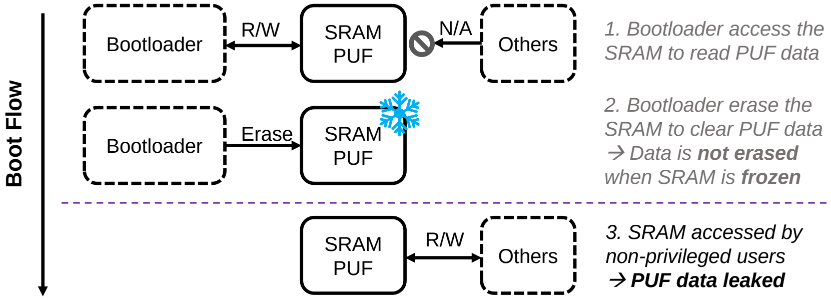

Boot Flow
Bootloader
R/W
SRAM PUF
N/A
Others
1. Bootloader access the SRAM to read PUF data
Bootloader
Erase
SRAM PUF
2. Bootloader erase the SRAM to clear PUF data → Data is **not erased** when SRAM is **frozen**
SRAM PUF
R/W
Others
3. SRAM accessed by non-privileged users → **PUF data leaked**

Figure 4-5 Data remanace attack example on a SRAM PUF.

As mentioned in [26], the proposed method of reading PUF data with a user program is feasible in theory, but it is not practical because too many variables are involved. Another approach is using a more invasive method that requires direct access to the PUF data using techniques such as micro-probing or laser voltage probing.

These invasive methods will typically require the preparation of a de-constructed sample, which may clear the SRAM PUF data. It is more likely that the data remanence effect of a frozen SRAM PUF may keep the power-up data exploitable for an attacker. Moreover, some of the countermeasures are only active when the circuit is powered-up. Using the data remanence effect allows an attacker to bypass these countermeasures by cutting power and attacking the residual data within the SRAM PUF.

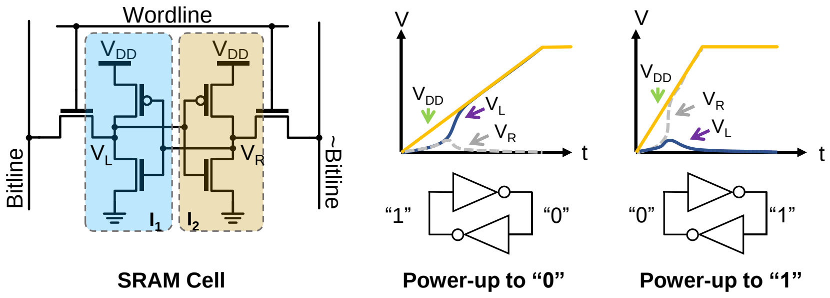
On the other hand, since NeoPUF and its Quantum Tunneling mechanism are insensitive to temperature, the varying temperature is therefore not a valid attack vector. The data remanence effect is not exploitable for NeoPUF, because there is no latch-type structure within the PUF-cell. Moreover, there is no static charge stored in the cells that could cause observable data remanence effects.

4.3.3. Voltage-glitch Attacks

While most PUF attacks focus on revealing hidden PUF data, simply inducing faults in derived data is also a valid attack. In addition to modifying temperature, as mentioned in the previous section, varying the supply voltage is also effective in inducing faulty PUF data. A conventional PUF solution is a post-processing algorithm with an error correction capability designed for the worst-case error percentage caused by Process Voltage Temperature (PVT) variations. The error-correction algorithm is typically chosen based on normal operating conditions. An attacker can use out-of-spec conditions to cause aberrations that will make the error count exceed the error-correction capability.

One of the simplest methods to create such conditions is to use a supply voltage outside circuit specifications. Since an SRAM PUF is very sensitive to the operating voltage, an attacker can raise the supply voltage, for example, from 1V to 1.5V. In this case, the resulting PUF data is likely to have so many errors that it becomes unrecoverable by the error correction algorithm. An attacker can also induce errors by adding random positive or negative glitches in the power supply while the PUF data is generated.

Moreover, because an SRAM PUF is also sensitive to its power-up conditions [27], an attacker can also induce faults by changing conditions when an SRAM PUF is powered up. For example, an SRAM cell may be changed to a different state if the power-up rate is modified, as illustrated in *Figure 4-6*. These voltage-related methods can all be exploited separately or in combination by an attacker to induce faults in the final output of an SRAM PUF.



SRAM Cell

Power-up to "0"

Power-up to "1"

Figure 4-6 Example illustration on the power-up state of an SRAM PUF being affected by the power-up ramp rate.

For NeoPUF, because it operates under a broader range of tolerances without bit errors, an attacker is therefore unable to induce faults through changes or spikes in supply voltage. In conclusion, NeoPUF has better immunity against fault attacks based on voltage glitches than an SRAM PUF, indicating that NeoPUF is a better solution in terms of physical security.

4.3.4. Helper-data Attacks

A conventional PUF-based solution using SRAM PUF requires helper data for error correction and bit selection that is stored in an NVM block. Keeping stored helper data confidential in most algorithms is unnecessary. However, it must remain error-free and unmodifiable. The error correction capability is only designed for errors occurring while reading the PUF, but not for errors in the helper data. Moreover, modifying helper data may be equivalent to manipulating PUF data, and that is a serious security threat for a system using a secret key derived from an SRAM PUF.

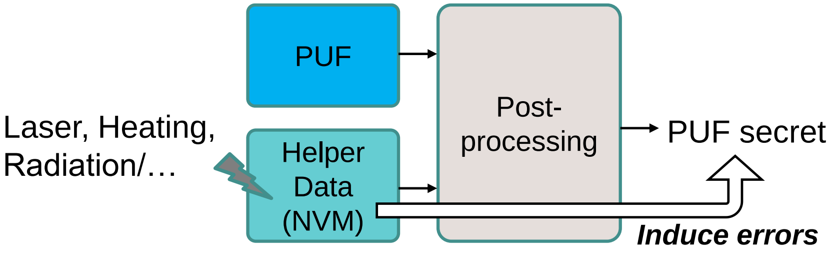


Figure 4-7 Example illustration of attacking the helper data storage.

The helper data is typically stored in a flash memory implemented in a floating gate technology. The charges captured in the floating gate are used to determine the stored data, and the charges will not be removed from the gate unless the cell is subjected to a high erase voltage. The helper data is considered stored permanently if the circuit is always operated within specified conditions.

However, an attacker can violate such rules to make the circuit and the memory block operate at more extreme conditions that cause data loss and read errors [28]. The extreme conditions can be locally forced with local heating from laser pulses or, more widely applied, by baking a chip in an oven or placing it inside a radiation chamber. In such cases, the helper data can no longer be considered error-free, and the final PUF output may have faults that an attacker can exploit.

An attacker can also try to program helper data to a specific value if the access policy is violated and the write permission to the helper data memory is wrongly granted. A straightforward approach for an attacker is to roll back the helper data to a previously recorded value after an update. Because the confidentiality of helper data is not secure, the previously programmed helper data can be recorded by an attacker. If a device needs to be discarded and the secret key must be destroyed, the simplest step is to erase the helper data to make the key unrecoverable.

However, as illustrated in *Figure 4-8*, an attacker who has already recorded the helper data and has obtained the discarded device can revive the device by giving it the ability to reconstruct the destroyed key. The same approach can be applied if the key is updated, and in this case, an attacker can make the device use an older key to perform critical cryptographic operations, in which the older key may be already compromised and is no longer considered secure. For NeoPUF, there are no such risks because helper data is not required. As a result, it is immune to helper data manipulation attacks and is therefore a better solution from this perspective.

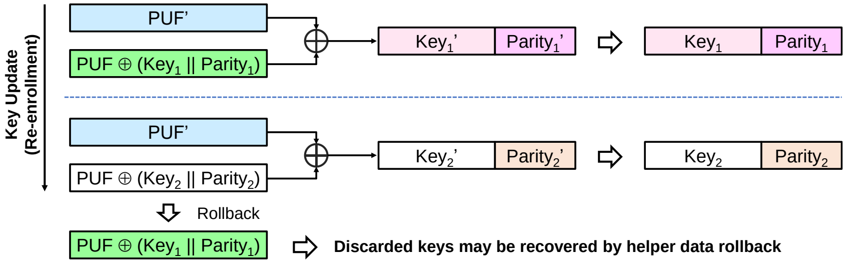


Figure 4-8 Possible rollback attack on helper data to recover a discarded key.

4.4. Conclusion

This chapter has introduced the critical aspects of choosing the right PUF solution. In addition to quantifiable PUF properties, the overall impact on circuit reliability, production yields, and system security should be considered.

As summarized in *Table 4-1,* compared with legacy SRAM PUF technology, NeoPUF is clearly a better choice with overwhelmingly better performance in terms of reliability and robustness. In addition, while an SRAM PUF brings potential risks to overall fabrication yield because of its high reliance on technology, environment, and aging effects, NeoPUF poses no risk to overall yield, making it a better choice for mass-production. Finally, when considering different security risks from various attack techniques, NeoPUF has greater immunity against such threats than SRAM PUF. It can therefore be concluded that NeoPUF is a better choice in all the discussed situations.

Table 4-1 Comparison table of Ideal PUF, SRAM PUF, and NeoPUF

	Ideal PUF	**SRAM PUF**	**NeoPUF**
Randomness (HW)	50%	45%~50%* [1]	~50%
Uniqueness (HD)	50%	40%~65%* [1]	~50%
Stability (BER)	0%	1%-20%* [1]	0%
Aging Effects	No	Yes	No (HTOL 1000hrs)
Pre/Post-processing	Unnecessary	Burn-in, TMV, Masking, ECC	Unnecessary
Traceability	No	Conditional	No
Instantly Ready	Yes	No	Yes
Radiation Hardness	Rad-Hard	Not Rad-Hard	Rad-Hard

5

A high-level overview of PUF-based HRoT solutions and applications. Basic security applications are introduced that can benefit from NeoPUF, including key generation and digital asset protection. Next, the basic requirements for a HRoT are described. The chapter closes with a description of the generic structure of a NeoPUF-based HRoT solution, namely PUFrt, and its essential use cases.

5. PUF-based Security Applications and Root of Trust Solutions

This chapter introduces generic PUF-based security applications and how they can benefit from NeoPUF technology. In addition, the concept and use cases of a PUF-based Root of Trust (RoT) will be illustrated.

5.1. Security Applications

Since PUF provides a unique secret for every device, many security applications can leverage this outcome to significantly enhance the security strength of edge devices and networks.

5.1.1. Application Fields

Given that a PUF is a more cost-effective solution compared to a reserved key-storage block in NVM, it is undoubtedly a good choice for applications requiring lightweight design. One crucial application in this category is the Internet of Things (IoT). In an IoT system, if a vast number of edge devices are deployed without strict supervision, those devices would be potentially vulnerable to both invasive and non-invasive physical attacks. As mentioned earlier, PUFs are more resilient to physical attacks than conventional key-storage solutions. Moreover, we have shown that NeoPUF technology has better resilience than conventional PUFs. It can therefore be concluded that NeoPUF is a superior choice for security applications in IoT devices.

Moreover, automotive electronics is a hot topic in the future and both vehicles and drivers will depend more on huge technological improvements provided by embedded electronic systems. While demand for vehicular electronics has rapidly increased, the security threats brought by increasingly sophisticated vehicular networks remain an essential issue to be solved. A search for security solutions in cars yields few choices because of the strict reliability and robustness requirements for embedded electronic systems and devices. Unlike most applications operating under indoor conditions, automotive electronics must work under much broader environmental conditions. For example, car electronics can reach very high temperatures when they are located close to an engine.

Due to such fundamental constraints, not all PUF solutions are suitable for automotive applications because they lack good reliability and robustness. NeoPUF, on the other hand, is highly reliable and robust, especially with its good data retention properties at high temperatures, making it a natural choice for automotive applications.

5.1.2. Key Management

Key generation is a significant application for PUFs, where each device has a unique secret key derived from secret parameters generated by an on-chip PUF circuit. This concept can be extended to a general key management scheme, which not only consists of key generation but also key-wrapping functionality and a secure access policy.

For the management of cryptographic keys, the confidentiality and integrity of keys are extremely important. For confidentiality, a key derived from NeoPUF has this property by nature because NeoPUF has good resilience against malicious attacks, and its data is safe from prying eyes. As described in *Figure 5-1*, using a PUF-based Root of Trust to derive the unique key for each chip is more secure and cost-efficient, comparing to the traditional methods such as key injection and utilizing standalone secure elements.

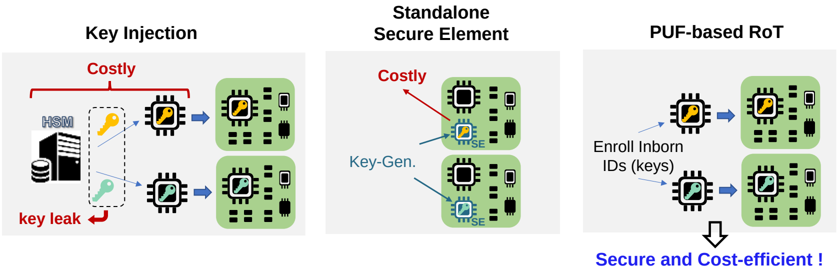


Figure 5-1 Comparision of different key provision methods.

One popular key management application using PUF-based Root of Trust is to derive public key pairs from the unique feature of the PUFs. As illustrated in *Figure 5-2*, a key generation function takes the PUF data as its input to derive a pair of a public key and a private key. Since both keys are derived from the device-unique PUF data, they can serve as the unique identifier for this device. The public key can be public and registered as the device's unique ID or a part of it. On the other hand, the private key should be kept secret and used to derive unique signatures to proof the device's identity in different circumstances.

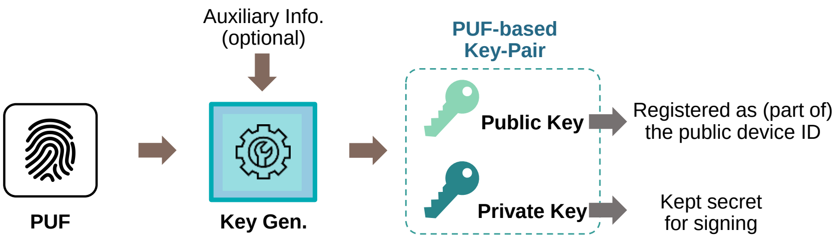


Figure 5-2 Example PUF-based public-key pair generation flow.

5.1.3. Key Wrapping

In practice, not all required cryptographic keys are derivable from PUFs. One example is a session key that is derived following a key-exchange protocol. In such cases, the confidentiality of these keys is not intrinsically guaranteed by the key source, because the keys or the secret parameters that are required for key derivation need to be stored in an NVM block, which has less immunity against tampering. To protect the confidentiality of these secrets, they can be encrypted using a PUF-based secret key, which acts as a root key or a so-called KEK (key encryption key), as the example illustrated in *Figure 5-3*.

The encrypted values can be stored within a non-secure memory block without the risk of being leaked. In this scenario, the PUF-based KEK is the Root of Trust that must have the highest security strength, implying that the uniqueness and physical security of the PUF should be well-qualified. NeoPUF is a viable source for deriving a KEK. For the integrity of cryptographic keys, it is important to make sure that all keys stored on-chip cannot be modified, both secret keys and public keys. For public keys, there are no confidentiality concerns, but these keys and the protocols relying on them are prone to malicious manipulations.

To ensure integrity, a PUF-based key can be used to derive message authentication codes for integrity checks of one or more keys located in the embedded NVM block. Moreover, a PUF provides a device-unique secret for each chip that can also be used to protect the internal operations of the key management agent and further enhances the resilience against tampering attempts.

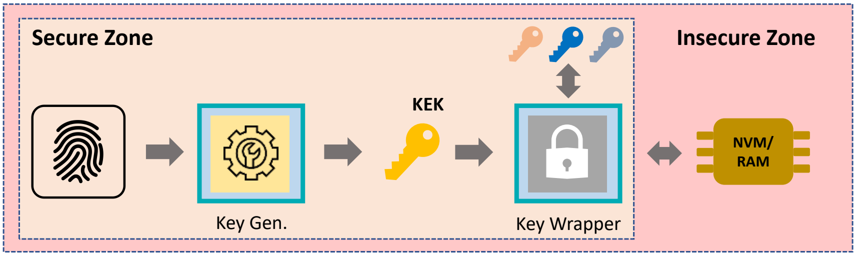


Figure 5-3 Example Key Wrapping scheme for storing keys in insecure zones.

5.1.4. Firmware/Model Protection

One more important security application for PUFs is to protect digital assets that are stored in edge devices. Such assets can be device-specific firmware, an artificial intelligence inference model, or silicon IPs. All these examples require costly development resources and may cause significant profit losses if digital assets are stolen from a device. To counter this type of threat, PUF-based solutions are able to provide strong protection against malicious attempts on different targets.

A set of firmware codes or an inference model stored in flash memory are vulnerable targets if the stored data is not encrypted and the flash itself is not security certified. One common practice is hardwiring a key into such a device and using this key to encrypt flash data. Assuming that this key is resilient to reverse engineering, an attacker would then be unable to decrypt the contents of the flash memory, even if it was possible to clone all the data stored within it.

This scheme, however, still allows an attacker to buy an over-produced device that only lacks the correct firmware or inference model. The attacker can then program the encrypted data into the flash memory. After programming the encrypted data, the repurposed device will function as a legitimate one because it can decrypt and use the correct digital assets. It will be even worse if the hard-wired key can be reverse-engineered, allowing not only surplus devices to work but also counterfeit devices.

A more secure scheme benefits from the fact that a PUF-based key is unique for every device. The firmware or the inference model is programmed into flash, either encrypted or in plaintext if the data is programmed in a secure environment. For example, as illustrated in *Figure 5-4* the AI assets like models and parameters may be programmed into the NAND flash in a factory or through an over-the-air (OTA) update. These assets are initially protected by a global encryption key that is the same for every device, and the encryption key will be changed to a local key derived from PUF. The main body of these essential assets will be (re)-encrypted using the unique PUF-based asset protection key. Under this scenario, the assets stored in flash will not fall into the wrong hands if the security of the PUF-based key is guaranteed.

Even if an attacker has the ability to copy the flash contents into another device that is not programmed with the correct firmware, this new device cannot decrypt or use the content because it does not have the same PUF and subsequent key. To meet expectations for high security, PUF-based keys must have very good physical security. For these reasons, NeoPUF is a suitable choice.

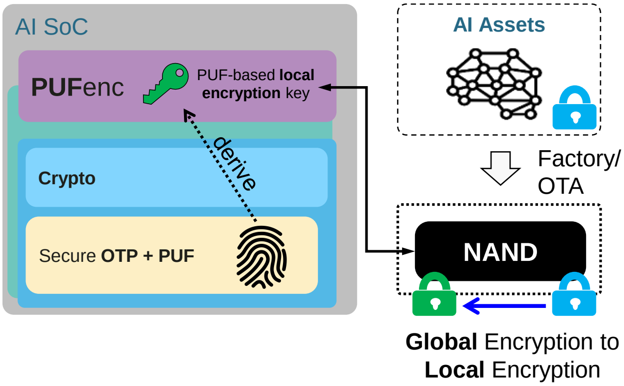

AI SoC
PUFenc
PUF-based local encryption key
derive
Crypto
Secure OTP + PUF
AI Assets
Factory/ OTA
NAND
Global Encryption to Local Encryption

Figure 5-4 Protecting AI Assets in NAND Flash using the PUFenc solution.

Execute-in-Place of encrypted codes

In some applications, the firmware or model are stored in an embedded or external NOR flash with a higher speed. In such cases, the step of loading the code into RAM before execution may be omitted, enabling the so-called “execute-in-place” feature. In a conventional scheme of executing encrypted codes, the code need to be first loaded and decrypted into RAM before execution, due to the speed limitation of data decryption.

By adding a high-speed cryptographic engine extension to the PUF-based root of trust, a secure execute-in-place solution can be realized. As illustrated in *Figure 5-5*, the NOR flash is encrypted by the PUF-based key, and can only be decrypted back using the same PUF-based key. While the execute-in-place feature is mainly enabled by the high-speed cryptographic engine, the security of this entire solution is ensured by the unique key derived and from the PUF.

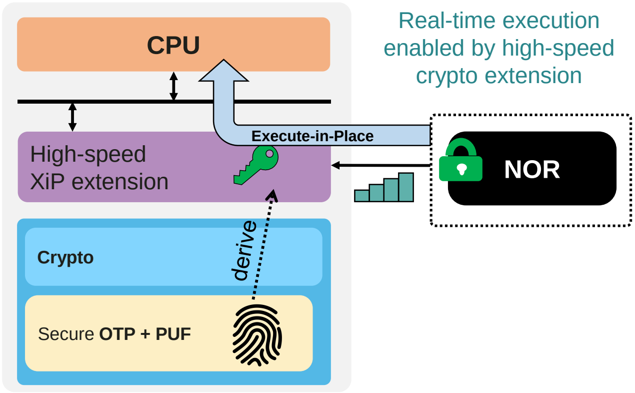


Figure 5-5 Enabling secure execute-in-place solution by combining PUF and high-speed crypto extension.

5.1.5. IP Protection

Another digital asset is silicon IP (intellectual property) which is sold by IP vendors to chip designers. While the IP vendor does not control the final product, it is difficult to verify whether IPs have been replicated without payment of the proper royalties. To prevent unauthorized replication, one solution would be to compare the number of manufactured chips containing the IP with the number reported to the IP vendor. This method is, however, infeasible if unauthorized chips are sold on the gray market, where transactions are hard to track.

A PUF can provide a simple solution to prevent unauthorized production. The concept is to embed an IP vendor's public key inside the delivered IP together with a PUF macro and some essential cryptographic functions that can be used later on to activate the IP. After the chip is manufactured with a hardened silicon IP, the PUF will provide device-unique secret parameters, and a public identifier can be derived from the PUF.

As illustrated in *Figure 5-6*, to perform the activation procedure, the public identifier is read from the PUF, which could be either a hashed value of the PUF data or a public key derived from the PUF data. This identifier is unique to the particular device and will be sent to the IP vendor together with an activation request. After receiving the activation request and the identifier, the IP vendor will sign this identifier using its private key, which is uniquely bound to the public key pre-installed in the IP. This signature goes back to the activation site and is now considered an activation token that will be programmed into the device.

The crypto functions within this IP block will verify the activation token using the pre-installed public key and the PUF data. This IP will then be correctly activated with all the functions enabled if the activation token is successfully verified. If the token is not verified or does not exist in the device, the IP will be disabled or only provide limited functionality.

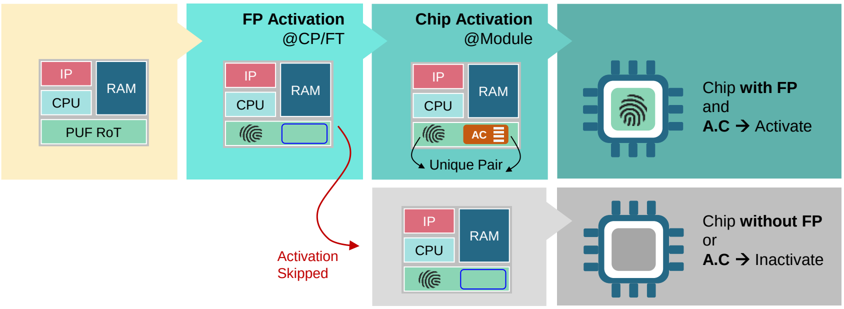


Figure 5-6 Example chip activation flow to prevent over-production.

This concept works because the identifier derived from PUF is unique for every device, making the activation token uniquely bound to a device and useless to activate an alternative one. Since an inactivated IP is useless, a chip designer will not be able to sell unauthorized chips, and the number of sold devices will be less than or equal to the activated devices. An IP vendor can therefore collect royalties based on the number of activated devices, without the need to check how many devices have actually been sold, preventing profit losses caused by unauthorized production.

5.2. PUF-based Root of Trust Solution

In the last part of this book, a brief introduction to Root of Trust solutions is discussed, providing some basics to help the readers understand more advanced topics in the following few books.

Several industrial security solutions built upon a PUF-based Root of Trust will be introduced in detail.

5.2.1. Hardware Root of Trust

A Root of Trust solution designed in hardware is an anchor for security at the physical layer. As illustrated in *Figure 5-7*, the security function in this SoC is enabled by the crypto-coprocessor accompanied by a hardware Root of Trust, in which the later one is utilized to protect and manage the secret parameters required by the security functions.

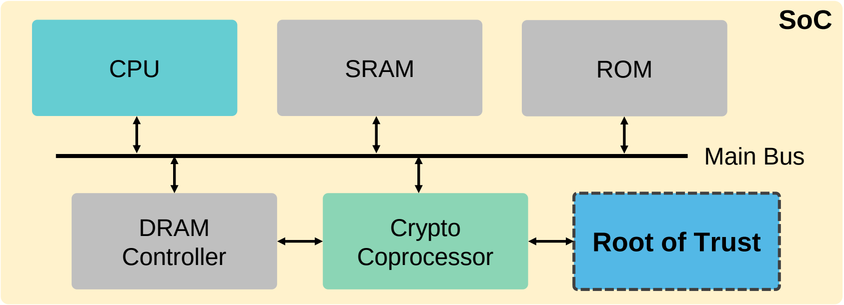


Figure 5-7 SoC with an integrated Root of Trust.

As mentioned at the beginning of this book, a PUF is an essential component for the design of a secure and reliable hardware Root of Trust. Nevertheless, it is worth remembering that there are solutions other than PUFs when creating a hardware RoT. Moreover, it should be noted that a standalone PUF cannot qualify as a good RoT solution without support from other hardware primitives.

A good hardware Root of Trust solution consists of a hardware block for randomness generation and another block that functions as tamper-proof storage. In a conventional RoT solution, the randomness-generation block is typically a true random number generator, which can derive permanent secrets such as a private key required for signing, and secrets that are required on the fly, such as a session key or a nonce. Tamper-proof storage is typically a secure non-volatile memory that is used to hold permanent secret keys and to secure parameters. At the same time, a PUF provides the functionality of randomness generation and tamper-proof storage that can therefore be used as a RoT building block.

Despite the many advantages of PUFs, they have limitations making it impossible to use them as a standalone RoT solution. One limitation is that PUFs cannot generate new random numbers on the fly, and they cannot be freely programmed with user-defined values. Consequently, even though PUFs can significantly improve the security of a RoT solution, other hardware primitives should also be used in order to fulfill the requirements of targeted security application.

In summary, a comprehensive RoT solution requires a secure storage, a PUF, a true random number generator (TRNG) and anti-tampering features, as illustrated in *Figure 5-8*. An OTP-based secure storage is used to keep the secret parameters safe. A unique chip fingerprint provided by PUF is also an essential component as mentioned earlier. A true random number generator can generate fresh randomness on demand, which is required in many cryptographic operations and security protocols. A complete set of anti-tamper design is usually desired in order to protect the RoT against various physical attacks.

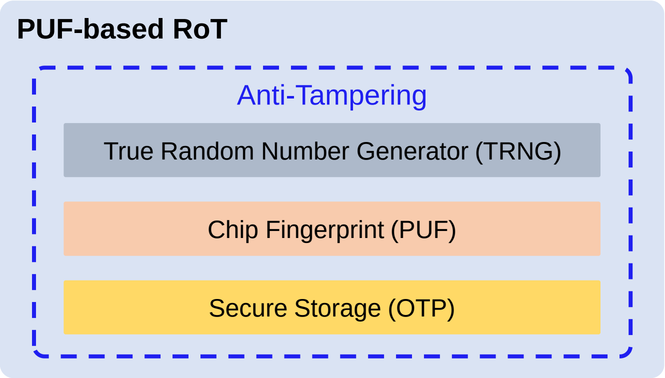


Figure 5-8 Simplified block diagram of a PUF-based Root of Trust.

5.2.2. Comparisons of RoT Solutions

There are several ways to achieve the required RoT functionality, and we, therefore, provide here an introductory comparison of these different RoT solutions. Firstly, a RoT may have its secret keys stored in an eFuse memory block. An eFuse memory block is constructed of narrow metal lines that can be programmed from closed to open circuits to represent “1”s and “0”s. This eFuse has high reliability and robustness that both allow secrets to be permanently kept in a chip. However, eFuse has a significant drawback that is programmed and unprogrammed eFuse cells can be clearly distinguishable under a microscope, as demonstrated in *Figure 5-9*. Therefore, a RoT based on eFuse is not an ideal solution due to its relatively poor resilience against visual inspection.

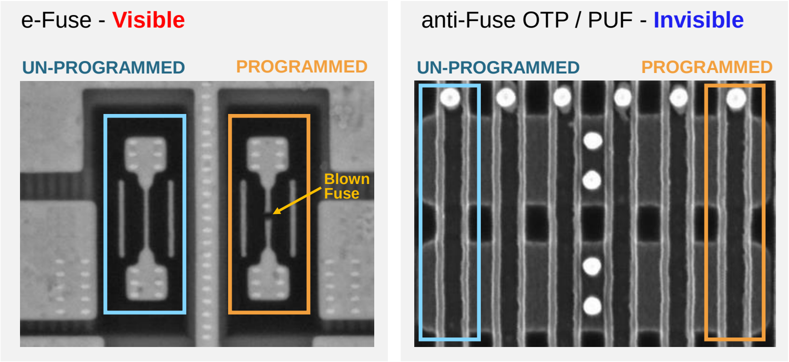


Figure 5-9 Visibility comparison between the eFuse and anti-fuse based OTP/PUF cells under a scan electron microscope.

For devices requiring better security, it has become more common to replace eFuse with an anti-fuse OTP memory, which uses a concept similar to NeoPUF: oxide-tunneling characteristics to distinguish the logic states stored in cells. Unlike NeoPUF, which is specially designed for high security, not all OTP designs target security applications, and therefore some may employ programming schemes that result in hard oxide ruptures, which are visible under inspection by an electron microscope. As a result, the only OTP schemes suitable for implementing a highly secure hardware RoT use Quantum Tunneling technology.

For both conventional eFuse-based or OTP-based RoT, protected secrets are not inborn. That is, the secrets are not created right after a chip is manufactured. In such cases, permanent secrets need to be programmed into the chip in a secure environment or generated on-chip using a true random number generator within a secure boundary. As a result, implementing these required secrets is more costly than having them inborn within the chips using PUFs.

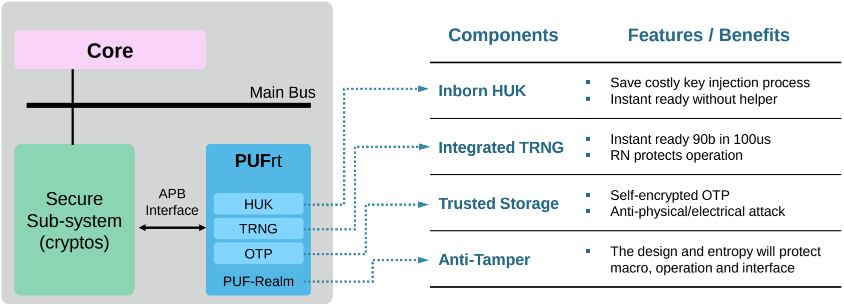
For these reasons, a more secure and cost-effective solution is to create a hardware RoT using NeoPUF and a secure anti-fuse OTP technology in which the PUF can provide the essential inborn secrets, and the secure OTP can keep the additional secrets or public information safe. This solution provides confidentiality to the stored secrets through the intrinsic security of the PUF and OTP technology, and it also provides integrity to these secrets because the PUF and OTP cannot be maliciously modified.

It should be noted that a comprehensive RoT solution should include a qualified true random number generator (TRNG) because random numbers are resources that are consumed in all kinds of cryptographic operations, no matter how the permanent secrets are generated or stored. As a result, for all possible RoT implementations, a TRNG is an essential circuit primitive that lies outside any comparison of various RoTs.

5.2.3. A Comprehensive RoT Solution Based on NeoPUF

As previously remarked, combining a Quantum Tunneling PUF and an anti-fuse OTP is an ideal solution to generate and keep secrets on-chip, and therefore our proposed comprehensive RoT solution has a structure as shown in *Figure 5-10*. The secret parameters are securely stored within the OTP or derived from the PUF. The stored secrets are further protected by the anti-tampering techniques that are supported by inborn PUF entropy and the dynamically generated randomness from the TRNG.

This concept can be understood as keeping essential secrets that are first protected by entropy through crypto operations such as key wrapping, while physical security is further guaranteed by an anti-tampering shell that features many countermeasures against different types of attacks.



Components	Features / Benefits
Inborn HUK	▪ Save costly key injection process ▪ Instant ready without helper
Integrated TRNG	▪ Instant ready 90b in 100us ▪ RN protects operation
Trusted Storage	▪ Self-encrypted OTP ▪ Anti-physical/electrical attack
Anti-Tamper	▪ The design and entropy will protect macro, operation and interface

Figure 5-10 Comprehensive PUF-based Root of Trust solution, PUFrt, and an example showing how it can be integrated into a secure computing system.

5.3. Conclusion

Several generic security applications are discussed in this chapter, providing an overview of how PUFs are used in the field. In addition, the main concepts and important considerations for constructing a hardware Root of Trust are also illustrated, together with an analysis of different possible RoT topologies.

In summary, NeoPUF can enable many security applications and works as a more reliable alternative to replace circuit primitives in existing solutions. NeoPUF can also be a core technology when constructing a general hardware RoT solution. Therefore, we introduce here a generic implementation of a NeoPUF-based RoT. More details about designing RoT solutions and an in-depth explanation of anti-tampering features will be provided in further editions of this PUF book series.

6

A summary of the material discussed in the previous chapters and a preview of future publications in this series.

6. Conclusions

By going through the essential concepts of PUFs, this book has provided a reference to aid in the design of a PUF or the choice of an existing PUF solution. Summaries of each chapter below will close the first book in this series, followed by a preview of the following books

At the beginning of this book, the importance of hardware security and the Root of Trust was illustrated, showing that device security requires a carefully planned design methodology with careful consideration of hardware primitives in this burgeoning IoT era.

The basic concepts of PUFs were introduced, including physical orientation, properties, design techniques, and evaluation methodologies. Through this overview covering research over the past two decades, we outlined the advantages of PUFs and the design challenges. It can be concluded that PUFs undoubtedly will play an essential role in current and future hardware security. In the meantime, designing a PUF that fulfills all a designer's expectations remains a very challenging task.

One of the main challenges is the poor reliability and robustness created by uncontrollable process-oriented device variations. Since the beginning of PUF development, several methods have been proposed to eliminate intrinsic reliability flaws, mainly relying on mathematical error correction algorithms and circuit redundancy. These methods have proven effective, but they also create design issues such as excessive use of circuit resources. It can be concluded that an intrinsically reliable PUF is essential for designing a secure system.

We have proposed a PUF using a Quantum Tunneling mechanism and demonstrated its state-of-the-art performance, making it a core technology to build a secure and reliable hardware Root of Trust. The Quantum Tunneling mechanism of NeoPUF provides a proven entropy source, which can generate unpredictable device-unique secret parameters for cryptographic operations. Enrolled NeoPUF-cells are reliable and provide physically untraceable storage of quantum-based entropy, allowing reliable operations throughout the entire device lifetime and across environments. From the detailed experimental analysis of NeoPUF circuits, it achieves all essential PUF requirements.

In later chapters, NeoPUF was compared with traditional SRAM PUFs from different perspectives. NeoPUF not only achieved better performance with its properties closer to an ideal PUF, but it also provides a more feasible solution for mass-production. Moreover, NeoPUF is less vulnerable to a wider variety of attack techniques.

Based on NeoPUF technology, different security applications, including key management and digital asset protection, can be implemented. With the well-qualified security features of NeoPUF, a hardware RoT solution can be constructed in combination with a secure OTP technology and a true random number generator. This hardware RoT solution enables more advanced security applications.

In summary, this book has demonstrated that PUFs have unique security features that are the anchor in a secure hardware implementation. Furthermore, NeoPUF has been shown to achieve state-of-the-art performance enabling secure Hardware Root of Trust solutions for a better architecture to secure the connected world in this IoT era.

7

Appendixes of the book, including Abbreviations, References, and Index.

Abbreviations

A	AI	Artificial Intelligence
	ASIC	Application Specific Integrated Circuit
B	BCH	Bose–Chaudhuri–Hocquenghem
	BER	Bit-error-rate
	BTI	Biased-Temperature Instability
C	CMOS	Complementary Metal-Oxide-Semiconductor
	CRP	Challenge-Response Pair
E	ECC	Error Correction Code
	EM	Electromigration
H	HD	Hamming Distance
	HOTL	High Temperature Operating Life
	HRoT	Hardware Root of Trust
	HW	Hamming Weight
I	IC	Integrated Circuit
	i.i.d.	Independent and Identically Distributed
	IoT	Internet-of-things
	IP	Intellectual Property
K	KEK	Key Encryption Key
M	MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
N	nBTI	Negative Biased Temperature Instability
	NIST	National Institute of Standard and Technology
	NMOS	N-type Metal-Oxide-Semiconductor Transistor
	NVM	Nonvolatile Memory
O	OTP	One-Time Programmable Memory
P	pBTI	Positive Biased Temperature Instability
	PEM	Photon Emission Microscopy

	PMOS	P-type Metal-Oxide-Semiconductor Transistor
	PUF	Physically Unclonable Function
R	RoT	Root of Trust
	RRAM	Resistive Random-Access Memory
S	SA	Sense-Amplifier
	SEE	Single Event Effect
	SEL	Single Event Latch-up
	SEM	Scanning Electron Microscopy
	SEU	Single Event Upset
	SoC	System-on-Chip
	SRAM	Static Random-Access Memory
	TDDB	Time Dependent Dielectric Breakdown
T	TEM	Transmission Electron Microscopy
	TID	Total Ionized Dose
	TMV	Temporal Majority Voting
	TRNG	True Random Number Generator
X	XOR	Exclusive-or

References

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” *Annual International Cryptology Conference*., Springer, Berlin, Heidelberg, 1999.

[2] G. Piret, and J.-J. Quisquater, “A differential fault attack technique against SPN structures, with application to the AES and KHAZAD.” *International workshop on cryptographic hardware and embedded systems,* Springer, Berlin, Heidelberg, 2003.

[3] D. Nedospasov, et al., “Invasive PUF analysis.” *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography,* IEEE, 2013.

[4] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security,* ACM, 2002, pp. 148-160.

[5] R. Maes, “Physically unclonable functions: Constructions, properties and applications.” *PhD Dissertation,* Katholieke Universiteit Leuven, Belgium 2012.

[6] K.-H. Chuang, “Highly Reliable Physically Unclonable Functions: Design, Characterization and Security Analysis.” PhD Dissertation, Katholieke Universiteit Leuven, Belgium, 2020.

[7] A. Rényi, et al., “On measures of entropy and information,” in *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1: Contributions to the Theory of Statistics, The Regents of the University of California, 1961,

[8] W. Killmann and W. Schindler, “A proposal for: Functionality classes for random number generators,” *BSI*, Bonn, Germany, 2011.

[9] Rukhin, A., et al. “A statistical test suite for random and pseudorandom number generators for cryptographic applications”, *NIST special publication 800, 22-r1a,* 2010.

[10] Grasser, T., et al., “The paradigm shift in understanding the bias temperature instability: From reaction–diffusion to switching oxide traps.,” in *IEEE Transactions on Electron Devices,* vol. 58, no. 11, 2011, pp. 3652-3666.

[11] E. Takeda and N. Suzuki, “An empirical model for device degradation due to hot-carrier injection,” in *IEEE Electron Device Letters*, vol. 4, no. 4, 1983, pp. 111-113.

[12] R. Degraeve et al., “New insights in the relation between electron trap generation and the statistical properties of oxide breakdown,” in *IEEE Transactions on Electron Devices*, vol. 45, no. 4, 1998, pp. 904-911.

[13] I. Blech, and C. Herring, “Stress generation by electromigration,” in *Applied Physics Letters,* vol. 29, no. 3, 1976, pp. 131–133.

[14] A. Alvarez, W. Zhao and M. Alioto, “14.3 15fJ/b static physically unclonable functions for secure chip identification with <2% native bit instability and 140× Inter/Intra PUF Hamming Distance separation in 65nm,” *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1-3.

[15] J. W. Lee, Daihyun 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,” 2*004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, 2004, pp. 176-179.

[16] J. Delvaux, “Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs,” in *IEEE Transactions on Information Forensics and Security,* vol. 14, no. 8, 2019, pp. 2043-2058.

[17] R. C. Bose, and D. K. Ray-Chaudhuri, “On a class of error correcting binary group codes.” *Information and control 3.1, 1960*, pp. 68-79.

[18] M. Liu, C. Zhou, Q. Tang, K. K. Parhi and C. H. Kim, “A data remanence based approach to generate 100% stable keys from an SRAM physical unclonable function,” *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017, pp. 1-6.

[19] R. Maes and V. van der Leest, “Countering the effects of silicon aging on SRAM PUFs,” *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 148-153.

[20] M. -Y. Wu et al., “A PUF scheme using competing oxide rupture with bit error rate approaching zero,” *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 130-132.

[21] K. -H. Chuang, E. Bury, R. Degraeve, B. Kaczer, D. Linten and I. Verbauwhede, “A Physically Unclonable Function Using Soft Oxide Breakdown Featuring 0% Native BER and 51.8 fJ/bit in 40-nm CMOS,” i*n IEEE Journal of Solid-State Circuits*, vol. 54, no. 10, 2019, pp. 2765-2776.

[22] Y. Pang et al., “25.2 A Reconfigurable RRAM Physically Unclonable Function Utilizing Post-Process Randomness Source With <6×10^{-6} Native Bit Error Rate,” *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2019, pp. 402-404.

[23] D. Jeon, J. H. Baek, Y. Kim, J. Lee, D. K. Kim and B. Choi, “A Physical Unclonable Function with Bit Error Rate < 2.3x 10^{-8} Based on Contact Formation Probability Without Error Correction Code,” *in IEEE Journal of Solid-State Circuits*, vol. 55, no. 3, 2020, pp. 805-816.

[24] L. Pantisano and K. P. Cheung, “Stress-induced leakage current (SILC) and oxide breakdown: are they from the same oxide traps?,” *in IEEE Transactions on Device and Materials Reliability*, vol. 1, no. 2, 2001, pp. 109-112.

[25] M.S. Turan et al., “Recommendation for the Entropy Sources Used for Random Bit Generation”, *NIST Special Publication 800-90B, 2018*.

[26] N. A. Anagnostopoulos, T. Arul, M. Rosenstihl, A. Schaller, S. Gabmeyer and S. Katzenbeisser, "Low-Temperature Data Remanence Attacks Against Intrinsic SRAM PUFs," *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 581-585.

[27] A. T. Elshafiey, P. Zarkesh-Ha and J. Trujillo, “The effect of power supply ramp time on SRAM PUFs,” *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 946-949.

[28] S. Skorobogatov, “Local heating attacks on Flash memory devices,” *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2009, pp. 1-6.

[29] Alex Schiffer, The Washington Post website, 2017, How a fish tank helped hack a casino, accessed 10 January 2023, <https://www.washingtonpost.com/news/innovations/wp/2017/07/21/how-a-fish-tank-helped-hack-a-casino/>.

[30] Cisco, Cisco Annual Internet Report 2018–2023 Whitepaper, updated March 9 2020, accessed 10 January 2023 <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>.

Index

Our Thanks

We dedicate this book to our friends, families and, of course, the whole team at **PUFsecurity** that made this project possible.

Together, we gave our time and effort to develop this from only a nascent concept into a fully realized series of books about PUF-based technology. This would have never happened without your continued support and commitment.

So, to each of you, we say thank you!

PUFsecurity Corporation.

8F-1, No. 5, Tai-Yuan 1st St., Jhubei City,
Hsinchu County,
302082, Taiwan
Tel: +886-3-560-1010
www.pufsecurity.com