NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

ARBITER PUF BASED FPGA CHIP IDENTIFICATION AND AUTHENTICATION METHODS WITH ENHANCED RELIABILITY AND MODELING ATTACK RESISTANCE

SIARHEI S. ZALIVAKA

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2018

ARBITER PUF BASED FPGA CHIP IDENTIFICATION AND AUTHENTICATION METHODS WITH ENHANCED RELIABILITY AND MODELING ATTACK RESISTANCE

SIARHEI S. ZALIVAKA

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement fot the degree of

Doctor of Philosophy

2018

Acknowledgements

I would like to express my gratitude to my supervisors for constant support and reasonable criticism of my research efforts. I doubt, that I could finish my Ph. D. study without their inspiration and passion about the research area I have chosen. First, I would like to thank Associate Professor Chang Chip-Hong, who offered me a research scholarship in Nanyang Technological University (NTU). It has been a great experience to work in world-class research group which is recognized all over the world. Prof. Chang has given many insights on critical thinking via research paper reviewing. His experience and expertise in the research influenced great improvement in my research and experimental design skills, paper writing and general knowledge. Then, I would like to thank Alexander A. Ivaniuk, Professor of Belarusian State University of Informatics and Radioelectronics (BSUIR) for inspiring great research ideas, motivational support and technical help in experimental design. His passionate attitude and brilliant mind have helped me to choose directions for my research and experiments.

I would like to express my appreciation to members of our research group both former (Zhang Le, Cao Yuan, Tay Thian Fatt, Sachin Kumar, Zhang Li, Liu Chaoqun) and current ones (Ho Truong Phu Truan, Zheng Yue, Wang Si, Zhang Yufei, Zhong Huishu, Liu Wenye). Beside research work, great discussions and technical help to each other, there were uncountable number of bowling parties, gatherings and other great events which trigger warm feeling in my heart.

Last but not the least I would like to thank my beloved ones for constant support in my Ph. D. journey to Singapore. Thanks to my parents for forming my attitude to life, moral support and inspirational Skype calls and emails. I would also like to express my deepest gratitude to my wife, Yuliya, who has been waiting for me in Belarus for a long time and has changed her usual life schedule to move to Singapore and support me in person.

Abstract

The last 25 years have witnessed an exponential growth of the number of devices connected to the Internet of Things (IoT) from a million in 1992 to 20 billions in 2017. Despite IoT has become widespread, this concept is still not well-established due to several reasons such as lack of standards, security and data protection issues, maintenance cost, etc. Since much of the sensitive personal data is transmitted via IoT devices, secure access control to this data can be highlighted as one of the most important challenges for this area. Classical hardware cryptographic methods have two major disadvantages, significant hardware overhead required for its implementation and non-volatile memory for secret key storage. One effective way to provide secure chip authentication with low overhead is the Physical Unclonable Functions (PUF). They are widely used as a cryptographic primitive to avoid the need for storing the key or secret that can be used to retrieve the device key in the non-volatile memory. PUF uses the intrinsic integrated circuit's manufacturing process variations to generate unique and random response to a given challenge to identify a chip. For reliable key generation, it is required that the responses of the PUF are highly stable against operating environment variations such as temperature and supply voltage variations. One of the most well-explored PUF design is Arbiter PUF (A-PUF), which has been utilized by Verayo to implement RFID ICs as well as by Xilinx to include PUF IP as a hardware root of trust for its new Zynq UltraScale+ devices. However, porting of existing Arbiter PUF designs that are not implemented as ASIC cores into FPGA platform suffers from poor reliability due to routing constraints. On the other hand, improving temporal stability of A-PUF responses makes the circuit vulnerable to modeling attack using machine learning methods. Thus, this research targets design and implementation of reliable and secure A-PUF on FPGA chips without built-in PUF. It also aims to overcome the limitation of using existing PUF IPs for authentication of FPGA-based IoT devices.

This thesis presents a comprehensive overview of state-of-the-art PUF designs and their ASIC and FPGA implementations. As a means for reliability enhancement, a new hybrid PUF based on A-PUF is proposed. Using the SR latch instead of D Flip-Flop as an arbiter makes it possible to expand the original response states to a ternary set stable 0, stable 1 and High Frequency Oscillation (HFO). The enhanced reliability and uniqueness were attested by experimental results implemented on FPGA platform. To further improve its reliability to the ideal 1.0 over a wide range (from -45 °C to +90 °C) of temperature, a challenge classification algorithm is introduced. The proposed method has been tested on identical FPGA chips of two different families and has shown no degradation on uniqueness. To prevent modeling attack, two approaches based on non-linear challenge processing are presented in this thesis. It has been shown that the proposed techniques are resilient against modeling attack by different machine learning algorithms, including the most advanced Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). The abovementioned contributions are utilized to build a low-cost authentication protocol based on a highly accurate model of A-PUF.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

According to Radiant Insights [1], the annual revenue of Field Programmable Gate Array (FPGA) market in 2014 was USD 3.92 billions and the revenue in 2022 is expected to double to USD 7.23 billions. One main reason for the rapid growth of FPGA market is the increasing complexity of Application Specific Integrated Circuits (ASIC) design escalated by the advanced manufacturing process technology. The chip design cost in 28 nm process node is estimated to be around USD 170 millions, which costs twice as much as in 45 nm node [2]. On the contrary, the barrier for entrance into the FPGA market is low and many companies can afford to design their products with advanced process technology nodes without having to invest heavily into the reticle cost, yield loss and development cost.

In recent years, the advent of SoC FPGA has further narrowed the gap in area, power consumption and interconnect delay from ASIC. As SoC designs on FPGA flourish and become increasingly sophisticated, the same common practice of reusing third party Intellectual Property (IP) cores is adopted to improve design productivity. Currently around 67% of all designed and manufactured ICs contain reusable IP cores [3], resulting in an increasingly buoyant IP theft and counterfeit electronics [4]. Reconfigurability and field update are unique and competitive advantages of FPGA, but the transmission, on power-up, of the programmable bitstream from off-chip source also openly exposes the functional definition of the IP cores and increases the risks of loading and executing unauthorized or maliciously tampered IPs. Several anti-counterfeiting methods have been proposed to combat semiconductor IP violations, such as self-destruction [5], obfuscation [6], periodic licensing [7], bitstream encryption [8], IP watermarking [9] and fingerprinting [10], and hardware metering [11]. Although the bitstreams can be encrypted and modern FPGAs are

also able to process the encrypted bitstreams, authentication is not provided to avoid the installation of unauthorized bitstreams in authorized devices or authorized bitstreams in unauthorized devices. The binding of the IP cores to the target devices mandates the use of a unique device key or identifier for bitstream encryption and device authentication. Keeping the device key in non-volatile memory (NVM) is, however, less secure than generating it on-demand by embodying a physical disorder system into the FPGA fabrics.

Physical Unclonable Function (PUF) provides an efficient means to generate chip-dependent signatures. According to Tuyl's definition [12], a PUF is a characteristic of a physical (digital) system which cannot be cloned (copied or reproduced) using another physical system. A PUF can also be described mathematically as a mapping $R = \mathrm{PUF}(CH)$, where CH is an external stimulus (challenge) applied to the PUF and R is an output (response) generated by the PUF. A number of PUFs have been realized in ASIC and specifically in FPGA, including memory-based PUFs (e.g., SRAM-PUF [13], DRAM-PUF [14], Butterfly PUF [15], SR latch PUF [16]), delay-based PUFs (e.g. Arbiter PUF (A-PUF) [17], digital PUF [18]), operational frequency based PUFs (RO-PUF [19], Bistable Ring PUF [20]), etc. Furthermore, FPGA platform creates an opportunity for designers to make use of blocks, which are specific to some particular chip families (e.g., Anderson PUF [21, 22], which make use of standard LUT blocks of Xilinx Virtex-5 FPGA, Monte-Carlo PUF [23] – Xilinx Spartan-6 FPGA, CARRY4 block).

The three dominating FPGA manufacturers Xilinx [24], Intel (former Altera) [25] and Microsemi [26] have recently announced including PUF implementations in their mass product designs for security purpose. Since these companies have occupied around 97% of the total FPGA market [27] in 2016, it demonstrates huge demand in security of their IP cores especially on high-end models, e.g. Xilinx Zynq UltraScale+, Altera Stratix 10 and Microsemi PolarFire. These manufacturers have created a repository of well developed IP cores, namely Xilinx IP Core [28], Altera MegaCore [29], Microsemi DirectCore and CompanionCore [30]. Currently, most PUF solutions are developed for ASIC and embedded in a form of an application-specific standard part (ASSP) in only several FPGA chip families. Therefore, there is great incentive to exploit the uncommitted and more flexible reconfigurable fabric to develop new PUF based solutions directly in FPGA to protect existing IP Cores.

1.1.1 Internet of Things (IoT)

The term Internet of Things (IoT) was coined in 1999. In less than 20 years, it has evolved into an enormous network of more than 6 billion connected smart devices [31]. According to the forcast of Statista Inc., the number of connected smart devices will hit 50 billions in 2020 [32]. This makes IoT one of the most rapid growing networks of widespread applications in smart home, smart factory, intelligent vehicle highway, wearable electronics, remote health care, agriculture, retail and supply chain, etc. IoT utilizes real-time analytics and machine learning methods for data processing and autonomous decision making [33]. This implies that a security failure of a smart device can create a domino effect that literally affects billions of users, leading to huge financial loss, privacy invasion and even life threatening consequences.

As IoT devices are usually resource constrained and have limited power budget, Physical Unclonable Functions (PUF) has gained a foothold for device authentication in IoT [34]. Unlike conventional cryptographic primitives, PUF does not require a persistent presence of secret key in device memory for identification. Instead, a number (usually exponentially many) of random but unique device signatures can be generated by querying an active PUF circuit with different stimuli. This challenge-response mapping mechanism is an inbuilt feature of a physical disorder system by virtue of its unpredictable manufacturing process variability. Even if identical mask is used, due to the random distribution of dopants and lithographic diffraction, the resulting device mismatches will cause each instance of the same PUF circuit to produce a discernibly different response to the same challenge. The response generated by interrogating a PUF can thus be used as a device signature which shares much of the desirable security features as human biometrics. Two of the most important features are: it is impossible to make an exact physical clone of a PUF instance, and it is also probabilistically unlikely to have two PUF instances producing the same response to the same challenge. The former gives PUF its name and the latter endows it with the non-repudiation property for device authentication.

Thus, if IoT devices are used in applications with tight restrictions on data access and small chip size, e.g. medical devices (artificial pacemaker, glucose pumps, etc [35]), weapons, government and industrial networks, special vehicles (police, ambulance, fire force), nuclear power plants control, etc.. Incorporating PUF circuits in IoT solutions for the abovementioned purposes is a good trade-off to afford reasonable degree of security under limited hardware resources. Nowadays, FPGA platform is becoming popular and widely used to implement IoT devices [36, 37, 38, 39], especially in image processing [40, 41], low power applications [42], cloud computing [43],

medicine [44], networks [45] and security [46, 47]. This rapid growth is boosted by increasing revenue of FPGA-based sensors and automotive electronics [48]. This data is also supported by recent release of new FPGA designed for IoT applications by one of the largest FPGA market players (Intel) [49]. Furthermore, Intel's revenue from IoT solutions has been also increased by 20% in 2016 [50].

One of the main advantages of FPGA platform against ASIC is its significantly reduced prototyping time (e.g. for designing a PUF [51]). Therefore, FPGA platform is usually used to prototype digital circuitry and later optimize its area, power and performance on ASIC [52]. Furthermore, FPGA provides a flexible way of testing the design directly and physically in silicon instead of by CAD simulations. This is especially important for PUF as they generate unique device signatures based on manufacturing process variation, which are not predictable in advance [53]. The advantages of FPGA versus ASIC as hardware implementation platform are listed below:

1. **Lower design cost**. According to a study [54], the time to market window for FPGA based designs is 3 to 6 months and for ASIC is 12 to 18 months. Thus, the design time may vary 3 to 6 times due to significant difference in design and manufacturing process turnaround time.

2. **Smaller cost of iteration**. Since FPGA design can be easily and repeatedly implemented and tested directly on physical chip, any mistakes made at design time can be detected earlier and corrected during the design phase. In contrast, ASIC chip can only be tested after manufacturing. The chips cannot be reused if there is any design flaws. Respining a design not only incurs high additional mask cost but also delays the product time to market.

3. **Scalability**. As a result of reconfigurability, FPGA designs can be easily scaled to suit different application requirements and scenarios. E.g., the number of stages of a PUF can be increased to expand the challenge-response space without incurring significant design effort and cost.

4. **Reconfigurability**. FPGA vendors provides a wide range of tools to exploit the flexible and reconfigurable fabric for in-system update, repairs and hardware patching [55]. Modern FPGAs also allow dynamic reconfigurability. PUF designs can fully make use of the reconfigurability and even dynamic reconfigurability by creating multiple states and contexts to increase the difficulty for an attacker to successfully model the challenge-response mappings.

5. **Intra- and inter-die variations of basic FPGA blocks**. FPGAs have been well-known for leading the adoption of new nanoscaled technologies with commercial products. Hence, the building blocks of each contemporary FPGA family carries a rich entropy source inherited from the random device mismatches introduced by the nanoscaled manufacturing process variations. Therefore, every class of FPGA chips can be protected uniquely by designing PUF around their specific embedded LUTs, Flip-Flops, memory, and digital signal processing blocks.

On the other hand, FPGA platform also suffers from a number of disadvantages:

1. **Design space exploration**. ASIC design has much larger design space exploration and can be highly optimized at different levels of design abstraction down to transistor level for chip area, power consumption and performance. Since FPGA platform is based on standard blocks which have already been manufactured, designer has limited design space and less opportunities to perform low-level customization. However, the gap has been closing down nowadays at higher design abstraction level as the FPGA fabrics and building blocks have been highly optimized these days with the help of advanced technologies and powerful electronic design automation tools.

2. **Unable to take advantage of analog and mixed signal designs**. Analog and mixed signal characteristics can be directly and efficiently modeled and utilized in ASIC implementation to build analog circuit PUFs (e.g. voltage transfer characteristic PUF [56], CMOS image sensor based PUF [57], current mirror PUF [58], etc). These parameters are also utilized to improve circuit reliability (including PUF designs [59]). Unfortunately, there are very few families of Field Programmable Analog Array (FPAA) and they are still very application-driven. FPAA contains very limited on-chip resources, which make it much less versatile than its digital cousin.

3. **Constraint in physical design**. Layout design and routing can be completed manually to enhance chip performance characteristics (e.g., design more symmetrical paths in Arbiter PUF [60]). Nonetheless, if a design can be implemented in FPGA with the required characteristics, there is a high probability that it can be ported in to ASIC without significant physical design optimization.

4. **Less cost effective for high volume**. According to a Any Silicon's report [61], if the number of ICs required to be manufactured is less than $4 \cdot 10^5$, ASIC design flow is more expensive than FPGA. Otherwise, ASIC platform is preferred in terms of performance per unit cost for high volume production.

In summary, there are potential room to improve the design of a digital PUF in FPGA platform considering the cost and convenience of carrying out in-system update, repair and experimenting with real silicon data with high scalability. Morever, if a good practical solution is found, the design can be ported to ASIC with less initial cost in the design phase. On the other hand, users of advanced FPGA are able to use on-board PUF implementations. However, according to Xilinx annual report [62], the company sells tenfold more core chips than advanced ones. Unfortunately core FPGA chips usually do not have embedded PUF and different PUF designs are to be implemented to meet different application requirements for security reasons. Therefore, there is a demand for FPGA based implementations of PUF design especially in low-cost IoT solutions.

Recent new releases and technology broadcasts indicate the growing interest in designing and implementing solutions for IoT in FPGA [63, 64]. This is in part due to efficiency is sufficient for these tasks and the design cost is much lower. Since the security challenges in IoT are not well addressed, good designs of PUF for FPGA implementation are expected to attract increasing focus and attention.

1.2 Research Objectives

This research is driven by delay based strong PUF implementation on FPGA platform. Recent works have shown that implementation of Arbiter PUF (one of the most known delay based strong PUF designs) suffers from low reliability if the routing is performed automatically by the tool [65]. On the other hand, if Arbiter PUF output values are stable, it can be easily predicted as each stage controlled by a challenge bit can be independently characterized and related mathematically by a linear delay model [66]. Therefore, the target of this work is *to devise a different delay based Arbiter PUF for FPGA implementation to overcome these major weaknesses by simultaneously possessing high reliability and resistance against modeling attacks.* This design can be practically utilized in authentication protocols by IoT devices. Therefore, to achieve the aim of this thesis, the following fundamental issues have to be addressed.

1. *Lack of fine placement and routing control* FPGA implementation of Arbiter PUF suffers from limited control of path delay due to asymmetry of paths generated by automatic placement and routing tools provided to a designer. Even with manual routing, it is hard to implement two unbiased symmetrical paths. This is because a successful implementation of aribiter PUF requires a difference in path delay that is contributed solely by manufacturing process variation instead of design-induced bias. If the delay bias induced by physical design is alongside with that by the manufacturing process variation, the uniqueness and randomness will suffer. If the design bias counteracts the effect of process variation, the delay difference of the two paths may fall within the aperture time of the arbiter circuit (D Flip-Flip in classical design). Metastability is the main cause of the poor reliability of arbiter PUF.

2. *Hardware cost in reliability enhancement* While error correcting code (ECC) can be used to improve the reliability of PUF in general, the hardware overhead is considerably high and impractical when the reliability of the raw PUF response is low. Since the reliability of Arbiter PUF is mainly contributed by metastability of arbiter circuit, the susceptibility of induction of metastable state by different challenges can be further tested using the a priori knowledge about linear additive PUF model. It has been shown that small changes in delay difference value due to environmental variations has low probability of flipping a response bit. This fact is considered in our approach to achieve a perfect reliability over a wide temperature range without resorting to expensive ECC.

3. *Conflict between reliability and unpredictability* On the other hand, if the PUF is made more reliable by filtering away the unstable responses, they can become more predictable. Significant reduction of challenge-response space to improve the reliability of Arbiter PUF can increase its modeling attack vulnerability. There are two approaches to address this problem. One approach is to modify the basic bit-slice design of a PUF itself. This method works until a new parametric model that relates its challenge and response mapping is discovered by an attacker. Then a machine learning algorithm can be derived to obtain the unknown parameters of this new mathematical model from known challenge-response pairs. The second approach is to complicate the known model by additional circuitry to obfuscate linear dependency between challenge and response values. According to recent research, both approaches do not completely

solve the problem [67]. However, obfuscation approach increases the difficulty and cost of an attacker, which may require for example larger set of challenge-response pairs for training, greater computational power and more complicated machine learning algorithms. It is difficult to evaluate the increase in effort of bit-slice modification of PUF as it requires a new mathematical model instead of existing model to break. It is, however, generally believe that the increase in resistance by such approach mandates a higher hardware cost.

4. *Attacks on authentication protocol* Recent modeling attacks utilizing Evolutionary Strategy [68] make use of correlation between the challenge-response transmitted during authentication. Despite "raw" values of challenges and responses are not sent from device to server and vice versa, the PUF model can be derived from the helper data, as well as the noisy response values caused by its low reliability. Unfortunately hand, using only carefully selected set of reliable challenge-response pairs (CPR) can also be exploited for building an accurate Arbiter PUF model [69]. Therefore, two requirements are instrumental for devising a secure authentication protocol based on strong PUF. First, the true challenge and response values have to be processed internally and refrained from transmitting openly during authentication. Second, exclusion of unreliable CRPs should not be done unconditionally as this will provides additional information to the attacker.

1.3 Contributions

The abovementioned research problems and objectives have been addressed by the following contributions: *1) Metastability detection technique for Arbiter PUF; 2) Reliability enhancement technique for Arbiter PUF based on challenge classification; 3) Low-cost fortification against modeling attack for Arbiter PUF; 4) Machine learning attack resistant authentication protocol.*

1.3.1 Metastability detection technique for Arbiter PUF

Proposed technique is designed to address the response instability problem caused by the metastability of delay flip-flop encountered in the FPGA implementation of Arbiter based Physical Unclonable Function (A-PUF). This thesis presents a new multi-arbiter approach to extract more entropy to extend the number of response bits to a single challenge. New multi-arbiter schemes based on the insertion of either

a four-flip-flop arbiter or SR latch arbiter after each pair of multiplexers in the configurable paths are proposed to detect the metastable state when two copies of test pulse arrive at the arbiter inputs almost simultaneously. The detected metastable states are distinguishable by the encoded multiple valued outputs of the arbiter. The codes corresponding to the metastable states collectively form a deterministic ternary state that can be recoded to one of the stable states to improve the uniqueness and reliability of the PUF. Our analysis shows that the proposed design can generate robust and reliable challenge-response pairs with a uniqueness of 0.4982 and a reliability of 0.9985 at the expense of a relatively small FPGA resource overhead (Chapter 3). This contribution is intended to improve reliability, uniqueness and randomness of classical A-PUF design.

1.3.2 Reliability enhancement technique for Arbiter PUF based on challenge classification

This work presents an expanded model for the propagation delay difference of A-PUF structure, which can be utilized to avoid responses with poor reliability without additional hardware overhead. The proposed delay model enables the classification of challenges into stable and unstable, and their proportions are device dependent and characterizable. The stable challenges can be further sifted to achieve a perfect reliability of 1.0 within the full operating temperature range of the target FPGA boards from -40° C to +90° C. Ideal uniqueness of 0.5 is obtained from the responses generated by the fully stable challenges filtered by our proposed algorithm. Despite the shrinking challenge-response space, our experimental results show that all 200 IDs generated for each ID length from 16 to 128 bits are unique with normalized hamming distance between IDs fall within the range of [0.48, 0.50]. The identified unstable challenges have also been analyzed to possess richer entropy, which is useful for random number generator (Chapter 4). This contribution is intended to improve the reliability and uniqueness of classical A-PUF design.

1.3.3 Low-cost fortification against modeling attack for Arbiter PUF

A-PUF with exponential number of challenges is an ideal candidate to realize lightweight and robust device authentication in Internet of Things applications. Unfortunately, it is particularly difficult to attain highly reliable responses and increase its modeling

attack resistance simultaneously. An approach is proposed to reduce the vulnerability of A-PUF to machine learning attacks without compromising its high reliability and uniqueness. It utilizes a multiple input signature register (MISR) to process the input challenges. Our experiment results show that the accuracy of predicting the responses of a MISR augmented 128-stage arbiter PUF in FPGA implementation by support vector machine and gradient boosting learning algorithms with a training set of 100,000 challenge-response pairs has reduced drastically from 98% to 50%. If design-for-testability is mandatory, the MISR can be reconfigured from an existing built-in logic block observer, making this approach virtually free from hardware overhead. Otherwise, the MISR carries a negligible hardware overhead of only 0.4% of the total available resources in an Xilinx ZC706 FPGA chip (Chapter 5, Section 2, 3 and 7). This contribution is intended to improve the machine learning attack resistance without compromising reliability, uniqueness and randomness of classical A-PUF design.

1.3.4 Machine learning attack resistant authentication protocol

An authentication protocol is proposed based on strong PUF, namely Arbiter PUF implemented in FPGA. The reliability of this proposed A-PUF implementation achieves reliability score of 1.0 utilizing three techniques, metastability detection, challenge classification and error correction. These three methods operate in tandem to reduce the hardware overhead required to improve the security in comparison to employing a classical A-PUF implementation. The high level of reliability enable a 100% accurate model of A-PUF to be constructed and resided on the server to avoid the need for transmitting the true values of challenges and responses during authentication. Furthermore, the challenge and response values are processed by MISR circuit which is implemented as Build-In Logic Block Observer (BILBO). Since MISR transformation is lossy, it cannot be recovered uniquely, which protects the CRP set from collected by the attacker. Several types of attacks namely modeling, random guessing, seed compromising, replay, man-in-the-middle and PUF instability are evaluated. Proposed protocol is practically secure against these attacks, i.e. attacker is required to have exponentially large number of CRPs and huge amount of computing power in order to break it (Chapter 5, Sections 4, 5 and 6). This contribution is intended to improve reliability and machine learning attack resistance without degradation in uniqueness and randomness of classical A-PUF design.

1.4 Organization

The remaining content of this thesis is organized as follows:

Chapter 2 briefly describes the background of this study, i.e. hardware metering methods which are applicable to protect IP cores against illegal usage and copying. This classification includes PUF, which can be categorized as non-functional passive metering methods for unclonable identification. The brief overview is followed by a description of PUF figures of merit, reliability, uniqueness, randomness and machine learning attack vulnerability. Classical PUF designs (Arbiter PUF, Ring Oscillator PUF and SRAM-PUF) and Bistable Ring PUF are reviewed and compared by hardware overhead, challenge-response space and PUF quality metrics. As a result of comparison Arbiter PUF is considered as a promising candidate to be used as a security primitive in FPGA designs.

Chapter 3 is devoted to several techniques for metastability detection in Arbiter PUF, which are implemented in FPGA to improve basic reliability of classical A-PUF design. Brief description of Multi-Arbiter PUF design and its figures of merit evaluated based on path length. This chapter describes two metastability detection techniques and their circuit implementations, namely 4-DFF based and SR latch based arbiter, with enriched A-PUF response alphabet. Proposed implementations are tested on reliability, uniqueness (classical and Sokal-Michener based) and randomness. These techniques are compared with classical DFF based implementation of A-PUF on hardware overhead. The implemented multi-arbiter PUFs are analyzed by correlation hypothesis testing to determine significantly correlated pairs of arbiters.

Chapter 4 presents a challenge classification algorithm based on A-PUF mathematical model, which takes into account metastability state. The metastability state is further analyzed based on simulation results to locate the susceptible region based on asymmetry of Arbiter PUF design. The proposed algorithms is able to classify challenges into stable and unstable according to its sensitivity to changes of the delay difference value. This method achieves reliability score of 1.0 over temperature range from -40° C to +90° C. The filtering of challenge-response pairs does not affect other basic quality of Arbiter PUF design. The identified stable CRPs can be used for FPGA ID generation algorithm while the unstable ones can be utilized to generate true random number sequences.

Chapter 5 introduces three challenge obfuscation methods which are designed to protect A-PUF against modeling attack. The simplest method is by applying a SHA-

256 hash function to a challenge value to make the dependency between challenge and response non-linear. An obvious disadvantage of this method is its dramatic increase in hardware cost which makes it unsuitable for lightweight authentication. Two other methods based on T Flip-Flop (TFF) register and Built-In Logic Block Observer (BILBO) are described and analyzed in terms of hardware overhead and security. This chapter reports a technique based on A-PUF with enhanced reliability, its 100% precise mathematical model and BILBO block are used advantageously to prevent machine learning attack. The three authentication protocols built around BILBO module are used to obfuscate linear dependency between challenge and response values and avoid transmitting the "raw" challenge and response values. Chapter 5 is concluded by an analysis of several attacks including modeling and random seed compromising.

Chapter 6 summarizes and concludes the major contributions of this thesis and outlines a few promising directions for future works based on the presented research.

Chapter 2

Background

2.1 Hardware Metering Taxonomy

Hardware metering has two main classes, passive and active [70]. Passive metering provides a means to massively track a unique IC or components of the IC tagging. The term "passive" means that the IP owner can only check if a product is original or not, but he is not able to change or control its functionality. Such methods are also classified as *passive nonfunctional identification* to differentiate them from some methods that can uniquely mark the function of a chip or generate unique identifier based on some functions of the IC. The latter group of methods is called *passive functional identification*. These methods are also passive, because they utilize the IC functionality, but leave its function accessible regardless of the authenticity of the chip. These two groups can generate both reproducible and non-replicable identifiers. Using unclonable identifiers generated by a PUF is a very effective way to prevent the IC from direct cloning.

Unlike passive hardware metering, active methods provide the legal owner or recipient of the IC to actively monitor, enable and disable the IC functionality to prevent unauthorized usage of an IC. Therefore, active metering is close to a functional identification by utilizing the chip functionality as a locking mechanism. However, active metering approaches make the functionality dependent on some chip tagging information (reproducible or non-replicable identifiers). This group of methods is called *internal active hardware metering*. The other group of methods relies on some external circuits, which are used for the hardware implementation of cryptographic algorithms. This class of approaches is called *external active hardware metering*. Both groups utilize the unique identifiers that are either physically clonable or unclonable.

The above taxonomy for hardware metering is depicted in Fig. 2.1.

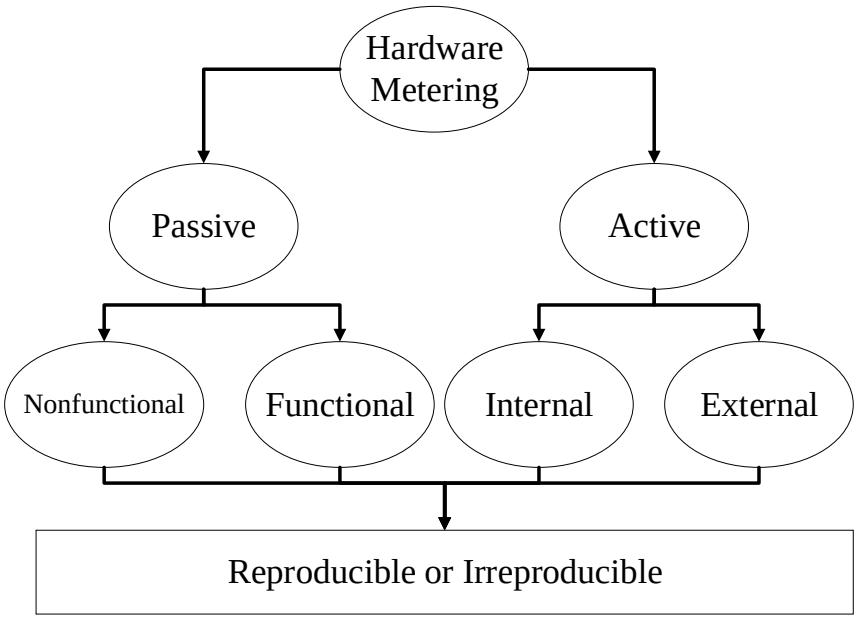


Figure 2.1: Hardware metering taxonomy.

2.2 Passive IC Metering

2.2.1 Nonfunctional Identification

2.2.1.1 Reproducible Identification

Initially one of the most natural and simple way to protect an IC against counterfeiting was embedding (carving) serial numbers into a chip structure. Semiconductor devices were not as widespread at the beginning of the epoch until the last two to three decades. So this countermeasure was quite effective against the forgery. Nowadays embedded serial numbers are still used as a first level of security against amateur chip counterfeiting. It still makes sense, because an average user is not even able to extract this serial number without destroying the chip packaging.

The next generation of identifiers was the digitally stored serial numbers. IC designers reserve special read-only registers for storing chip ID without having to physically carve them onto the semiconductor. Unlike the embedded serial numbers, digitally stored IDs require the attacker to have some general IC design knowledge to steal the secret. Therefore, this approach is more complicated and claims to require more effort for a successful counterfeiting.

Both methods are quite basic and easy to overcome. Embedded serial numbers can be modified or cloned through replicating the chip by rogue chip manufacturer. Therefore, this approach is not able to prevent the theft of legal serial numbers and use them as a genuine identification for a directly cloned semiconductor devices. Digitally stored numbers are better protected against direct IC copying and require cloning the design or special ID registers. However, it is no longer a challenge for the attackers today to replace or remove these securely stored or hard-coded serial numbers with relatively small effort. More sophisticated or state-of-the-art protection methods are required to combat the more advanced piracy and counterfeiting.

2.2.1.2 Unclonable Identification

To overcome the abovementioned limitations, several PUFs were introduced by Pappu [71] which can be designed by silicon-based devices [72]. The basic idea of PUF implementation is to utilize the intrinsic stochastic properties of the IC to generate chip-unique, unclonable and random signatures for device authentication and electronic tagging. A PUF can be considered as a physical system with an external stimulus, which is called a *challenge* and an output, which is called a *response*. The response is produced as a result of feeding the challenge to the PUF. The PUF can be fully described by a set of challenge-response pairs (CRPs). One key characteristic of the PUF is that it is impossible to build a mathematical model that can generate exactly the same set of CRPs without involving the chip.

There are many characteristics of the physical systems which can be used for a PUF implementation. The first PUF that used laser beam directed with a particular angle to generate a unique light pattern was introduced in [71]. The operation frequency difference of ring-oscillator circuits caused by manufacturing process variations was proposed by Gassend [72] as a source of unique signatures. The direct delay-based PUF was introduced by Lee [60] who used propagation delays of two racing signals as the basis of a PUF response generation. The IC coating was also used for authentication and identification purposes by Tuyls [73]. The reset state of memory cells was utilized in several types of PUF (SRAM-PUF [74], DRAM PUF [14], Butterfly PUF [15], SR latch PUF [16]). Zhang [75, 76] showed possible ways to design PUFs based on emerging memory technologies (e.g. Spin-Transfer Torque and Phase Change Memory). The fixed pattern noise of CMOS image sensor was used by Cao [57] to generate PUF CRPs at the sensor level. Approaches to the implementation of PUF are not limited to the abovementioned, but they are good enough

for an insight into the diversity and rich sources of parametric mismatches due to manufacturing process variations that can be utilized to design a PUF.

In term of its applicability in authentication protocols and attack resilience, two different types of PUF have been defined based on the sizes of their CRP space [13]. When the number of CRPs growth exponentially along with a linear growth of PUF parameter, the PUF is classified under *strong PUF*. The CRPs of the strong PUF can be used directly and conveniently for authentication purpose without specific peripheral protection, because the probability of guessing a correct combination is rather small (2^{-k}, where $k \approx 100$ is the dimension of the PUF parameter). On the other hand, strong PUF is susceptible to machine learning attacks due to the large number of CRPs that can be collected and learnt by the attackers to infer the unknown CRPs. On the contrary, *weak PUFs* have a rather limited number of CRPs. Therefore, it cannot be used directly in an authentication protocol without some form of protection for its internal challenge and response to avoid the CRPs from being extracted by the attackers.

Although PUF had provided a significantly enhanced security for chip identification and secret key generation, they have some vulnerabilities that need special attention. For example, weak PUFs need to be safeguarded against the side-channel attacks as they are usually used as entropy source for secret key generation in hardware implementation of cryptography algorithms to avoid the need to store the key in non-volatile memory (NVM). Strong PUFs are vulnerable to modeling attacks [77] as they are widely used to simplify secure communication in cryptographic protocol where the external CRPs used in the information exchanges may be eavesdropped or intercepted.

2.2.2 Functional Identification

The first known functional metering approaches were based on the design of a unique control path for each chip. However, if the ICs are fabricated from the same mask, they will have the same control path and this approach is not useful for the purpose of preventing over-production. Therefore, it was proposed to design a single datapath controlled by multiple variations of the same control path [78, 79]. To implement this technique, a small part of a chip needs to be left programmable so that the control path can be reprogrammed after fabrication. To achieve an adequate variety of control paths, it was proposed to permute variables assigned to some registers. Since a particular permutation is programmed post-silicon, it provides a unique way

to redesign the control path. It was demonstrated that multiple permutations can be made with relatively low hardware overhead.

The other functional metering approaches are based on the replication of states and transitions with respect to the design constraints [80]. The state-transition graph (STG) is recolored with added states and transitions to fulfill this requirement. Another solution is based on embedding a test machine into the device FSM [81]. The identifier embedded into the design can be detected off-chip and the hardware overhead is relatively low (less than 5%).

However, the above techniques have two major disadvantages. First, the identifiers do not link with a controlling mechanism to lock and unlock the design; Second, the identifiers can be easily cloned, because they are generated according to the function of the design. Therefore, these approaches cannot prevent direct cloning of the design, when the attacker (through insider or chip manufacturer, for example) has an access to the design files for fabrication.

2.3 Active IC Metering

In addition to providing a unique IC tagging, active metering also provides a locking mechanism to allow activation, deactivation and control of the chip's activities after manufacturing. To implement these augmented features in hardware, it is required to have the functionality of the chip dependent on some unclonable and unique chip's identifier. The most suitable component, which can meet this requirement is none other than the PUF.

The general Active IC metering flow [82] is shown in Fig. 2.2. Two parties are involved in this business model: a design house (left side of Fig. 2.2), who owns the IP rights for the fabricated semiconductor circuits and a foundry (right side of Fig. 2.2), who manufactures the contracted number of ICs according to the design provided by the IP owner.

The final chip design and fabrication processes consist of the following steps:

1. The designer creates a high level description of the design with hardware description language (HDL). To lock the functionality, the designer is required to extract the FSM and find appropriate places for embedding a locking mechanism.

2. After the FSM analysis, the designer extends the original FSM to a boosted FSM (BFSM) through states and/or transitions replication.

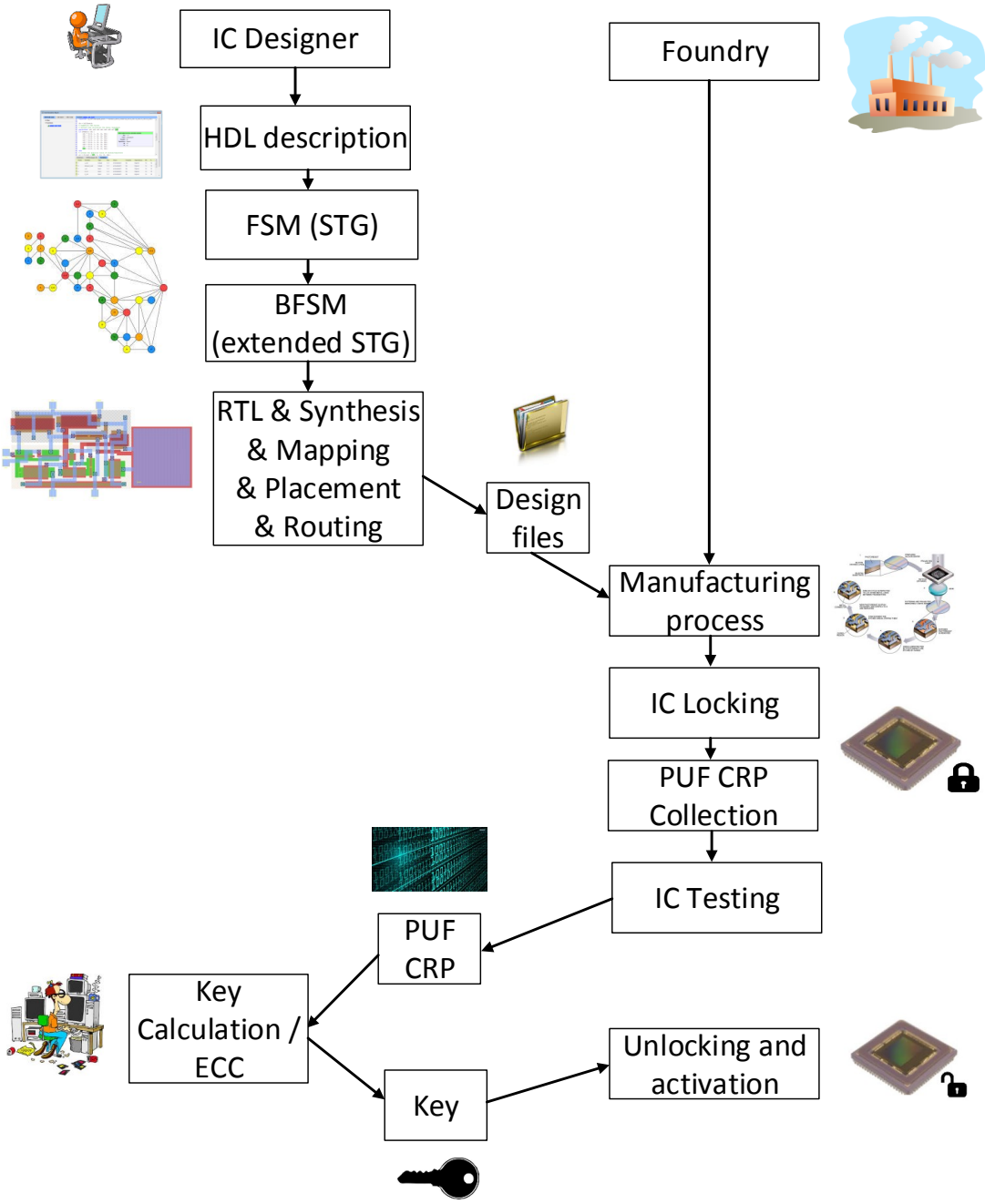


Figure 2.2: The general active IC metering flow.

3. After the locking function has been embedded into the design, the IP owner will carry out the usual IC design routine (RTL, synthesis, mapping, layout, placement and routing, and design verification).

4. The design house transfers the design files and test vectors to the foundry for manufacturing.

5. The foundry fabricates the contracted number of semiconductor chips. Each IC already contains an unclonable ID, which is usually implemented as a weak PUF with Error Correction Codes (ECC) added to improve its reliability. If the foundry that produces the chips is outsourced by the design house, there is a great risk that the untrusted foundry may overbuild the chips and sell them in the "grey" market. Therefore, the IC function has to be locked before it is sent for fabrication and can only be unlocked by design house or legal buyers of the chips after the chips have been manufactured and returned to the design house.

6. The manufacturer collects the CRP from the unclonable identifier unit and tests the fabricated IC.

7. The foundry sends collected CRPs to the design house. Since BFSM is dependent on the CRP values, the key for unlocking the design can be computed only when both the BFSM structure and the CRPs from the unclonable identifier of the chip are known.

8. Finally, the design house uses the key or provides the key to the legal chip buyers or IP licensers to unlock the fabricated chip. This key is not applicable for the other ICs manufactured from the same design as the BFSM structure depends on the CRPs, which are unique for each fabricated chip.

Thus, Active IC metering approach can protect an outsourced chip design against overbuilding.

2.3.1 Internal Active IC Metering

Internal methods of active metering utilize the FSM of an IC to obtain a locking mechanism through the states and transitions of its corresponding STG [82, 83]. Assume that the original FSM of an IC has S states, which can be implemented with at least $F = \lceil log_2(S) \rceil$ Flip-Flops (FFs).

If the FSM is extended to a BFSM by adding S' complementary states, the new FSM will have $F'' = \lceil log_2(S+S') \rceil$ FFs. Additional transitions should also be inserted to provide the connectivity of the new STG, i.e., every additional state should be reachable from some existing states. Let the number of additional FFs be denoted as $F' = F'' - F$. With the number of additional FFs grows only logarithmically with the number of states, it is affordable to set $S' \gg S$.

The design should contain a PUF to generate a random code, which is actually the response to some fixed challenge. The PUF determines which state will be chosen as an initial state. The BFSM requires $F'' = \lceil log_2(S + S') \rceil$ FFs for a proper operation, so the PUF response should contain F'' bits.

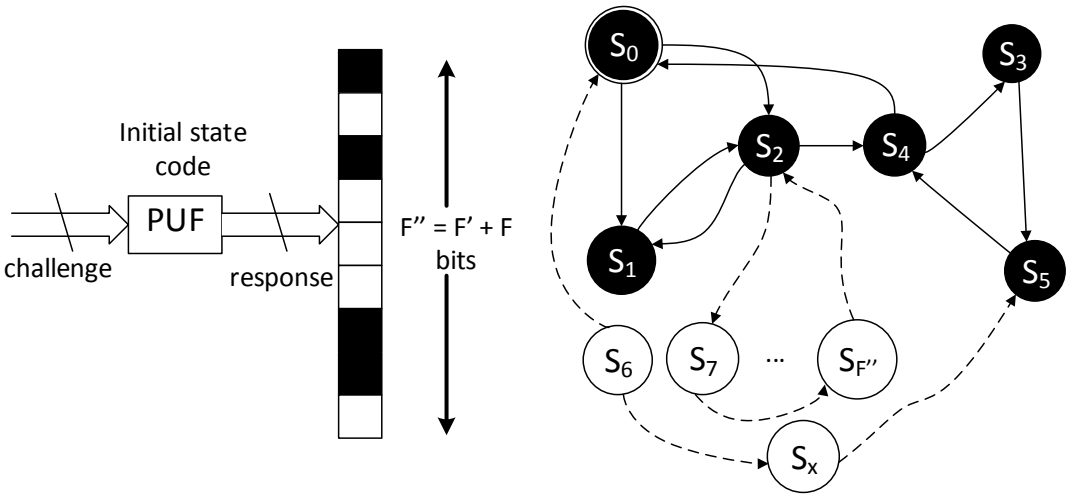


Figure 2.3: Active metering technique based on addition of fake states.

When a chip is powered up, the PUF generates initial state code (see Fig. 2.3, where state S_X is an initial state). The original FSM states are shaded and the additional states are left unshaded. All the extra states are considered as the dead end states because if the customer does not know the structure of the BFSM, it is impossible to leave these states to reach the correct reset state. If the initial state lies among the original FSM states, there is only one chance to reach the reset state. However, the probability of this is relatively small as $P = \frac{2^F}{2^{F''}} = \frac{1}{2^{F'}} = 2^{-F'}$ and $2^{F'} \gg 2^F$.

To prevent a successful random guess from obtaining the right combination to reach the reset state from the added states, the BFSM is designed in such a way that there is only one valid route from each complementary state to the reset state.

Therefore, without knowing the real structure of the BFSM, the attacker will not be able to take control over the manufactured IC. The structure of the BFSM is the design house's secret and cannot be transferred with the design files. The route from the initial state to the reset state is the key combination, which is unique for each manufactured IC because the responses of the PUF to the given challenges are random and chip-dependent.

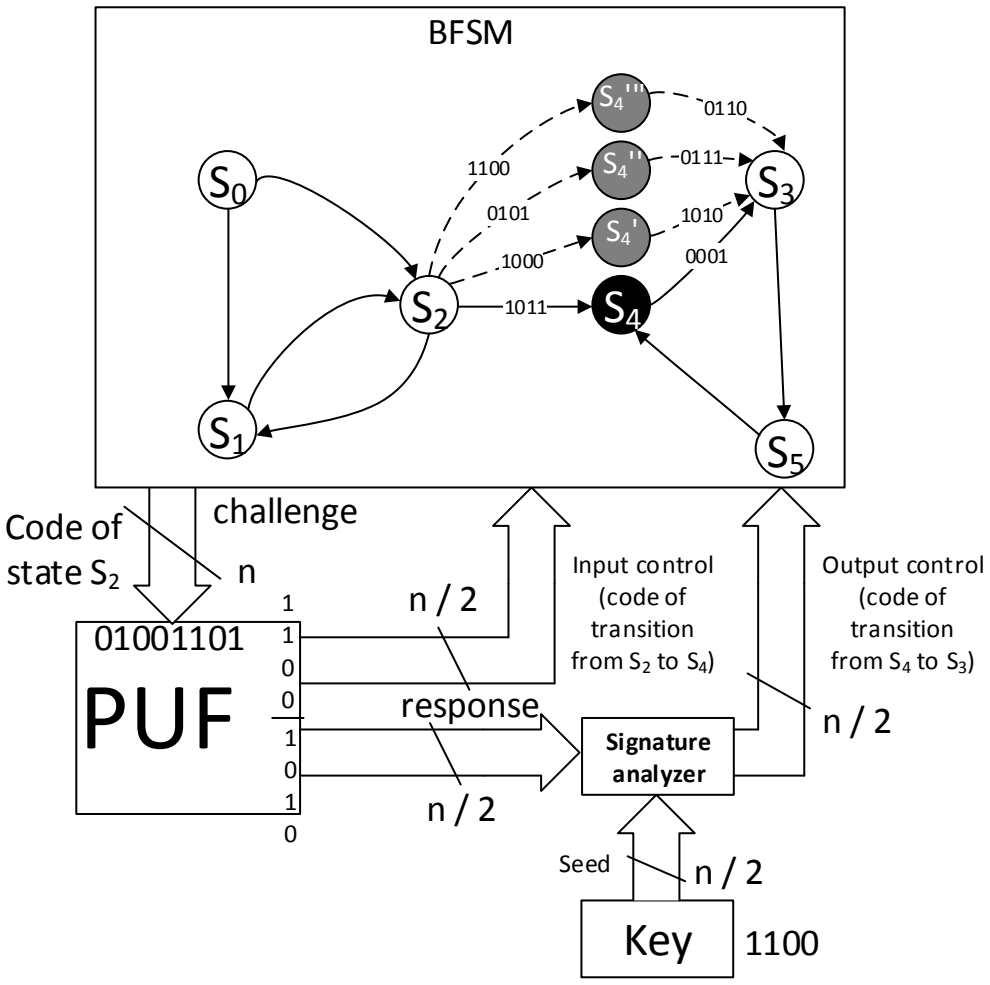


Figure 2.4: Active metering technique based on replication of existing states.

Another internal active metering approach was proposed in [84]. Unlike [82], only a few states of FSM are replicated. Assume that state S_i is selected for replication, then all transitions from and to this state are copied to the replicated states S_i'. The PUF challenge is the code for the initial state and a part of the PUF response is used

as a transition code to the replicated state. Thus, after powering up the chip, the FSM will be in a replicated state. To leave this state, an output transition is to be generated based on a second part of the PUF response and a key provided by the designer. The PUF response and secret key are fed to the signature analyzer and finally the code for a proper output transition will be generated.

An example of this active metering method is shown in Fig. 2.4. Let the PUF challenge be "01001101". Based on this challenge, S_2 is chosen as an initial state of the FSM. Then according to the first part of the PUF response ("1100"), transition to state S_4''' will take place. Assume that a signature analyzer simply carries out an XOR operation. To effect a transition from state S_4''' to an original state S_3, the key required is "1100". By XORing the second part of the PUF response "1010" with the key "1100", a valid code "0110" is obtained for the required transition. It should be noted that the length of the PUF response should be long enough to prevent a brute force attack, i.e., the probability of guessing the correct transition code should be 0.0001 or lower. This approach requires much less hardware resources, but larger PUF response and secure signature analyzer are required.

2.3.2 External Active IC Metering

An external active IC metering uses asymmetric cryptographic algorithms to lock the IC, so it requires a particular external secret key to perform the operation. This method was first proposed in [85] and was named EPIC after the Acronym for Ending Piracy of Integrated Circuit. The general structure of this approach is shown in Fig. 2.5.

According to the public key cryptography protocol, the designer has to generate a private and public key pair (master keys, or MKs). The private master key (MK-Pri) shall be kept secret and never be sent to another person. Each fabricated IC should generate its own pair of keys using an embedded PUF. A cryptographic locking mechanism is embedded in the register transfer level (RTL) of the design.

The locking mechanism of EPIC was implemented using XOR gates in addition to several noncritical wires and a common key (CK) register. A valid CK will enable the original functionality of the IC, but when CK is incorrect, the behavior of the chip will be changed . To prevent collusion by in-house designers, CK is a random number generated by a PUF. Thus, each IC can only be unlocked by the CK generated by the PUF after the chip has been manufactured.

To activate a chip, the foundry has to generate a pair of random chip keys (RCKs) private (RCK-Pri) and public (RCK-Pub) using the embedded PUF. The designer

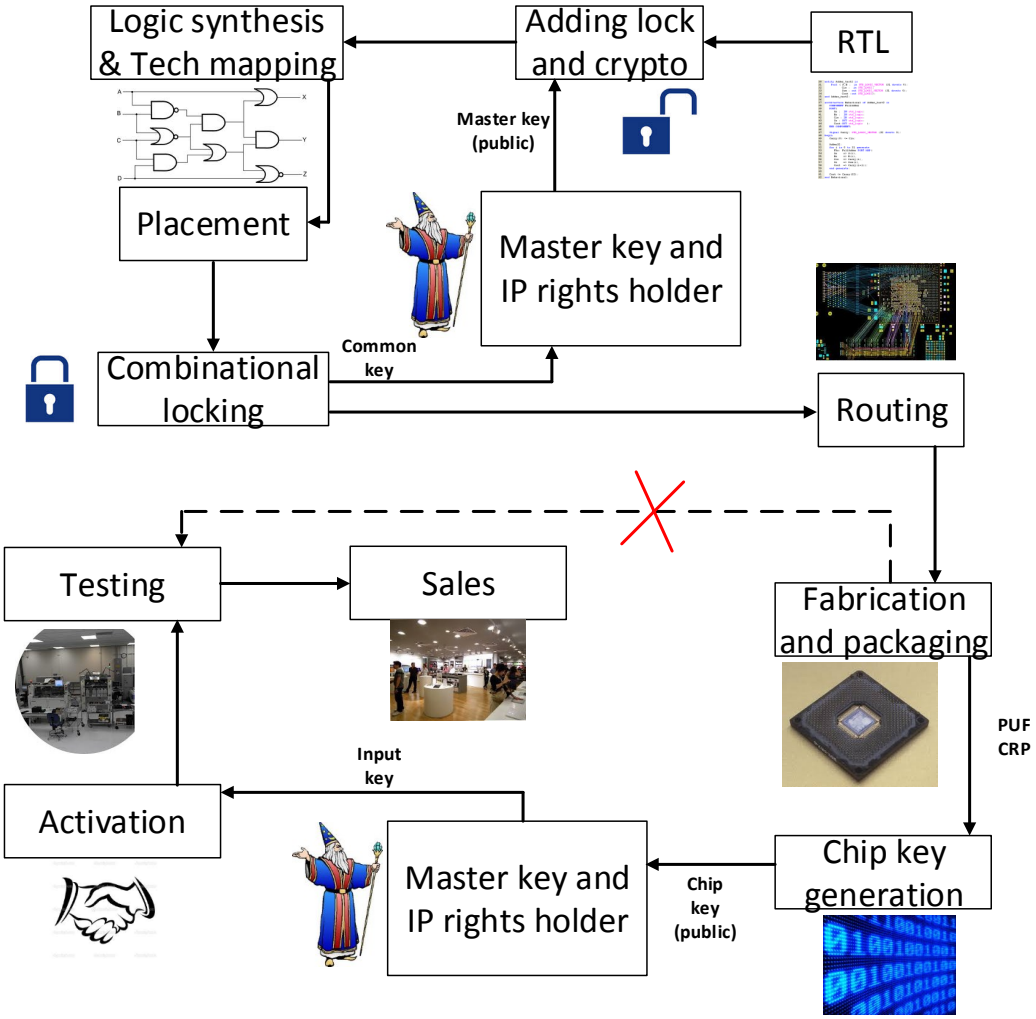


Figure 2.5: Ending Piracy of Integrated Circuits (EPIC). General structure.

can authenticate the foundry by the signature signed by the RCK-Pri before sending the foundry an input key (IK), which consists of CK encrypted with the RCK-Pub and signed by the private MK of the designer. The manufactured chip can then decrypt IK using the RCK-Pri and the public MK. The recovered CK will be applied to unlock the logic of the chip. Thus, the manufacturer can help to test the chip and sell it to the end customers.

Since all keys were generated by the internal PUF, they are unique for each fabricated IC and inapplicable for other copies.

2.4 PUF figures of merit

Almost all advanced hardware metering methods are based on PUFs. Therefore, high quality PUF design can have many applications in different approaches to protect IP cores against counterfeiting. Reliability, uniqueness, randomness and machine learning attack resistance is usually used to evaluate the quality of a PUF design.

2.4.1 Reliability

The reliability of a PUF measures the temporal reproducibility or stability of the responses to the same input challenge. The reliability can be quantified using the Bit Error Rate (BER), which is the average percentage of erroneous response bits obtained at different periods of time or operating environments. Let R_i be a reference response vector of size N to some challenge CH produced by a PUF instance. Each element of R_i can be computed by a majority voting algorithm as follows:

$$R_i = \frac{max(n_0, n_1)}{n_0 + n_1} \tag{2.1}$$

where n_0 and n_1 are the numbers of zero and one responses, respectively in E tests, and $n_0 + n_1 = E$. Alternatively, value of R_i can be determined as a response value during the first test ($e = 1$).

Let $R_{i,e}$ be the response to the same challenge generated by the same PUF at different time $e = 1, 2, \ldots, E$. The reliability S can be computed by [86]:

$$S = 1 - BER = 1 - \frac{1}{E}\sum_{e=1}^{E}\frac{HD(R_i, R_{i,e})}{N} \tag{2.2}$$

If $K \in [1, 2^N]$ challenges CH^{Ω_i} (i = 1, 2, ..., K) are applied to a PUF, then the average reliability can be calculated by:

$$S_{avg} = \frac{1}{K}\sum_{i=1}^{K} S(CH^{\Omega_i}) \tag{2.3}$$

where $\Omega = \sum_{i=0}^{N-1} 2^i \cdot CH_i$ is the decimal representation of an input challenge $\{CH_{N-1}, CH_{N-2}, \ldots, CH_0\}$.

The minimum reliability can be calculated by:

$$S_{min} = min(CH^{\Omega_1}, CH^{\Omega_2}, \ldots, CH^{\Omega_K}) \tag{2.4}$$

Ideal PUF should have a reliability score of 1. However, this is almost impossible to achieve from the raw responses without applying additional post-processing techniques.

2.4.2 Uniqueness

The uniqueness quantifies the dissimilarity of the responses generated by different PUF-instances to the same challenge. Average inter-die Hamming Distance (HD) can be used to estimate the uniqueness. Let R_u and R_v be the N-bit responses generated from two different PUF-instances u and v respectively to the same challenge CH. The uniqueness U for m PUF-instances can be computed by [86]:

$$U = \frac{2}{m(m-1)} \sum_{u=1}^{m-1} \sum_{v=u+1}^{m} \frac{HD(R_u, R_v)}{N} \tag{2.5}$$

Ideally, $U = 0.5$, which is the maximum percentage of bit differences that can be obtained between both the true and complement forms of two binary vectors.

2.4.3 Randomness

Common way of testing randomness of PUF responses is Shannon's entropy [87]. Let R be the N-bit response of a PUF, and p_0 and p_1 be the probabilities of zero and one bits, respectively in a response vector. These probabilities can be estimated as:

$$p_\alpha = \frac{k_\alpha}{N}, \alpha \in \{0, 1\}. \tag{2.6}$$

Then Shannon's entropy $H(R)$ can be evaluated by:

$$H(R) = -\sum_{\alpha \in \{0,1\}}^{\alpha} p_\alpha \cdot \mathrm{Log}_2 p_\alpha. \tag{2.7}$$

Despite entropy estimates randomness quite well it does not take into account mutual dependencies, patterns, correlations between symbols in random sequence of bits generated by PUF, which are important in cryptographic applications. Therefore, a more generally adopted way to assess randomness is NIST package [88], which contains 15 statistical tests to determine if a generated sequence fulfills key statistical properties possessed by a truly random one. These tests are following:

1. Frequency (Monobit) Test.

2. Frequency Test within a Block.

3. Runs Test.
4. Test for the Longest Run of Ones in a Block.
5. Binary Matrix Rank Test.
6. Discrete Fourier Transform (Spectral) Test.
7. Non-overlapping Template Matching Test.
8. Overlapping Template Matching Test.
9. Maurer's "Universal Statistical" Test.
10. Linear Complexity Test.
11. Serial Test.
12. Approximate Entropy Test.
13. Cumulative Sums (Cusum) Test.
14. Random Excursions Test.
15. Random Excursions Variant Test.

There are multiple other packages, which are essentially testing the same characteristics of a generated sequence (e.g. Diehard [89], TestU01 [90], Dieharder [91]). A PUF is considered to have good randomness if a significantly long sequence of responses generated by it pass all the tests of a true random number stipulated by one or more of these statistical test packages.

2.4.4 Machine learning attack resistance

Since most of the PUFs have binary response alphabet $\{0, 1\}$, machine learning attack can be considered as a binary classification task. There are well-known metrics for quality assessment of classifiers, namely accuracy, precision, recall and F-score [92]. There are four possible outcomes of binary classification: Class 1 can be predicted correctly (TP) or incorrectly (FP), and Class 0 can be also predicted wrongly (FN) or correctly (TN). All of these cases are reflected in the confusion matrix (see Table 2.1).

Most frequently used classification quality metric is accuracy score, which can be determined as a fraction of correctly classified objects to the total number of objects. It can be expressed as:

Table 2.1: General representation of confusion matrix

	Class 1	**Class 0**
Predicted 1	True Positive, TP	False Positive, FP
Predicted 0	False Negative, FN	True Negative, TN

$$A = \frac{TP + TN}{TP + FP + FN + TN}. \tag{2.8}$$

However, if the classes are not well-balances, e.g. the number of positive objects (Class 1) is significantly greater than the number of negative objects (Class 0), then the above accuracy (A) measure is inadequate. Therefore, two additional metrics were introduced. The Precision (P) metric determines the level of confidence in positive class, which can be calculated by:

$$P = \frac{TP}{TP + FP}. \tag{2.9}$$

On the other hand, it is required to assess how well a classifier can find the positive objects, i.e. the fraction of true positive objects in total number of positive objects. The Recall score (R) can be used for this purpose and it can be determined by:

$$R = \frac{TP}{TP + FN}. \tag{2.10}$$

To tune machine learning algorithms, it is convenient to use one metric which unites Precision and Recall. This metric is the F-score and it can be computed as a harmonic mean of P and R:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R}. \tag{2.11}$$

This metric is close to zero if either P or R are close to zero and can assume a high value only when both values are high. This equation can be tuned by choosing β which is a real number in range [0, 1].

Despite there are four metrics to access the effectiveness or success rate of a machine learning attack, only the accuracy score is frequently used by researchers as PUF responses should be balanced for security purpose. Since the attacker is trying to build a binary classifier, $A = 0.5$ implies that the predictive model is as good as random guessing, and the PUF being attacked is considered to have low vulnerability to the modeling attack. In contrast, A bigger than 0.9 implies that the PUF design is highly predictable by the model.

2.5 Classical PUF designs

Classical PUF designs usually include Arbiter PUF, Ring Oscillator PUF and SRAM PUF. However, we included Bistable Ring PUF for fair comparison as it is implemented in FPGA and can be classified as a strong PUF.

2.5.1 Arbiter PUF

The Arbiter PUF was coined in [60]. It is classified as a delay-based PUF as the response is generated based on the intrinsic timing differences of two topologically and functionally identical paths in an IC due to its intra-die variations. As shown in Fig. 2.6, a set of symmetrical paths can be designed by N multiplexer stages.

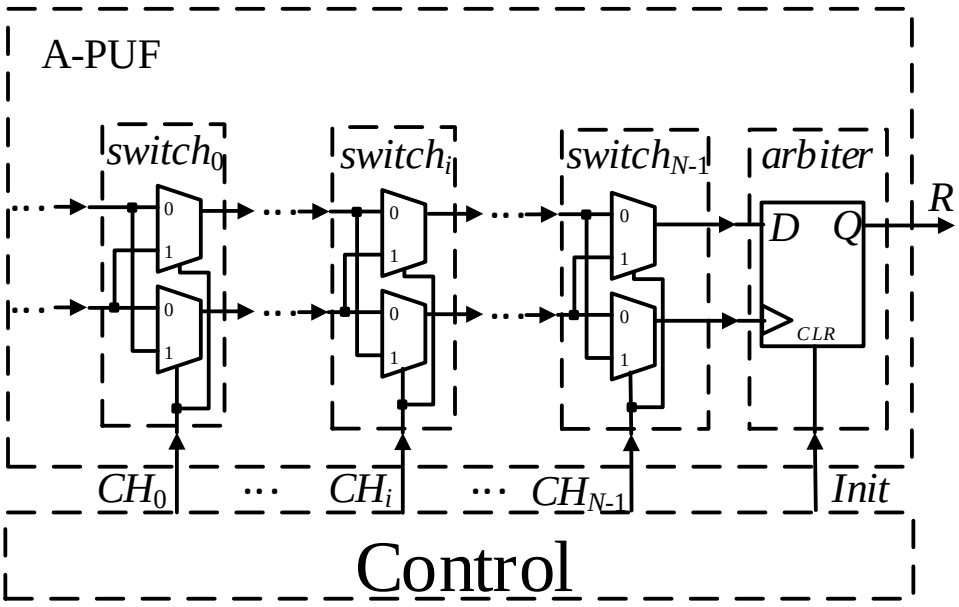


Figure 2.6: Classical arbiter PUF circuit.

Each stage $switch_i$ consists of a pair of multiplexers configured as a cross-bar switch. The data select lines of the two multiplexers in stage $switch_{N-1}$ are both fed by CH_{N-1}, where CH_i is the i-th bit of an N-bit challenge. Altogether 2^N different symmetrical paths can be selected by an N-bit challenge. The output signals of the two multiplexers at the last stage are connected respectively to the data and clock

inputs of a DFF (*arbiter*). The DFF acts as an arbiter to determine which of the signals along the two symmetrical paths is faster. When the output signal in the upper multiplexer is faster, the arbiter outputs a logic 1; otherwise, it outputs a logic 0.

2.5.2 Ring Oscillator PUF

Typical structure of Ring Oscillator PUF (RO-PUF) [72] includes N inverter chains, each with a negative feedback controlled by an AND gate, two N-input multiplexers for choosing a particular pair of ring oscillators ro_i ($i = 0, 1, \ldots, N-1$), two counters (cnt_0, cnt_1) for frequency computation and a comparator (cmp_0) for generating the final response bit R.

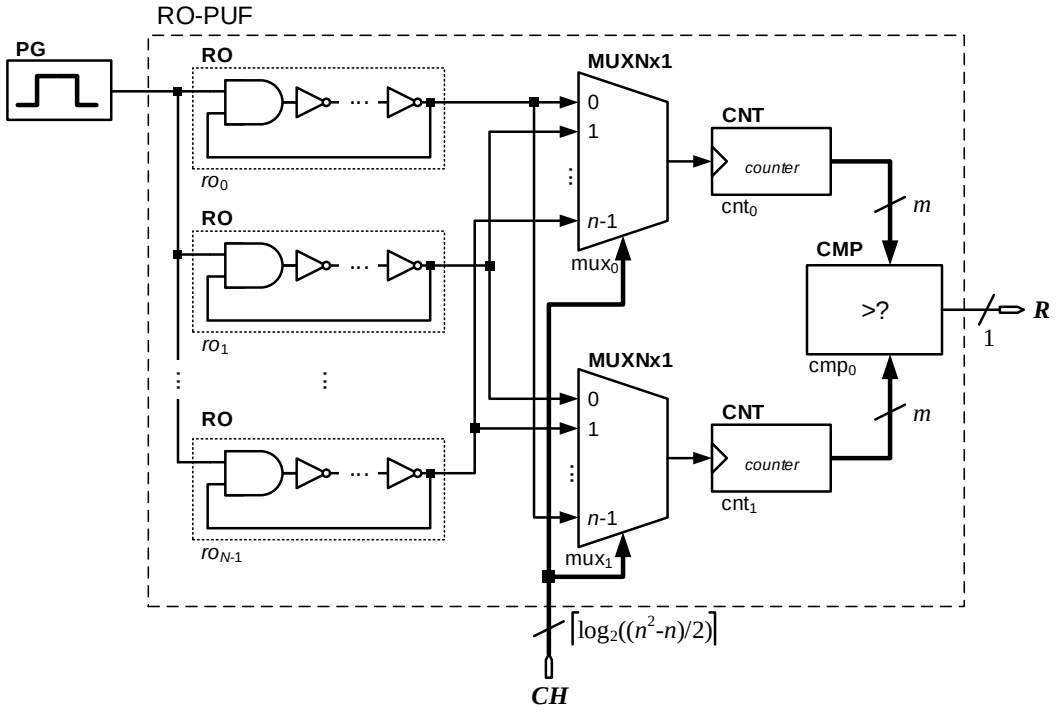


Figure 2.7: Classical ring oscillator PUF circuit.

The input challenge CH codes a pair of ring oscillators ro_i and ro_j ($i \neq j$). Then the outputs of the chosen ROs are propagated to two m-bit counters (cnt_0 and cnt_1) to compute their corresponding frequencies. If frequency of ro_i is less than ro_j, the comparator will generate a logic 0, and a logic 1 otherwise. Theoretically, the number of challenges is $\frac{N \cdot (N-1)}{2}$. In actual fact, it is much smaller due to the symmetry and

transitivity between pairs of ring oscillators. A classical design of RO-PUF is shown in Fig. 2.7.

2.5.3 SRAM PUF

The main idea behind memory based PUFs is to harness the asymmetrical trip-point of the cross-coupled inverters in a memory cell. Therefore, after power on or reset, the initial state of the cell will be determined by the manufacturing process variations that caused this asymmetry. A 6-transistor SRAM cell (see Fig. 2.8) was initially used to design a PUF [93] for both chip identification and true random number generation.

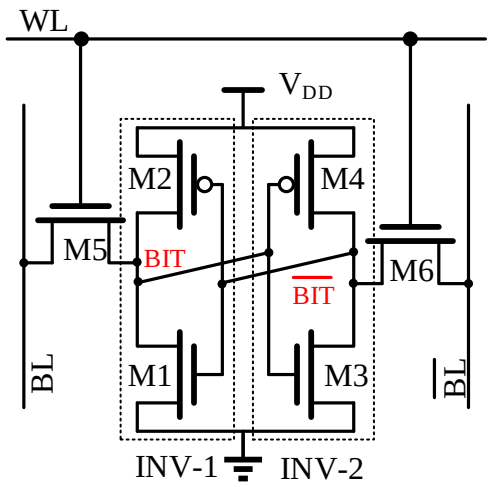


Figure 2.8: Standard 6-transistor SRAM cell.

Unfortunately, FPGA based implementations are rather unbalanced due to the utilization of already manufactured memory elements. The study in [94] shows the following distribution of memory cells reset values for the Virtex II Pro FPGA family: 90% of cells settles in logic 0, 9% in logic 1 and around 1% has equal probability of logic 0 and 1. Therefore, the memory cells have to be manufactured as symmetrical as possible to make good quality PUF design and to avoid a dominance of logic 0 or logic 1 reset values.

2.5.4 Bistable ring PUF

Bistable ring PUF is similar to the memory elements as it is based on even number of inverters which can emulate a memory cell [20]. This PUF contains N identical blocks,

each of which is implemented by two NOR gates, a multiplexer and a demultiplexer.

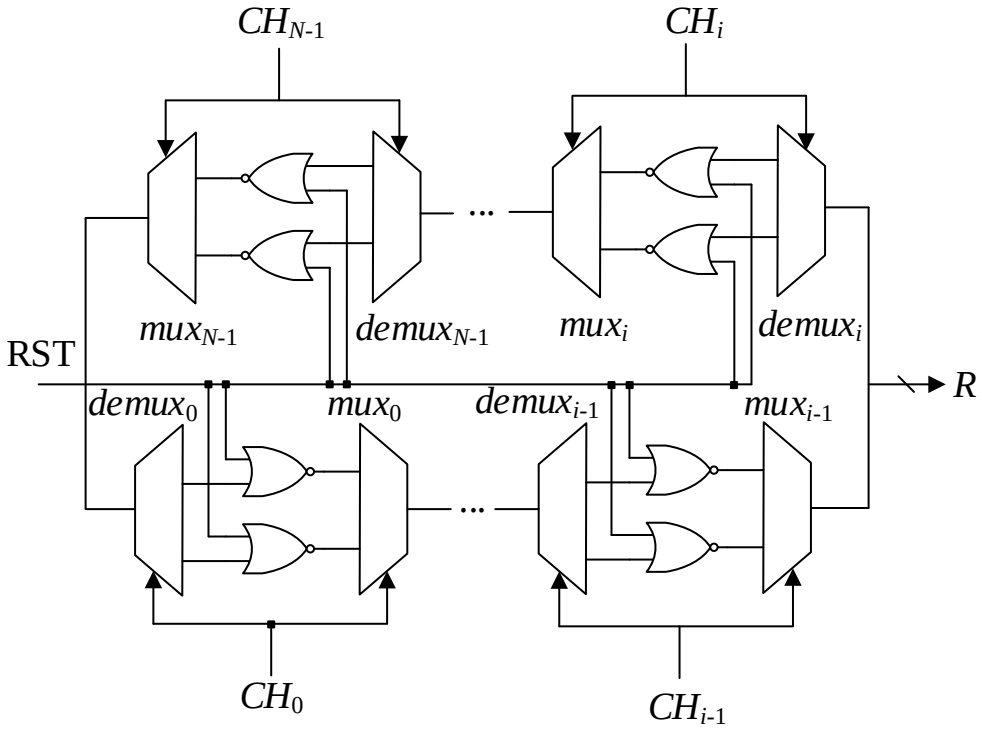


Figure 2.9: Bistable ring PUF circuit.

A reset signal controls the initial state of an N-stage bistable ring. The challenge bit CH_i ($i = 0, 1, \ldots, N-1$) configures each pair of inverters (NOR gates) by controlling their inputs (mux_i) and output ($demux_i$). The response value R can be generated based on the chosen block (in Fig. 2.9, the i-th pair of inverters is chosen). Thus, a bistable ring PUF can generate 2^N possible CRPs and is therefore classified as a strong PUF. The stability of the generated response value depends on the settling time of the configured ring. For 90% of the CRPs, the settling time does not exceed 50 ms and these response values are considered stable. On the other hand, the metastable values have a settling time of more than 50 ms and the final response value can be either logic 0 or logic 1.

2.5.5 Comparison

In this thesis, two major PUF applications are considered. The first one is the key (identifier) generation, which demands high reliability to provide unique and stable

keys for encryption or authentication protocols. The second one is the random number generation, which requires the PUF response sequence to be highly unpredictable and statistically similar to the true random noise. The PUF circuit has to be enhanced by correction scheme to enhance its reliability and compression scheme to improve the initial PUF noise [95]. Classical PUFs are implemented to serve in both modes, key and random number generation, for the sake of fair comparison. The PUF designs in comparison are evaluated by five parameters (CPR set size, hardware overhead, i.e. number of LUT and Flip-Flop blocks utilized in FPGA chip, uniqueness, randomness and reliability). The result of this comparison is shown in Table 2.2.

Table 2.2: Comparison of FPGA based PUF implementations

PUF	**CRP set size**	**# of LUT blocks**	**# of Flip-Flops**	**Uniqueness**	**Randomness (# of NIST tests passed out of 15)**	**Reliability**
A-PUF	2^{128}	324	128	0,473	15	0,957
RO-PUF	2^{12}	1288	945	0,399	13	0,979
SRAM-PUF	2^{10}	349	142	0,422	15	0,853
Bistable Ring PUF	2^{128}	512	2	0,435	13	0,972

RO-PUF and SRAM-PUF are both weak PUFs and should be processed by a hash function to be used as cryptographic primitive to secure the CRP set from being stolen by an attacker. Bistable Ring PUF has better reliability and poorer randomness than A-PUF, but it requires additional time overhead to generate random bit as its settling time is at least 50 ms. Therefore, to generate a large number of random bits (e.g. 10^6), more than 70 hours are required for the generation and post-processing, which is unacceptable for used as TRNG. Despite Arbiter PUF suffers from low initial reliability and its linear additive model is prone to attacks by machine learning algorithms, it is an ideal candidate to be used as a security primitive in FPGA chip as it has exponentially big CRP space, acceptable hardware overhead and comparatively less sensitive to temperature and supply voltage changes unlike RO-PUF [96] and SRAM-PUF [97].

2.6 Summary

This chapter reviews taxonomy of different hardware security methods to protect IP cores against counterfeiting and unauthorized access. The survey has shown that cryptographic primitive likes Physical Unclonable Function is highly desired, if not indispensable, in both passive and active hardware metering techniques. The key quality metrics (reliability, uniqueness, randomness and machine learning attack resistance) of PUF designs are also defined and clarified in this chapter. Classical PUF designs and recently introduced bistable ring PUFs were analyzed based on their hardware overhead, CRP set size and quality. The comparison has shown that Arbiter PUF remains a promising candidate to be implemented in FPGA platform. Possible disadvantages of classical A-PUF design such as low reliability and high vulnerability to machine learning attack are to be addressed in the following chapters to enhance the practical application of this design in authentication protocols, key generation and true random number generation.

Chapter 3

Metastability Detection for Reliability Enhancement of FPGA implementation of Arbiter PUF

3.1 Introduction

One main issue associated with the use of PUF as a cryptographic key or unique chip identifier in an authentication protocol is the minute mismatches of these exploited electrical characteristics within an IC are often instable and sensitive to temporal operating condition variations. This causes the responses to the same challenges to change from time to time. This problem is applicable for any kind of PUF as manufacturing process variations are not stable over environmental changes (e.g. temperature, supply voltage). To overcome this imperfection, error correction codes (ECC) and post processing can be applied to reduce the error rate of PUF response (e.g. [98]). Alternatively, existing structure of the PUF could be modified to stabilize the response by using devices with opposite parametric sensitivity to counteract the changes in certain operating condition such as [59]. If the native PUF quality is poor, the reliance on strong ECC to achieve high reliability is discouraged due to its high hardware overheads.

In general, Arbiter PUF is more popular for FPGA implementation as it requires less hardware resources than RO-PUF [99] and has bigger challenge-response space than memory-based PUFs. Being a strong PUF [13], it does not require additional post-processing to prevent replay attacks on authentication protocol. However, FPGA implementation of classical A-PUF is known to have poor reliability [100] due primarily to the metastability of D Flip-Flop (DFF) arbiter and routing constraints of FPGA [17]. Our experiments have shown that the average and minimum relia-

bility of an $N = 128$ stage classical A-PUF implemented on Xilinx Artix-7 FPGA are 0.5769 and 0.5648, respectively. Due to the routing constraints, it is especially hard to design symmetric electrical paths on a FPGA platform [65]. The timing mismatches in asymmetrically designed paths may outweigh the mismatches due to the unpredictable process variations, resulting in low unpredictability.

Several approaches have been proposed to enhance the reliability of A-PUF. Error Correction Codes is typically used to improve the reliability [101], with additional hardware overhead for ECC algorithm implementation and NVM for helper data storage. Maximum likelihood decoding scheme [102] is able to achieve the same reliability as ECC with lower hardware overhead, but it has the drawback of requiring more than one PUF instances for key generation. More arbiters can also be added to decrease the number of utilizable challenges and increase the reliability of the response at the expenses of hardware overhead [103]. Besides, the outputs of multiple arbiters are correlated, which may compromise the response unpredictability. Last but not least, hybrid PUF [95] combines the advantages of different kinds of PUF but this approach suffers from similar problem of requiring additional hardware resources, which are not suitable for resource-constrained applications.

In this section, we propose two main techniques to overcome the poor reliability of native A-PUF by replacing the DFF by enhanced 4-DFF and SR latch based arbiters in a multi-arbiter PUF (MA-PUF) scheme. The proposed techniques are able to detect and encode the metastable outputs of the PUF, which allows them to be isolated or recoded into the stable logic zero or one to improve the randomness, uniqueness and stability of the response. In addition, the flexibility of selecting different arbiters of the MA-PUF chain as outputs also expands the CRP space and enables better trade-offs to address different security requirements.

3.2 Multi-arbiter PUF Architecture

To enhance the reliability and unpredictability of classical A-PUF, a multi-arbiter PUF (MA-PUF) is proposed in [103], where each pair of multiplexers is connected to an arbiter as in the last stage of classical A-PUF, as shown in $MA\text{-}PUF_1$ of Fig. 3.1. Multiple response bits to an input challenge can be extracted from different stages of the multiplexer chain in any order by changing the address *Adr* to the end multiplexer.

Instead of generating multiple response bits from one A-PUF chain, one response bit can also be selected from each MA-PUF to generate a multi-bit response vector from several MA-PUF chains, as shown in Fig. 3.1. All MA-PUF chains share a

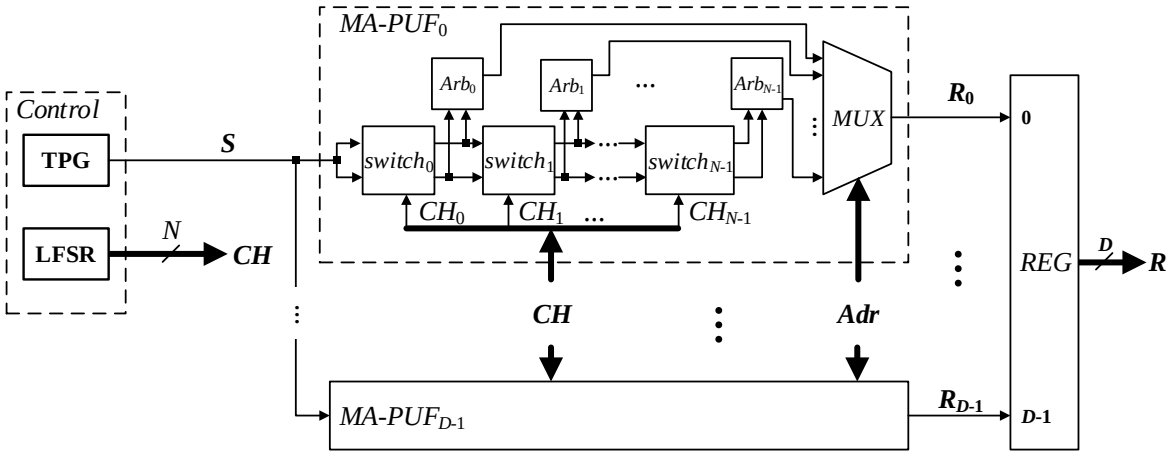


Figure 3.1: Multi-chain Multi-arbiter PUF Circuit.

Table 3.1: Parameters used for multi-chain MA-PUF experiments

Parameter	Definition
N	The maximum length of challenge vector, which is also the maximum number of cross-bar connected multiplexer stages in each MA-PUF chain.
i	The index of stage length, $i = 0, \ldots, N-1$
C	The number of applied challenge vectors, $C \leq 2^N$. Each challenge vector will produce one response vector.
c	The index of a challenge vector, which is also the index of its corresponding response vector, $c = 0, \ldots, C$
D	The number of MA-PUF chains on an FPGA, which is also the length of the response vector.
d	The index of a MA-PUF chain, $d = 0, \ldots, D-1$
F	The number of tested FPGAs
f	The index of an FPGA
E	The number of tests. Each test involves the application of the same set of C challenge vectors to all multi-chain MA-PUF implemented on F FPGAs.
e	The index of a test, $e = 1, \ldots, E$
$R_{e,f,d,i}(c)$	The response vector to the applied challenge vector c generated by multiplexer stage length N in chain d of the multi-chain MA-PUF implemented on FPGA f. $R_{e,f,d,i}(c) \in \{0,1\}$ for classical A-PUF circuit, $R_{e,f,d,i}(c) \in \{0,1\}^d$ for multi-chain MA-PUF)

common pseudorandom N-bit CH (CH_0, CH_1, ..., Ch_{N-1}) generated by a linear feedback shift register (LFSR). A test pulse generator (TPG) produces the common

input signal S to the first stage of each MA-PUF chain to launch the propagation delay race on the rising edge of S. The response vector to each challenge is stored in the output register REG.

A multi-chain MA-PUF circuit was implemented with $D = 8$ MA-PUF chains ($MA\text{-}PUF_d \, \forall d \in [0, 7]$). Each chain contains $N = 128$ cross-bar arbiters. To evaluate the uniqueness and reliability of this multi-chain MA-PUF scheme as the number of stages n in each chain changes, this PUF was tested $E = 30$ times on $C = 10{,}000$ pseudorandom challenges. The parameters involved in this experiment is summarized in Table 3.1.

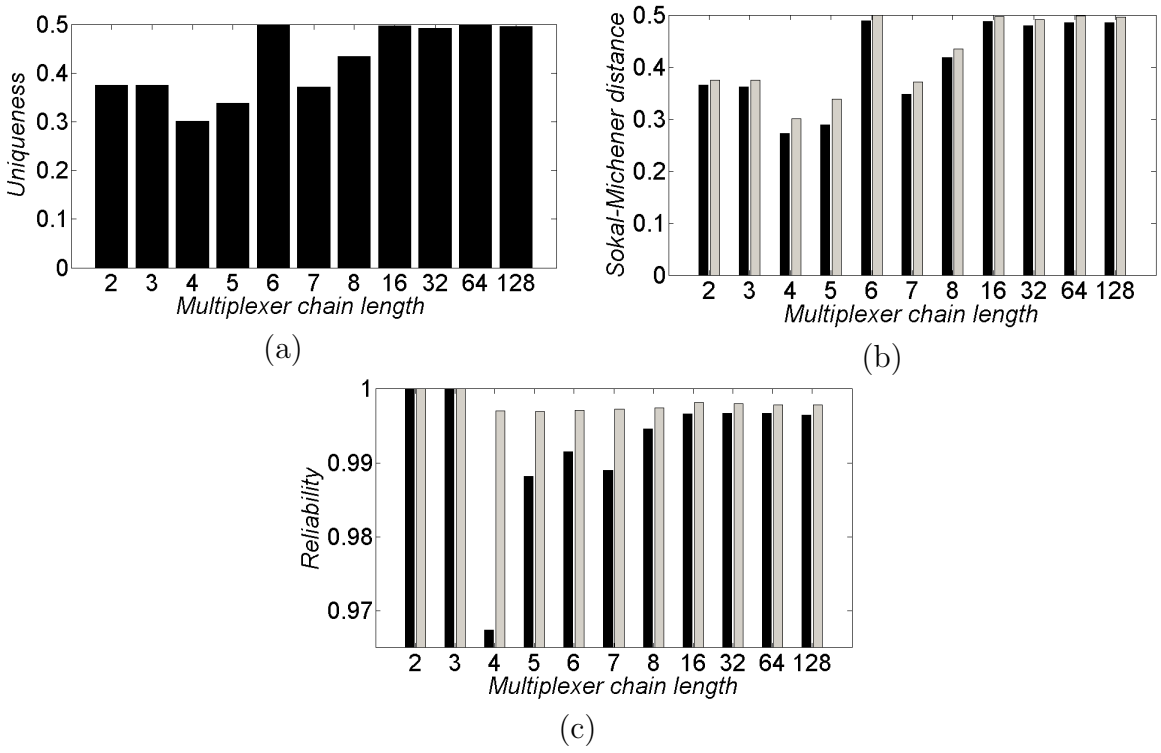


Figure 3.2: Multi-chain MA-PUF characteristics versus multiplexer chain length: (a) Uniqueness, (b) Sokal-Michener distance (average and minimum), (c) Reliability (average and minimum)

Our experiments show that the quality of MA-PUF depends on the multiplexer chain length n. The uniqueness calculated based on Hamming distance and the dissimilarity calculated based on Sokal-Michener distance [104] are plotted against n in Figs. 3.2(a) and 3.2(b), respectively. For multiplexer chain of less than 16 stages, there are clear signs of ill metastability effects that exhibit no specific trend of variation. For n greater than 16, the uniqueness and Sokal-Michener distance

approximate the ideal value of 0.5 and vary within a narrow range of [0.491, 0.498]. The same tendency and narrow spread are observed for the reliability plotted in Fig. 3.2(c) after the 16-th cross-bar arbiter stage. The multi-chain MA-PUF provides stable response values with an average reliability of 0.9982 and a minimum reliability of 0.9965. Such unique, random and stable response values can only be generated by classical FPGA-based A-PUF with ECC or manual routing. Thus multi-chain MA-PUF enables an FPGA chip to be uniquely identified with a great flexibility in CRP generation by changing the chain length n. The MA-PUF response values can be chosen from among many stable and uncorrelated arbiter responses from each chain by changing the address value Adr of Fig. 3.1, making it useful as a reconfigurable strong PUF. In what follows, methods to identify and resolve the metastable outputs of arbiters are proposed.

3.3 Identification of Metastable Arbiter Bits

3.3.1 4-DFF based arbiter

Although the CRP space has been enlarged by MA-PUF, the metastable operation of arbiter affects the reliability of MA-PUF responses, especially when the length N of the multiplexer chain is short.

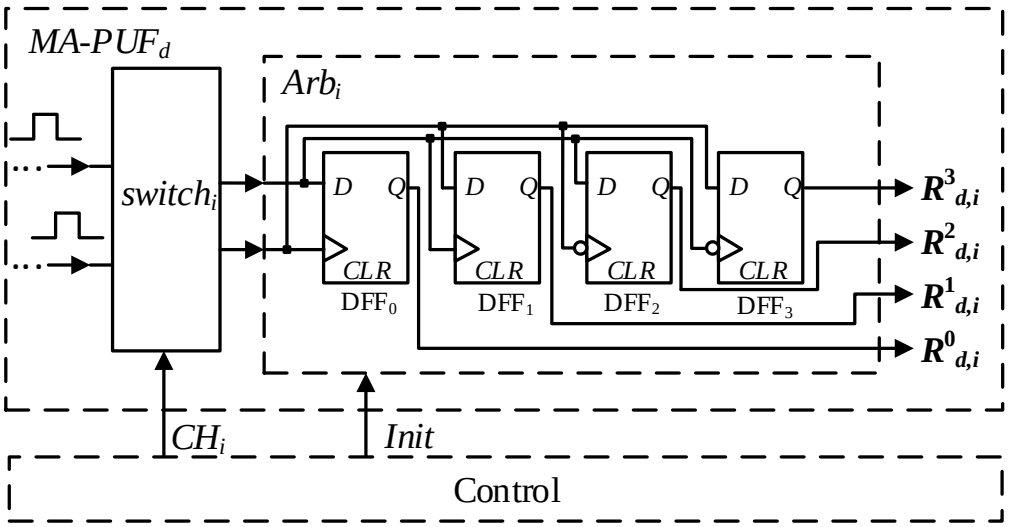


Figure 3.3: MA-PUF with enhanced arbiter.

The metastable states of MA-PUF can be detected by substituting the DFF arbiters by multi-valued arbiters as shown in Fig. 3.3. The connections of the two racing signal pulses to adjacent DFFs alternate between the D and clock inputs. The outputs, $R^1_{d,n}$ and $R^0_{d,n}$, of the first two FFs are triggered by the rising edges of the signals in the upper and lower multiplexer paths, respectively. The outputs, $R^3_{d,n}$ and $R^2_{d,n}$ of the last two FFs are triggered by the falling edges of the signals in the upper and lower multiplexer paths, respectively. Each of these output bits is asserted high at their respective active clock edge when the launching signal in the upper path is faster and asserted low if it is slower.

Furthermore, this arbiter circuit implementation is able to detect differences in pulse duty cycles. Classical A-PUF design assumes that duty cycles of two copies of a test pulse should be the same, but in fact they can be comparable or significantly different as shown in Fig. 3.4. Single DFF based arbiter circuit will generate logic 1 in both cases unlike the proposed one as it takes into account both rising and falling edges of a test pulse.

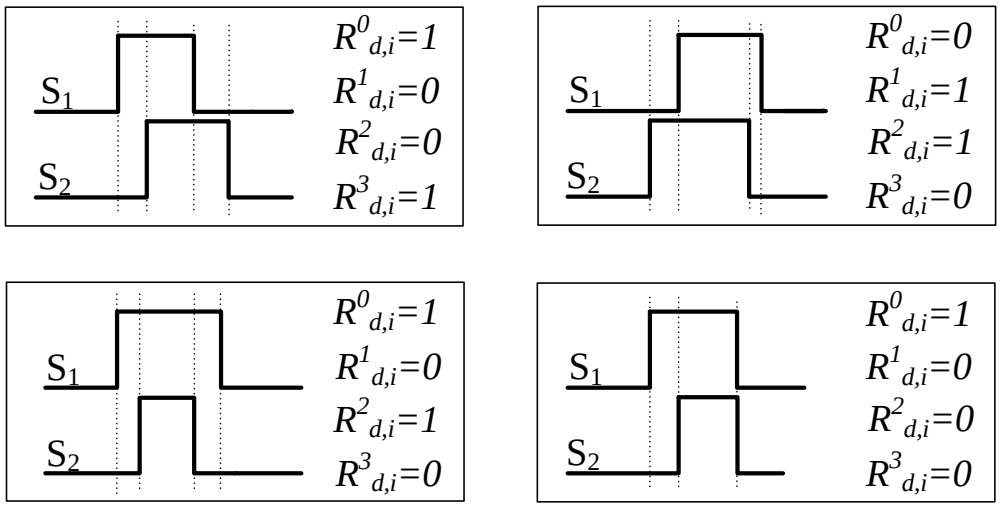


Figure 3.4: Different pulse duty cycles detection example.

The distribution of different quaternary logic outputs collected from a single arbiter in $E = 30$ tests of 10,000 CRPs is shown in Fig. 3.5.

Only the values "0110" and "1001" occur with approximately equal high probability and can be taken as the stable one and zero bits, respectively. Due to the routing restriction, the timing difference between two long signal paths may be negligibly small that causes the absence of some response values like "0000", "0100", "1000",

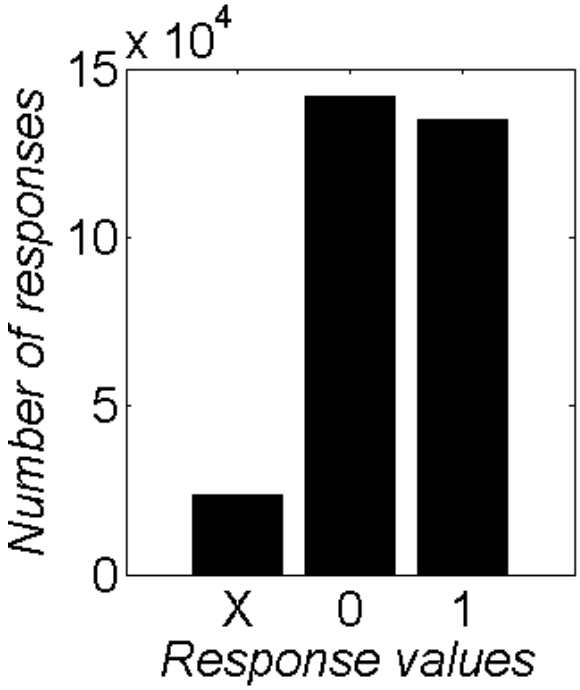


Figure 3.5: Distribution of 0, 1 and X outputs of MA-PUF with (a) 4-DFF based arbiter.

"1100", "1101", "1110" and "1111". The rest of the values that rarely occur are due to the metastable outputs of the arbiter. To improve the reliability and unpredictability of MA-PUF, these codes can be converted to "0110" as its probability of occurrence is less than that of "1001" (see Fig. 3.5).

3.3.2 SR latch based arbiter

Alternatively, the metastable oscillatory outputs of an arbiter can be detected and isolated by using a SR latch in place of a DFF, as shown in Fig. 3.7. The proposed SR latch arbiter consists of two cross-coupled NOR gates and two DFFs. Unlike the classical A-PUF, the proposed SR latch based arbiter is triggered by the falling edge of the test pulse. This is because metastable operation occurs when both S and R inputs transit from logic one to logic zero simultaneously and stay at logic zero for a long time. A frequency counter with at least two DFFs is required to capture the oscillatory metastable operation of SR latch. Three output states of SR latch are possible: stable logic zero, stable logic one and high frequency oscillatory (HFO) states [105].

HFO oscillation state has been detected in our experiments by implementation of

SR latch in K155 transistor-–transistor logic elements. The screen shot of Tektronix TBS1052B oscilloscope is shown in Fig. 3.6.

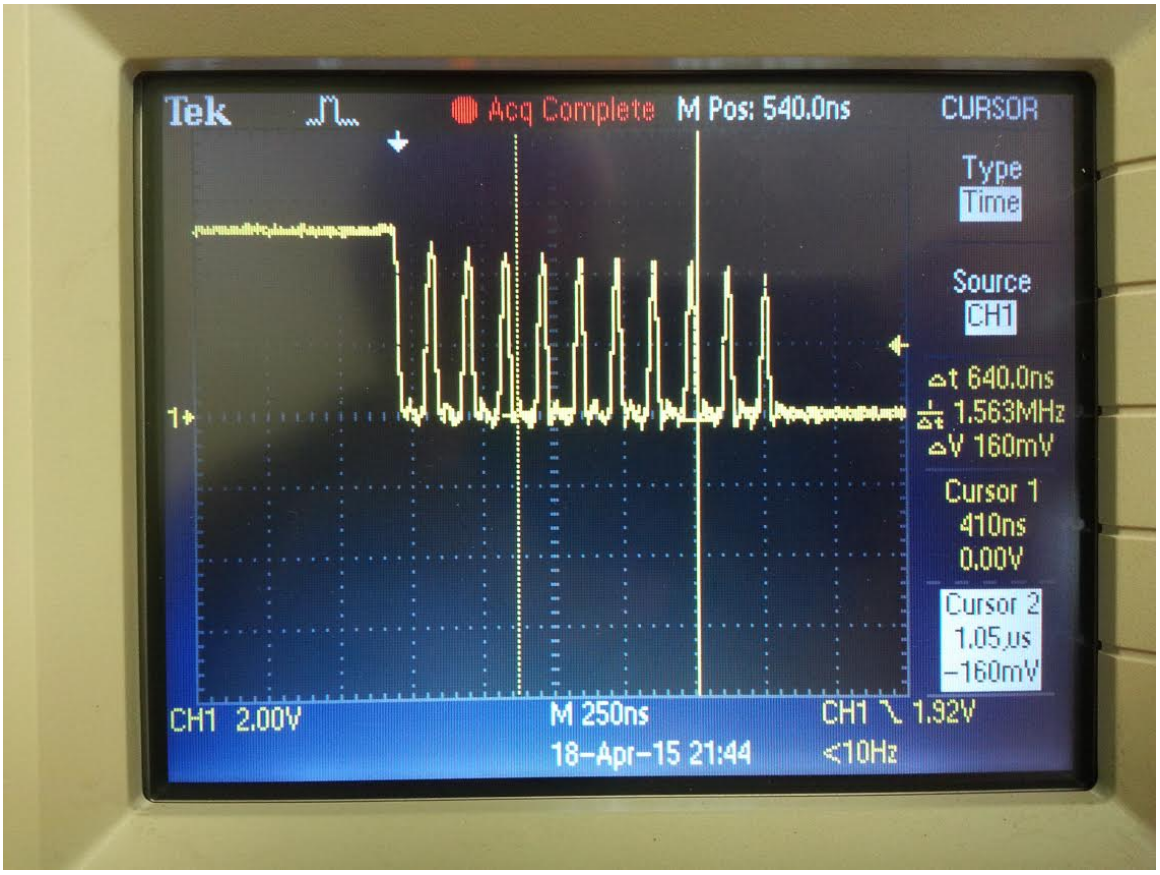


Figure 3.6: High frequency oscillation in oscilloscope screen.

These output states are encoded by a 2-bit binary counter. Stable zero state is encoded as $(R^0_{d,n} = 0, R^1_{d,n} = 0)$, stable one state as $(R^0_{d,n} = 1, R^1_{d,n} = 0)$ and HFO state as $(R^0_{d,n} = 1, R^1_{d,n} = 1)$. Thus, the output of the lower DFF $(R^1_{d,n})$ indicates if an arbiter response is stable $(R^1_{d,n} = 0)$ or metastable $(R^1_{d,n} = 1)$.

The distribution of the ternary response values {stable zero, stable one, HFO} is shown in Fig. 3.8. Around 10% of the response values are detected to be the unstable HFO. The ternary code $(R^0_{d,n} = 1, R^1_{d,n} = 1)$ corresponding to HFO can be arbitrarily but deterministically converted to either $(R^0_{d,n} = 0, R^1_{d,n} = 0)$ or $(R^0_{d,n} = 1, R^1_{d,n} = 0)$ to balance the probabilities of stable zero and stable one states (all metastable values will be converted to stable zero values when the number of occurrences of stable zero is less than stable one and vice versa). It is also observed that the oscillation frequency before the metastable operation settles to its final stable value changes with the input challenge to the MA-PUF. A more accurate

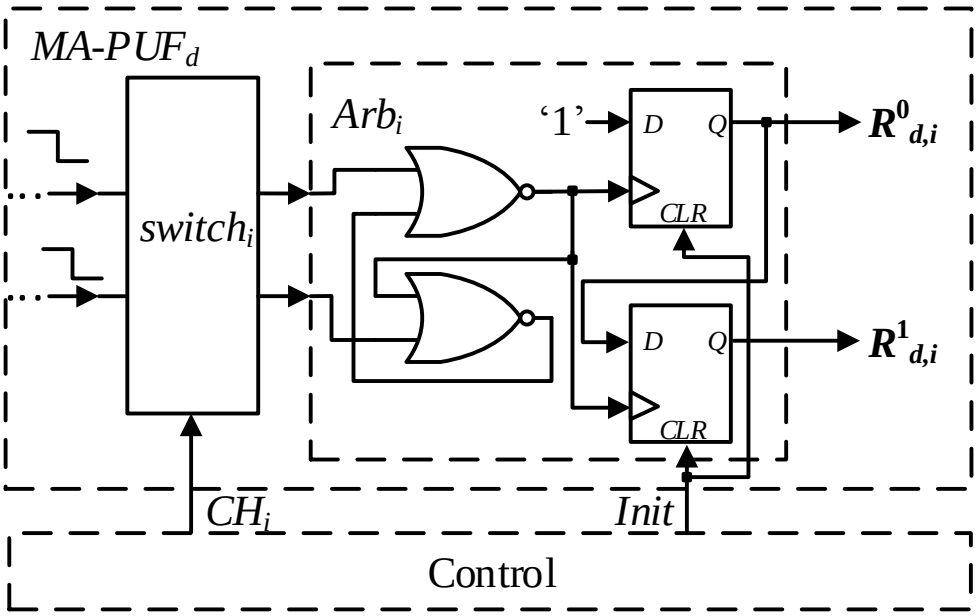


Figure 3.7: SR latch based arbiter.

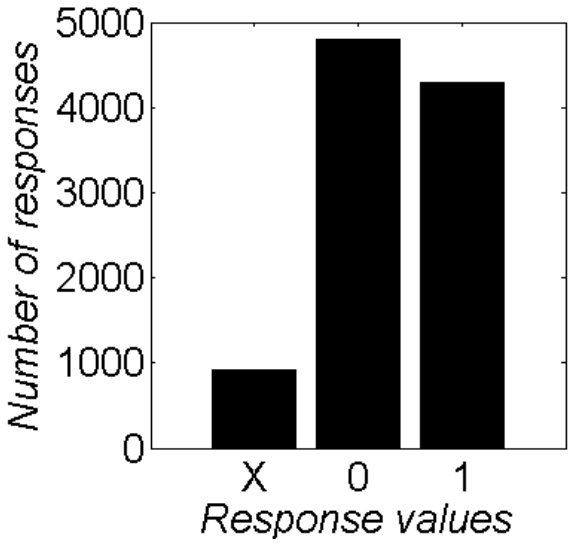


Figure 3.8: Distribution of 0, 1 and X outputs of MA-PUF with (a) 4-DFF based arbiter.

characterization of the HFO states for different input challenges can be accomplished by increasing the length of the frequency counter.

3.4 Experiment Results

The same single-chain MA-PUF designs with 4-DFF and SR latch arbiters were coded in VHDL, synthesized by CAD Xilinx ISE 14.7 and implemented on 10 Digilent Nexys-4 Artix-7 FPGA boards. Transfer of challenges and responses between the FPGA boards and personal computer was made via the UART interface. Every PUF instance was tested in $E = 30$ tests with $C = 10{,}000$ challenges generated by a 128-bit LFSR counter. It takes around 25 minutes to collect 10,000 CRPs from this system. The uniqueness, reliability and randomness for both enhanced MA-PUFs are then evaluated by considering both cases of keeping the codes of multi-valued arbiters corresponding to the metastable states as a ternary state (which means that X is equally probable to be recoded as a 1 or a 0 state) as well as converting them all to 1 state. The resource overheads of MA-PUF and enhanced MA-PUFs are also compared with the classical A-PUF implementation on the same FPGA.

3.4.1 Uniqueness

Ideally, $U = 0.5$, which is the maximum percentage of bit differences that can be obtained between both the true and complement forms of two binary vectors. The average uniqueness for 10,000 CRPs obtained from 10 MA-PUFs is 0.4972 with 4-DFF arbiter and 0.4982 with SR latch arbiter, which is very close to the ideal value.

If the codes corresponding to the metastable states identified by the 4-DFF arbiter and the HFO state of the SR latch based MA-PUF can be arbitrarily assigned to either a zero or a one state, then they can be represented by a ternary state X. The Hamming distance between two ternary vectors with each element belongs to the set $\{0, 1, X\}$ is redefined according to Table 3.2, where the Hamming distances between X and 0, and between X and 1 are both 0.5.

Table 3.2: Hamming distances between 0, 1 and X

Values	**0**	**1**	***X***
0	0	1	0.5
1	1	0	0.5
X	0.5	0.5	0

The average uniqueness computed with the modified HD for 10 PUF instances with 10,000 CRPs is 0.4959 for the 4-DFF arbiter based PUF and 0.4929 for the SR latch arbiter based PUF, which are also very close to the ideal value.

To estimate the uniqueness of generated unique identifier obtained by the concatenated responses for 10,000 challenges, the modified Sokal-Michener [104] distance between two ternary vectors is used. The variables that denote the nine possible combinations of every possible pairs of ternary values obtained at the same position i of two ternary vectors, v_1 and v_2, are listed in Table 3.3.

Table 3.3: Different combinations of ternary values for Sokal-Michener distance calculation

Variable	$v_1(i)v_2(i)$
a	11
b	01
c	10
d	00
e	XX
f	$0X$
g	$X0$
h	$1X$
i	$X1$

To account for the fact that state X can be recoded into either state 0 or state 1, the modified Sokal-Michener distance is calculated by:

$$D_{Sokal-Michener} = \frac{b+c}{m} + \frac{0.5\cdot(f+g+h+i)}{m} = \frac{2\cdot(b+c)+f+g+h+i}{2\cdot m} \tag{3.1}$$

If the state X for the multi-valued arbiters is recorded into the stable 1 state, the similarity of two binary vectors v_1 and v_2 of length m can be expressed as:

$$S_{Sokal-Michener} = \frac{a+d}{m} \tag{3.2}$$

The distance between two binary vectors can then be computed as $1-S_{Sokal-Michener}$.

For MA-PUF with 4-DFF based arbiter, the average and minimum Sokal-Michener distances are 0.4971 and 0.4868, respectively if all Xs are recoded as 1s, and 0.4954 and 0.4867, respectively if all Xs are preserved. On the other hand, for the MA-PUF with SR latch based arbiter, the modified average and minimum Sokal-Michener distances are 0.4983 and 0.4888, respectively if X is recoded as 1, and 0.4929 and 0.4854, respectively if X is preserved.

3.4.2 Reliability

Since the output alphabet is expanded to a ternary set {0, 1, X}, the corresponding R_i value is modified as follows:

$$R_i = \frac{max(n_0, n_1, n_X)}{n_0 + n_1 + n_X} \tag{3.3}$$

where n_0, n_1 and n_X are the numbers of zero, one and metastable states, respectively in E tests, and $n_0 + n_1 + n_X = E$.

The Hamming distance used in the reliability computation is also changed according to the definition of the variables given in Table 3.2. 10,000 CRPs are collected in 30 experiments from the MA-PUF implemented on 10 FPGA boards. The average and minimum S obtained for MA-PUF with 4-DFF based arbiter are 0.9986 and 0.9979, respectively by recoding X to 1, and 0.9985 and 0.9978, respectively by retaining X. The proposed MA-PUF with SR based arbiter has very similar high average and minimum reliability S of 0.9985 and 0.9978, respectively with recorded X, and 0.9975 and 0.9965, respectively with X retained.

3.4.3 Randomness

Ten million response bits were generated and split into 100 blocks of 100,000 bits each. For MA-PUF with SR latch based arbiters, the X-bits were arbitrarily substituted by zero or one bits by a pseudorandom number generator. The results of NIST test are shown in Table 3.4.

The outputs of both MA-PUFs have successfully passed all NIST tests and hence, their responses can be considered as statistically meeting the criteria of a true random number sequence.

3.4.4 Hardware overhead analysis

The hardware overheads for different types of single-chain arbiter PUF implementation are shown in Table 3.5. Despite large number of registers used compared to the classical A-PUF, these additional resources are negligible compared with the expanded CRP space and their enhanced reliability and uniqueness for highly robust cryptographic key generation. The entire MA-PUF system includes the UART controller, 128-bit LFSR counter, PUF storage registers and the PUF itself. The complete MA-PUF system consumes only less than 4% of the total logic slices and less than 1% of registers available in the target FPGA chip. In fact, the additional arbiters of

Table 3.4: NIST test results

Test Description	**Passed/Total**		**P-value**	
	4-DFF	SR	4-DFF	SR
Frequency (Monobit) Test	100/100	100/100	0.74	0.53
Frequency Test within a Block	100/100	100/100	0.12	0.53
Runs Test	100/100	100/100	0.74	0.12
Test for the Longest Run of Ones in a Block	100/100	100/100	0.53	0.99
Binary Matrix Rank Test	100/100	100/100	0.35	0.91
Discrete Fourier Transform (Spectral) Test	100/100	100/100	0.53	0.21
Non-overlapping Template Matching Test	97/100	98/100	0.73	0.76
Overlapping Template Matching Test	100/100	100/100	0.74	0.91
Maurer's "Universal Statistical" Test	100/100	100/100	0.07	0.02
Linear Complexity Test	100/100	100/100	0.51	0.43
Serial Test	100/100	100/100	0.21	0.74
Approximate Entropy Test	100/100	100/100	0.35	0.62
Cumulative Sums (Cusum) Test	100/100	100/100	0.12	0.07
Random Excursions Test	10/10	10/10	—	—
Random Excursions Variant Test	10/10	10/10	—	—

4-DFF and SR latch based MA-PUF over the basic A-PUF consume less than 0.08% and 0.4% of logic slices and less than 0.4% and 0.2% of registers, respectively, of the total resources available on the FPGA.

Table 3.5: Hardware overhead comparison

Component	# slice LUTs	# slice registers
A-PUF	256 / 63400	1 / 126800
MA-PUF	298 / 63400	128 / 126800
4 DFF	302 / 63400	512 / 126800
SR latch	506 / 63400	256 / 126800
Entire system	2494 / 63400	1263 / 126800

By sacrificing a relatively small part of the total FPGA hardware resources, MA-PUFs provides greater flexibility to use any arbiter outputs from the 16th to 128th multiplexer stages as entropy sources to generate better quality multi-bit response to a given challenge.

3.5 Generation of Unique Chip Identifier by MA-PUF

The proposed MA-PUF systems can be used for unique chip identification and strong authentication. There are three possible ways for the generation of chip identifiers:

1. The initial challenge is used as a seed of the LFSR, which generates the select signals to the multiplexers and *Adr* to select arbiters for the response bit generation. The LFSR generates additional k *Adr* words from the input challenge to produce a k-bit identifier.

2. The applied challenge is used directly to configure the signal paths. Specific arbiters in the multiplexer chain are chosen at design time. The arbiters can be selected based on inter-arbiter uniqueness, high reliability and better randomness after the characterization the MA-PUF. The resources occupied by the unselected arbiters can then be freed.

3. Similar to the second method except that part of the challenge is used for arbiter selection. This approach is more secure, but requires a longer challenge.

Table 3.6: Correlation between arbiter outputs

Arb_i	121	122	123	124	125	126	127	128
121		N	N	N	N	N	S	N
122	N		N	N	S	N	N	N
123	N	N		N	S	N	S	N
124	N	N	N		S	N	N	N
125	N	S	S	S		N	S	N
126	N	N	N	N	N		N	N
127	S	N	S	N	N	N		N
128	N	N	N	N	N	N	N	

Table 3.6 shows the results of the hypothesis test for correlation of eight adjacent arbiters in a single chain of MA-PUF. The correlation between the two arbiters with the row and column indexes is marked S for statistically significant and N for statistically insignificant. The black colored cells represent self-correlation of the same arbiter, which is always 1. The results show that uncorrelated adjacent arbiters are rare.

The arbiters should preferably be selected so that the minimum chain length is 16. In addition, one should also avoid addressing adjacent correlated arbiters in sequence

to generate a multi-bit response. The distance between sequentially addressed arbiters should preferably be random and large.

Proposed MA-PUF design has significantly larger response space comparing to classical design and it can be used to generate unique chip ID with better performance as only one clock cycle is required to generate an N-bit response instead of N clock cycles in ordinary A-PUF. In this case each additional response bit costs only 2 LUT blocks and 2 Flip-Flops in FPGA design. This is a reasonable hardware overhead to save extra clock cycles and increase response space for A-PUF. On the other hand, experiment results have shown that the proposed MA-PUF design has higher observability due to additional arbiters which are correlated. These results are not surprising as A-PUF design is vulnerable to machine learning attacks [106] and increasing the number of bits in the response leads to reduced complexity in maintaining the resistance against modeling attack. Thus, despite better reliability, additional countermeasure against machine learning attacks is needed, which will be discussed in Chapter 5.

3.6 Summary

This chapter presents and analyzes two techniques to identify and exploit metastable states to improve the uniqueness and reliability of multi-arbiter PUF design. The proposed 4-DFF and SR latch based MA-PUF designs have been implemented on FPGA platform (Digilent Nexys-4 Artix-7 FPGA board). Our experimental results show that these MA-PUFs possess very high-quality uniqueness, reliability and randomness compared with the classical arbiter PUF. The quality of these MA-PUFs has insignificant fluctuation (around 0.001) if the first 16 arbiters are excluded. With a small hardware overhead, these MA-PUFs greatly expand the CRP space for strong PUF applications and provide additional agility in generating unique chip identifiers for FPGA device authentication and integrity protection.

Chapter 4

Challenge Classification for Reliability Enhancement of Arbiter PUF in FPGA Implementation

4.1 Introduction

Since arbiter metastability is the main culprit of response instability, previous chapter presents metastability state detection techniques ultizing either 4-DFF based or SR latch based arbiter to significantly improve the average and minimum reliabilities. This technique incurs some detection overhead of metastable states and recoding overhead due to unbalanced response alphabets. Previous chapter also shows that different arbiters of a multi-arbiter PUF chain can be selected to output multiple response bits to expand the CRP space to have better trade-off between hardware cost and reliability.

In this chapter, a new insight into the asymmetric path delays of basic A-PUF is provided, taking into consideration that hardware resources can be more precious than response generation time in FPGA-based PUF applications and the abundant challenges of A-PUF for which not all of them produce stable responses. An expanded A-PUF delay model is established to separate the input challenges into two classes of challenge quadruples, "stable" and "unstable". A challenge filtering algorithm is proposed with a knob to tune the extent of reliability enhancement. By merely sifting the "persistently stable" quadruple challenges, perfect reliability of 1.0 can be achieved over a temperature range from $-40°C$ to $90°C$ without incurring any additional hardware overhead. Except the time penalty to process the input challenges, other quality of the underlying A-PUF remains either unaffected or improved. Based on the challenge classification, an algorithm is proposed to generate FPGA IDs with

perfect reliability and ideal uniqueness (in statistical sense). Furthermore, the "unstable" challenges identifiable by the proposed technique can be singled out for better random number sequence generation. This chapter is organized as follows. The theoretical underpinning of A-PUF delay model for the proposed challenge classification is introduced in Section 4.2. The classification of "stable" and "unstable" challenges and A-PUF reliability enhancement algorithm are detailed in Section 4.3. Simulation results, and temperature reliability tests, uniqueness, randomness and ID generation experiments on physical FPGA implementation are presented and discussed in Section 4.4. Section 4.5 concludes the chapter by highlighting the limitation of challenge filtering in addressing other A-PUF problems.

4.2 Delay Model of Arbiter PUF

Typically, an arbiter PUF is implemented with a test pulse generator (TPG), N connected path-swapping switches ($switch_i$, $i = 0, 1, \ldots, N-1$) and an arbiter circuit ($arbiter$). The basic A-PUF structure is depicted in Fig. 4.1.

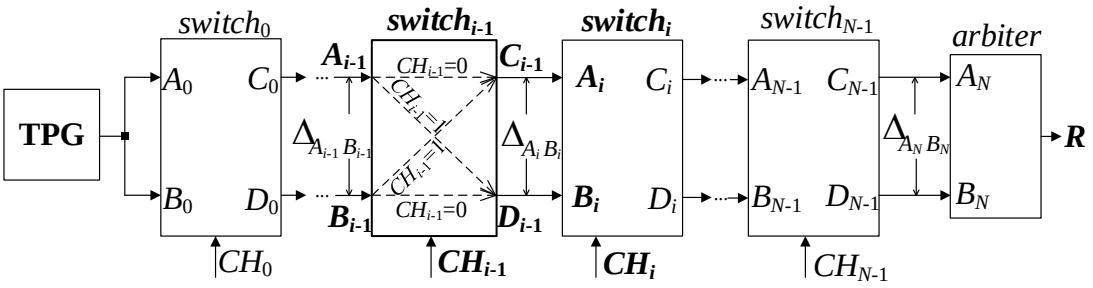


Figure 4.1: Structure of an N-stage A-PUF.

Each switching element ($switch_i$) can be described by a 4-port block with two input ports, A_i and B_i, and two output ports, C_i and D_i. Depending on the binary value CH_i of the input challenge, the switch $switch_i$ either operates in straight mode or exchange mode. When $CH_i = 0$ (straight mode), A_i is connected directly to C_i and B_i is connected directly to D_i. When $CH_i = 1$ (exchange mode), the connections of the input ports to the output ports are swapped such that A_i is connected to D_i and B_i is connected to C_i (see Fig. 4.1). The switch can be implemented in different ways by using either ordinary multiplexers [17] or tristate buffers [107]. The N switching elements are cascaded in a chain with the inputs of the first switch

$switch_0$ connected to a common test pulse generated by TPG and the outputs of the last switch $switch_{N-1}$ routed to the inputs of an arbiter.

The most critical parameter that determines the likelihood of flipping the response bit by a bit change of the challenge is the difference between the arrival time of the same test pulse traversing in different paths to the two input ports of a switch. Let t_p and t_g denote the time taken for the edge (rising edge for DFF based arbiter and falling edge for SR latch based arbiter) of a test pulse launched from TPG to reach ports p and g, respectively, where $p,\ g\ \in\ \{A_i, B_i, C_i, D_i\}$. Then, the difference in their arrival time $\Delta_{pg} = t_p - t_g$. For example, $\Delta_{A_iB_i} = t_{A_i} - t_{B_i}$ represents the difference in the arrival time of the test pulse to the input ports A_i and B_i. The propagation delay from the input port B_i to the output port C_i can also be expressed as $\Delta_{C_iB_i} = t_{C_i} - t_{B_i}$. Apparently, $\forall p = g$, $\Delta_{pg} = 0$, and $\forall p \neq g$, $\Delta_{pg} = -\Delta_{gp}$.

To meet the quality criteria of a good PUF, $\forall i = 1, \ldots, N$, the sets of values, $\{\Delta_{C_{i-1}A_{i-1}}, \Delta_{D_{i-1}A_{i-1}}, \Delta_{C_{i-1}B_{i-1}}, \Delta_{D_{i-1}B_{i-1}}\}$ and $\{\Delta_{A_iC_{i-1}}, \Delta_{B_iD_{i-1}}\}$, should be unique and non-replicable not only for a particular implementation of A-PUF on a single chip, but also across a set of chips. This means that the two timing paths traversed by the test pulse from the input ports, A_{i-1} and B_{i-1}, of $switch_{i-1}$ to the input ports, A_i and B_i, of $switch_i$ due to the different logic states of CH_{i-1} should be uniquely different from one chip to another.

Unfortunately, metastability state can appear at the arbiter output R, which reduces its reliability. This situation arises when the propagation delay difference between the two test pulse paths falls within the interval $[t_{low},\ t_{high}]$. The values of t_{low} and t_{high} are determined by the timing specifications of the arbiter. If the arbiter is implemented as a DFF, the value of $t_{low} = -t_{hold}$ and $t_{high} = t_{setup}$, where the hold time t_{hold} (t_{setup}) of the DFF is defined as the minimum time during which the signal on the data input B_N is required to be stable after (before) the active clock's edge has arrived at the clock input A_N. If the arbiter is implemented using SR latch, t_{low} and t_{high} will depend on the ratio of the rise/fall time to propagation delay and the voltage gain of the constituent NOR gates that are connected in a cross-coupled configuration to provide the positive feedback required to maintain the output in a stable state [105].

Taking into account the metastability conditions, the final state on the output port R of an A-PUF can be determined by the propagation delay difference $\Delta_{A_NB_N}$ as follows:

$$R = \begin{cases} 0 \text{ if } \Delta_{A_N B_N} \leq t_{low}, \\ 1 \text{ if } \Delta_{A_N B_N} \geq t_{high}, \\ X \text{ if } t_{low} < \Delta_{A_N B_N} < t_{high}. \end{cases} , \tag{4.1}$$

where X represents a metastable state.

The arrival time difference $\Delta_{A_i B_i}$ of the launching pulse to a switching element $switch_i$ can be calculated recursively by the following function:

$$\Delta_{A_i B_i} = \gamma(\delta_{i-1}^{CH_{i-1}}, \Delta_{A_{i-1} B_{i-1}}), \tag{4.2}$$

where $\delta_{i-1}^{CH_{i-1}}$ is the propagation delay through $switch_{i-1}$, which is dependent on the challenge bit CH_{i-1}. $\Delta_{A_{i-1} B_{i-1}}$ is the difference in arrival time of the test pulse between the two inputs of $switch_{i-1}$.

If $CH_{i-1} = 0$, then $\Delta_{A_{i-1} B_{i-1}}$ is given by $\Delta_{A_i B_i} = (t_{A_i} - t_{A_{i-1}}) - (t_{B_i} - t_{B_{i-1}}) + (t_{A_{i-1}} - t_{B_{i-1}}) = \Delta_{A_i A_{i-1}} - \Delta_{B_i B_{i-1}} + \Delta_{A_{i-1} B_{i-1}}$. If $CH_{i-1} = 1$, then $\Delta_{A_i B_i} = (t_{A_i} - t^{B_{i-1}}) - (t_{B_i} - t_{A_{i-1}}) + (t_{B_{i-1}} - t_{A_{i-1}}) = \Delta_{A_i B_{i-1}} - \Delta_{B_i A_{i-1}} - \Delta_{A_{i-1} B_{i-1}}$. Thus, $\delta_{i-1}^0 = \Delta_{A_i A_{i-1}} - \Delta_{B_i B_{i-1}}$ and $\delta_{i-1}^1 = \Delta_{A_i B_{i-1}} - \Delta_{B_i A_{i-1}}$. Taking into account that the propagation delay differences of the two pairs of path in straight and exchanged modes within each switch are relatively small and comparable, we can reasonably assume that the absolute values of δ_{i-1}^0 and δ_{i-1}^1 are both equal to a minute value δ_{i-1}. Then, $\delta_{i-1}^0 = -\delta_{i-1}^1 = \delta_{i-1}$ and $\delta_{i-1}^1 = -\delta_{i-1}^0 = -\delta_{i-1}$, respectively, and (4.2) can be re-expressed as:

$$\begin{aligned} \Delta_{A_i Bi} = (1 - CH_{i-1}) \cdot (\delta_{i-1} + \Delta_{A_{i-1} B_{i-1}}) + \ CH_{i-1} \cdot (-\delta_{i-1} - \Delta_{A_{i-1} B_{i-1}}) = \\ (1 - 2 \cdot CH_{i-1}) \cdot (\delta_{i-1} + \Delta_{A_{i-1} B_{i-1}}). \end{aligned} \tag{4.3}$$

The sign and magnitude of $\Delta_{A_i Bi}$ are determined by CH_i and $(\delta_{i-1} + \Delta_{A_{i-1} B_{i-1}})$, respectively. Let

$$Sign_{i-1} = 1 - 2 \cdot CH_{i-1}. \tag{4.4}$$

Then, $Sign_{i-1} = 1$ when $CH_{i-1} = 0$ and $Sign_{i-1} = -1$ when $CH_{i-1} = 1$. Equation (4.3) can be rewritten into a simpler and more intuitive form as follows:

$$\Delta_{A_i B_i}(CH_{i-1}) = Sign_{i-1} \cdot \delta_{i-1} + Sign_{i-1} \cdot \Delta_{A_{i-1} B_{i-1}}. \tag{4.5}$$

Since the test pulse is applied directly into the inputs of $switch_0$ (see Fig. 4.1), $\Delta_{A_0 B_0} \approx 0$. From (4.5), the value of $\Delta_{A_1 B_1}$ can be calculated as:

$$\Delta_{A_1B_1}(CH_0) = Sign_0 \cdot \delta_0. \tag{4.6}$$

Combining (4.5) and (4.6), the value of $\Delta_{A_2B_2}$ can be calculated by

$$\Delta_{A_2B_2}(CH_1) = Sign_1 \cdot \delta_1 + Sign_1 \cdot \Delta_{A_1B_1} = Sign_1 \cdot \delta_1 + Sign_1 \cdot Sign_0 \cdot \delta_0. \tag{4.7}$$

By unrolling the recursion, it is evident that $\Delta_{A_2B_2}$ depends on both challenge bits preceding $switch_2$. To be exact, the argument of $\Delta_{A_2B_2}$ in the left hand side of (4.7) should include both CH_1 and CH_0. Similarly, $\Delta_{A_3B_3}$ can also be expanded into:

$$\begin{aligned} \Delta_{A_3B_3}(CH_2, CH_1, CH_0) = Sign_2 \cdot \delta_2 + \\ + Sign_2 \cdot Sign_1 \cdot \delta_1 + Sign_2 \cdot Sign_1 \cdot Sign_0 \cdot \delta_0. \end{aligned} \tag{4.8}$$

Finally, $\Delta_{A_NB_N}$ can be computed by:

$$\Delta_{A_NB_N}(CH_{N-1}, CH_{N-2}, \ldots, CH_0) = \sum_{j=0}^{N-1}(\delta_j \prod_{i=j}^{N-1} Sign_i). \tag{4.9}$$

For ease of exposition, each possible combination of an input challenge of length N is represented by an integer variable Ω, where $\Omega = \sum_{i=0}^{N-1} 2^i \cdot CH_i$ is a decimal representation of an input challenge $\{CH_{N-1}, CH_{N-2}, \ldots, CH_0\}$, with the least significant bit (LSB) CH_0 applied to $switch_0$.

Consider an input challenge of all zeros, CH^0 with

$$\Delta_{A_NB_N}(CH^0) = \delta_{N-1} + \delta_{N-2} + \ldots + \delta_0 = \sum_{i=0}^{N-1} \delta_i. \tag{4.10}$$

By flipping only the challenge bit to the first switching element $switch_0$, we have $CH_0 = 1$ and $CH_i = 0$, $\forall i \ = \ 1, \ldots, N-1$ and

$$\begin{aligned} \Delta_{A_NB_N}(CH^1) = \delta_{N-1} + \delta_{N-2} + \ldots - \delta_0 = \\ \sum_{i=0}^{N-1} \delta_i - 2 \cdot \delta_0 = \Delta_{A_NB_N}(CH^0) - 2 \cdot \delta_0. \end{aligned} \tag{4.11}$$

Assume that an A-PUF circuit produces a stable output response R (see (4.1)) for the input challenge CH^0. Since the absolute value of $2\delta_0$ is much smaller than

the value of $\Delta_{A_N B_N}(CH^0)$, it is very unlikely that the response value will change by changing the input challenge from CH^0 to CH^1.

However, if the input challenge value is $CH^{2^{N-1}} = \{1, 0, \ldots, 0\}$, then:

$$\Delta_{A_N B_N}(CH^{2^{N-1}}) = -\delta_{N-1} - \delta_{N-2} - \ldots - \delta_0 = -\sum_{i=0}^{N-1} \delta_i = -\Delta_{A_N B_N}(CH^0). \quad (4.12)$$

By comparing (4.10) and (4.12), if the most significant bit (MSB), which is the challenge bit applied to the last switching element, is flipped, it is highly probable that the response R obtained by the original input challenge CH^0 will be inverted.

For further analysis, consider an arbitrary challenge CH^{Ω} and three other challenges, $CH^{\Omega'}$, $CH^{\Omega''}$ and $CH^{\Omega'''}$, obtained by inverting only the LSB, only the MSB, and both the LSB and the MSB of CH^{Ω}, respectively. In other words, $|\Omega - \Omega'| = 1$, $|\Omega - \Omega''| = 2^{N-1}$, $|\Omega''' - \Omega''| = 1$ and $|\Omega''' - \Omega'| = 2^{N-1}$.

When only the LSB is inverted in a challenge CH^{Ω}, according to (4.11), the sign of δ_0 will toggle. The absolute difference between $\Delta_{A_N B_N}$ for CH^{Ω} and $CH^{\Omega'}$ can be estimated from (4.9) by:

$$\begin{aligned} |\Delta_{A_N B_N}(CH^{\Omega}) - \Delta_{A_N B_N}(CH^{\Omega'})| = |Sign_{N-1} \cdot Sign_{N-2} \cdot \ldots \cdot Sign_1 \times \\ \times \delta_0 \cdot (Sign_0 - (-Sign_0))| = |2 \cdot \delta_0|. \end{aligned} \quad (4.13)$$

For an arbitrary challenge CH^{Ω} with $\Omega < 2^{N-1}$, MSB $= 0$ and $Sign_{N-1} = 1$. Then, according to (4.9),

$$\begin{aligned} \Delta_{A_N B_N}(CH^{\Omega}) = \delta_{N-1} + Sign_{N-2} \cdot \delta_{N-2} + \ldots + \\ + Sign_{N-2} \cdot Sign_{N-3} \cdot \ldots \cdot Sign_0 \cdot \delta_0. \end{aligned} \quad (4.14)$$

For the challenge $CH^{\Omega''}$, where $\Omega'' = \Omega + 2^{N-1}$, MSB $= 1$ and $Sign_{N-1} = -1$. Hence,

$$\begin{aligned} \Delta_{A_N B_N}(CH^{\Omega''}) = -(\delta_{N-1} + Sign_{N-2} \cdot \delta_{N-2} + \\ + \ldots + \ Sign_{N-2} \cdot Sign_{N-3} \cdot \ldots \cdot Sign_0 \cdot \delta_0). \end{aligned} \quad (4.15)$$

By comparing (4.14) and (4.15), it is apparent that

$$\Delta_{A_N B_N}(CH^{\Omega}) = -\Delta_{A_N B_N}(CH^{\Omega''}) \tag{4.16}$$

and their absolute difference is

$$|\Delta_{A_N B_N}(CH^{\Omega}) - \Delta_{A_N B_N}(CH^{\Omega''})| = 2 \cdot |\Delta_{A_N B_N}(CH^{\Omega})| \tag{4.17}$$

It can be deduced from (4.13)-(4.17) that inverting the LSB of an arbitrary challenge CH^{Ω} leads to only a small change in the total propagation delay difference and is outcome inconsequential whereas inverting the MSB can change the winner of the race. Furthermore, there is a non-zero probability of finding one or more challenges that can equalize the delays of two competing paths such that their final propagation delay difference $\Delta_{A_N B_N} \approx 0$. This is because the magnitudes and signs of δ_i for each $switch_i$ are different for different logic state of CH_i. Consequently, there is always a subset of challenges that cause the outputs of the arbiter to fall into the metastable state, i.e., $R = X$.

In order to obtain a stable response bit, the propagation delay difference should meet the following condition:

$$R = \begin{cases} 0 & \text{if } \Delta_{A_N B_N} \ll t_{low}, \\ 1 & \text{if } \Delta_{A_N B_N} \gg t_{high}. \end{cases} \quad . \tag{4.18}$$

The following conditions are valid for an "ideal" arbiter: $|t_{high}| = |t_{low}|$ and $t_{low}, t_{high} \to 0$. As a result, for an "ideal" arbiter, inversion of the LSB of a challenge has low probability of flipping the response bit but inversion of the MSB of a challenge has high probability of flipping the response bit. When the inversion of the MSB of a challenge does not lead to a bit flip of the response, it implies that $\Delta_{A_N B_N}$ has fallen into the metastable region of the arbiter. Due to geometric asymmetry of the devices, the metastable region of real silicon implementation of arbiter is not fixed and may offset from zero. Hence, $|t_{high}| \neq |t_{low}|$, $|t_{high}| > 0$ and $|t_{low}| > 0$, as evinced by the parametric model simulation of the implementation of a single DFF-based arbiter on a Nexys 4 DDR Artix-7 FPGA, which shows $t_{hold} \in [-0.104, 0.024]\ ns$ and $t_{setup} \in [0.154, 0.295]\ ns$.

4.3 Proposed Challenge Classification and Reliability Enhancement Algorithm

4.3.1 Delay model based classification of A-PUF challenges

Based on the delay model described above and known PUF testing technique [108], the challenge of an A-PUF can be classified according to its responses obtained by flipping either its LSB, MSB or both these bits simultaneously. For each challenge CH^{Ω}, a quadruple of responses $\{R_0, R_1, R_2, R_3\}$ is defined, where $R_0 = PUF(CH^{\Omega})$, $R_1 = PUF(CH^{\Omega'})$, $R_2 = PUF(CH^{\Omega''})$ and $R_3 = PUF(CH^{\Omega'''})$. A challenge CH^{Ω} is classified as a stable challenge if its quadruple meets the following requirement:

$$R_0 = R_1 = \overline{R_2} = \overline{R_3} \tag{4.19}$$

From (4.19), it is evident that all stable challenges can only have one of the two possible types of quadruple, i.e., $\{R_0, R_1, R_2, R_3\}$ is either {0, 0, 1, 1} or {1, 1, 0, 0}. All other challenges that have the quadruple $\{R_0, R_1, R_2, R_3\}$ different from {0, 0, 1, 1} and {1, 1, 0, 0} are classified as unstable challenges. As expected from the earlier propagation delay difference analysis among CH^{Ω}, $CH^{\Omega'}$, $CH^{\Omega''}$ and $CH^{\Omega'''}$, the stable challenges are linearly dependent. This means that with the knowledge of at least one CRP of a quadruple ((CH^{Ω}, R_0), ($CH^{\Omega'}$, R_1), ($CH^{\Omega''}$, R_2), ($CH^{\Omega'''}$, R_3)), it is possible to predict the remaining three CRPs of the quadruple. From security point of view, it is better to reduce the cardinality of the original CRP set of an A-PUF to a quarter by grouping and treating the four challenges CH^{Ω}, $CH^{\Omega'}$, $CH^{\Omega''}$ and $CH^{\Omega'''}$ as one single unique quadruple to increase the A-PUF resistance against modeling attack. After all, the A-PUF will still have an exponentially large number of unique challenges after this grouping.

Fig. 4.2 plots the differential arrival time at the arbiter $\Delta_{A_N B_N}$ versus the challenge. Depending on the challenge, $\Delta_{A_N B_N}$ can be either positive or negative. To separate the challenges with $R = 0, 1, X$ (see (4.1)) into three distinct regions, the challenges are sorted according to their differential delay $\Delta_{A_N B_N}$ along the axis of abscissas. For example, from the simulation of an FPGA implementation of 16-stage A-PUF, the differential path delays of the two endpoint challenges corresponding to $\alpha = 0010111101111111$ and $\beta = 1010111101111111$ are $-4.257\ ns$ and $3.778\ ns$, respectively. In this case, $\alpha = \beta''$. Clearly, the endpoint challenges and their differential path delays may vary from one FPGA to another.

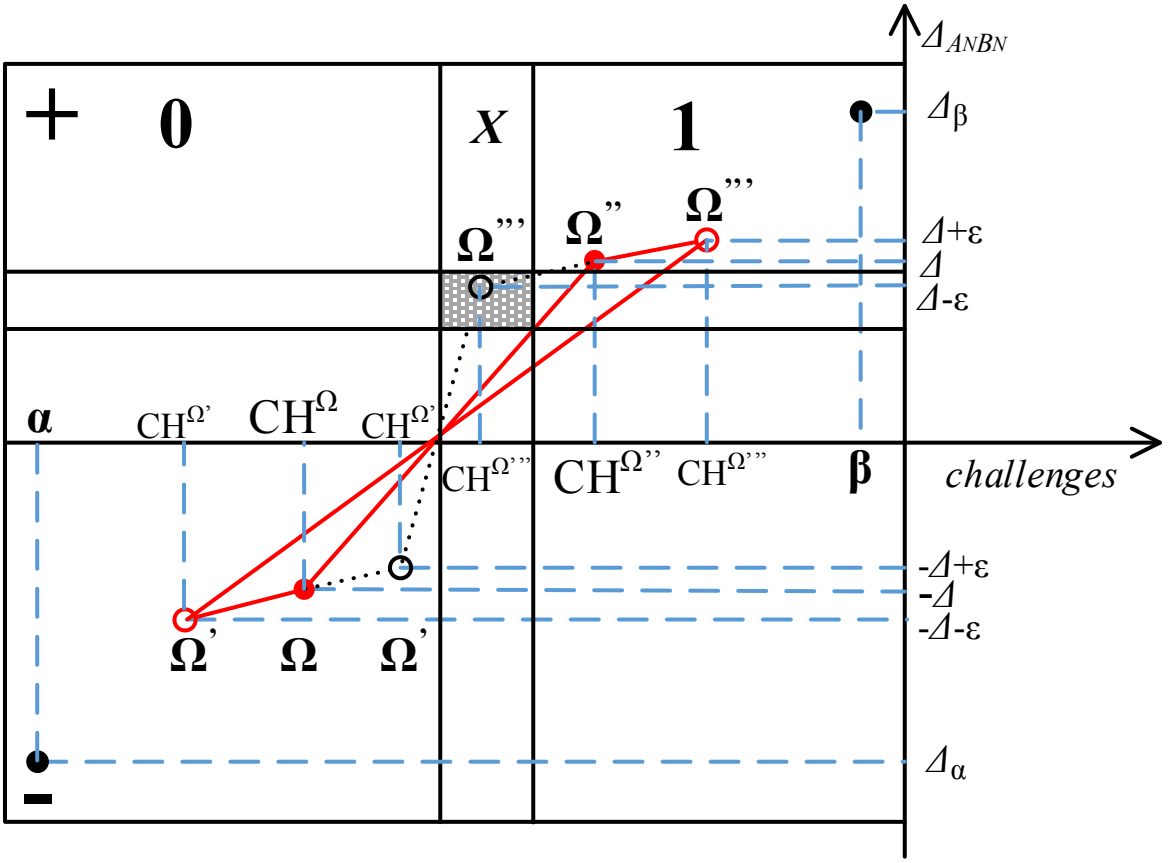
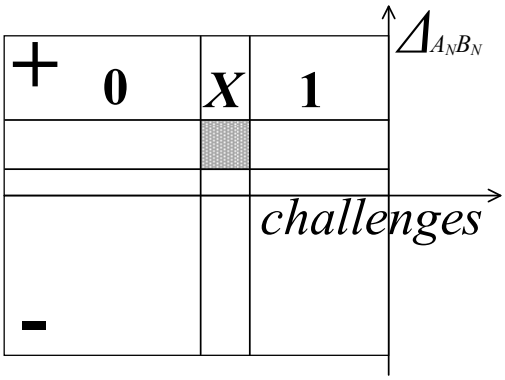


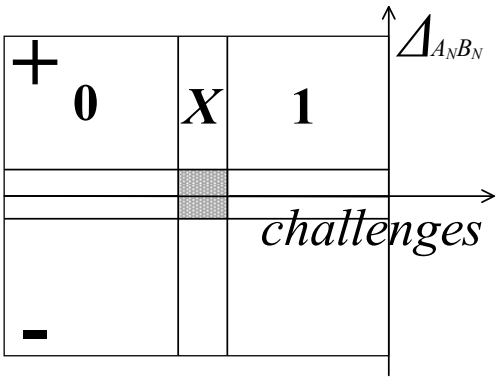
Figure 4.2: Illustration of quadruple challenge-response pair (CRP) for an arbitrary input challenge CH^{Ω}.

The challenges are thus divided into three bands in the horizontal axis based on their responses, which can be either a stable logic one state (1), a metastable state (X) or a stable logic zero state (0). Both the position and width of the X band in the $\Delta_{A_N B_N}$ axis (colored gray in Fig. 4.2) that demarcate the stable 0 and stable 1 response bands are dependent on the timing specifications, t_{low} and t_{high}, of the arbiter circuit. They may differ from one FPGA chip to another. The same arbiter circuit may also have subtly different t_{low} and t_{high} when it is configured onto different fabrics within the same FPGA chip.

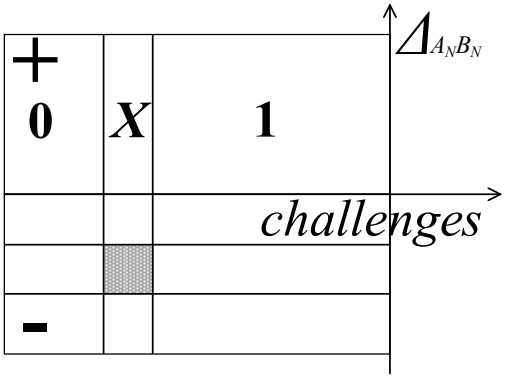
In general, the differential delays of the quadruple (CH^{Ω}, $CH^{\Omega'}$, $CH^{\Omega''}$, $CH^{\Omega'''}$) for an arbitrary challenge CH^{Ω} form approximately two congruent triangles with a common vertex lying on the horizontal axis of Fig. 4.2. Since the transition from



(a) Positive metastable region



(b) Bipolar metastable region



(c) Negative metastable region

Figure 4.3: Possible metastable region locations.

a challenge CH^{Ω} with $\Delta_{A_N B_N} = \Delta$ to $CH^{\Omega'}$ can result in either an increment or a decrement of Δ, the two possible pairs of congruent triangle are differentiated by solid and dotted lines in Fig. 4.2. Among the four types of input challenge transition, Ω to Ω', Ω'' to Ω''', Ω to Ω'' and Ω' to Ω''', the first two types do not cause a response bit flip in the case of a stable challenge and they may if the challenge is unstable. The last two types have a high probability of changing the polarity of differential delay to cause a response bit inversion. As shown in Fig. 4.2, even if the differential delay due to a change in challenge is large enough to alter its sign, the response can still be unstable if it falls into the metastable region (see the challenge Ω''').

Three main types of metastable band are depicted in Fig. 4.3. They are identified by the polarities of $\Delta_{A_N B_N}$ values that span the entire metastable band, which can

either be all positive, all negative or bipolar. In any case, all CRPs for the metastable state X are located in the rectangular box formed by the gray color interception region of the horizontal and vertical bands. Unfortunately, the exact values of $\Delta_{A_N B_N}$ for different input challenges are inaccessible and difficult to measure accurately after the A-PUF is implemented on FPGA. Nonetheless, the three different metastable regions depicted in Fig. 4.3 can be detected by the quadruple responses of {0, 0, 0, 0}, {1, 1, 1, 1} and {X, X, X, X}. Since the values of $\Delta_{A_N B_N}$ produced by the challenges, CH^{Ω} and $CH^{\Omega''}$, have the same magnitude but opposite sign (see (4.16)), the transition from CH^{Ω} to $CH^{\Omega''}$ have to cross the zero ordinate. Similarly for the responses produced by the challenges $CH^{\Omega'}$ and $CH^{\Omega'''}$. Hence, if there exists a challenge CH^{Ω} that produces a response quadruple {0, 0, 0, 0}, then all four corners of the two congruent triangles fall in the $R = 0$ plane. It implies that the challenges are located below metastable band and the $R = 0$ region is wider than the $R = 1$ region. This corresponds to the case of Fig. 4.3a. For this case, it is impossible to have the response quadruple {1, 1, 1, 1} and {X, X, X, X}. This is because the ordinates of logical one and metastable bands are both above zero, changing the sign of $\Delta_{A_N B_N}$ due to the transitions from CH^{Ω} to $CH^{\Omega''}$ and from $CH^{\Omega'}$ and $CH^{\Omega'''}$ will cause the response bit to flip since the challenges are ordered in ascending $\Delta_{A_N B_N}$. Based on the same argument, only {1, 1, 1, 1} but not {0, 0, 0, 0} and {X, X, X, X} can exist for the case of Fig. 4.3c, and response quadruple {X, X, X, X} can only be found in the case of Fig. 4.3b.

4.3.2 Reliability enhancement algorithm

As the location and width of metastable band are not fixed even for two identical arbiters implemented on the same FPGA chip, further filtering of stable challenges is needed to eliminate those that are more likely to have their response bits landed onto the metastable region. Specifically, the variations in operating conditions may cause δ_i of a few intermediate random switching elements $switch_i$ besides $switch_0$ and $switch_N$ to change in the direction that inverts their corresponding signs $Sign_i$ of (4.9). If these few intermediate sign inversions do not result in a sign inversion of the final differential delay $\Delta_{A_N B_N}$, then the response bit will be more reliable. Since the sign of δ_i (i.e., its coefficient in (4.9)) can be affected by a change in $Sign_j$ of any switching element $switch_j$ $\forall j \geq i$, and $Sign_j$ can be changed by changing its corresponding challenge bit CH_j, we can winnow out the susceptible challenges CH^{Ω} by flipping only a small number of randomly selected bits $i \in [1, N-2]$ of CH^{Ω} to test if it will result in a response bit flip.

```
1: procedure RECODE(Challenge)
2:     N ← number of stages
3:     Sign ← ∅
4:     Sign_challenge ← array of N " − "
5:     if Challenge[N − 1] = '1' then
6:         Sign ← " − "
7:     else
8:         Sign ← " + "
9:     end if
10:    Sign_challenge[N − 1] ← Sign
11:    index ← N − 2
12:    while index ≥ 0 do
13:        if Challenge[index] = '1' then
14:            Sign ← −Sign
15:        end if
16:        Sign_challenge[index] ← Sign
17:        index ← index − 1
18:    end while
19:    return Sign_challenge
20: end procedure
```

Figure 4.4: Challenge recoding algorithm.

Based on (4.9), every challenge CH^{Ω} can be represented by their coefficients $\prod_{i=j}^{N-1} Sign_i$ of δ_i $\forall i \in [0, N-1]$ and $Sign_i \in \{1, -1\}$. Fig. 4.4 describes the procedure to recode the binary representation of an N-bit $Challenge$ into a sequence $Sign\_challenge$ of N plus and minus signs.

For example, an 8-bit challenge $CH = \{1, 0, 1, 1, 0, 0, 1, 0\}$ will be recoded into $\{-, -, +, -, -, -, +, +\}$ corresponding to $\Delta_{A_8B_8} = -\delta_7 - \delta_6 + \delta_5 - \delta_4 - \delta_3 - \delta_2 + \delta_1 + \delta_0$. The proposed algorithm is further illustrated using this example as follows.

1. $Challenge = \{1, 0, 1, 1, 0, 0, 1, 0\}$ (line 1, parameter of function).
2. $N = 8$ (line 2).
3. $Sign = \emptyset$ (line 3).
4. $Sign\_challenge = \{" - ", " - ", " - ", " - ", " - ", " - ", " - ", " - "\}$ (line 4).
5. $Challenge[N-1] = Challenge[7] =$ '1', Then algorithm proceeds to line 6 as statement is true (line 5).
6. $Sign =$ "$-$". Then algorithm proceeds to line 10 as else part is skipped (line 6).

7. $Sign\_challenge[N-1] = Sign\_challenge[N-1] = Sign = \text{“}-\text{”}$ (line 10).

8. $index = N - 2 = 6$ (line 11).

9. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

10. $Challenge[index] = Challenge[6] =$ '0'. Then algorithm proceeds to line 16 as if is skipped (line 13).

11. $Sign\_challenge[index] = Sign\_challenge[6] = Sign = \text{“}-\text{”}$ (line 16).

12. $index = index - 1 = 5$. Then algorithm proceeds to line 12 to check loop condition again (line 17).

13. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

14. $Challenge[index] = Challenge[5] =$ '1'. Then algorithm proceeds to line 14 as if condition is satisfied (line 13).

15. $Sign = -Sign = \text{“}+\text{”}$ (line 14).

16. $Sign\_challenge[index] = Sign\_challenge[5] = Sign = \text{“}+\text{”}$ (line 16).

17. $index = index - 1 = 4$. Then algorithm proceeds to line 12 to check loop condition again (line 17).

18. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

19. $Challenge[index] = Challenge[4] =$ '1'. Then algorithm proceeds to line 14 as if condition is satisfied (line 13).

20. $Sign = -Sign = \text{“}-\text{”}$ (line 14).

21. $Sign\_challenge[index] = Sign\_challenge[4] = Sign = \text{“}-\text{”}$ (line 16).

22. $index = index - 1 = 3$. Then algorithm proceeds to line 12 to check loop condition again (line 17).

23. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

24. $Challenge[index] = Challenge[3] =$ '0'. Then algorithm proceeds to line 16 as if is skipped (line 13).

25. $Sign\_challenge[index] = Sign\_challenge[3] = Sign =$ " $-$ " (line 16).

26. $index = index - 1 = 2$. Then algorithm proceeds to line 12 to check loop condition again (line 17).

27. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

28. $Challenge[index] = Challenge[2] =$ '0'. Then algorithm proceeds to line 16 as if is skipped (line 13).

29. $Sign\_challenge[index] = Sign\_challenge[2] = Sign =$ " $-$ " (line 16).

30. $index = index - 1 = 1$. Then algorithm proceeds to line 12 to check loop condition again (line 17).

31. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

32. $Challenge[index] = Challenge[1] =$ '1'. Then algorithm proceeds to line 14 as if condition is satisfied (line 13).

33. $Sign = -Sign =$ " $+$ " (line 14).

34. $Sign\_challenge[index] = Sign\_challenge[1] = Sign =$ " $+$ " (line 16).

35. $index = index - 1 = 0$. Then algorithm proceeds to line 12 to check loop condition again (line 17).

36. $index \geq 0$. Then algorithm proceeds to line 13 as loop condition is satisfied (line 12).

37. $Challenge[index] = Challenge[0] =$ '0'. Then algorithm proceeds to line 16 as if is skipped (line 13).

38. $Sign\_challenge[index] = Sign\_challenge[0] = Sign =$ " $+$ " (line 16).

39. $index = index - 1 =$ -1. Then algorithm proceeds to line 12 to check loop condition again (line 17).

40. $index < 0$. Then algorithm proceeds to line 19 as loop condition is not satisfied (line 12).

41. Algorithm returns recoded challenge value $Sign\_challenge = \{\text{“}-\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}-\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}+\text{”}\}$ (line 19).

Using this representation, additional tests can be performed to filter out the less reliable challenges. This is achieved by randomly altering a small number, say k, of intermediate bits in a stable challenge CH^{Ω} to emulate the changes in δ_i of the corresponding switching elements $switch_i$, where $i \in [1, N-2]$. If the responses of CH^{Ω} due to all the 2^k possible combinations of bit alteration remain unchanged, then the challenge CH^{Ω} will be retained. This approach of identifying persistently stable challenges is described by the pseudo code in Fig. 4.5.

This algorithm is illustrated below using the same input challenge $CH = \{1, 0, 1, 1, 0, 0, 1, 0\}$.

1. $Challenge = \{1, 0, 1, 1, 0, 0, 1, 0\}$ (line 1, parameter of function).

2. $N = 8$ (line 2).

3. $k = 2$ (line 3).

4. $bit = \{0, 0\}$ (line 4).

5. $index = 0 \leq k - 1 = 1$. Then algorithm proceeds to line 6 as loop condition is satisfied (line 5).

6. $range = [1, N-2] = [1, 6]$ (line 6).

7. $switch\_index = \text{random}([1, N-2]) = \text{random}([1, 6]) = 3$ (line 7).

8. $bit[index] = bit[0] = switch\_index = 3$ (line 8).

9. $index = index+1 = 1$. Then algorithm proceeds to line 5 to check loop condition again (line 9).

10. $index = 1 \leq k - 1 = 1$. Then algorithm proceeds to line 6 as loop condition is satisfied (line 5).

11. $range = [1, N-2] = [1, 6]$ (line 6).

12. $switch\_index = \text{random}([1, N-2]) = \text{random}([1, 6]) = 5$ (line 7).

```
1: procedure CHECK(Challenge)
2:     N ← number of stages
3:     k ← number of intermediate stages
4:     bit ← array of k integers
5:     for index = 0 to k − 1 do
6:         range ← [1, N − 2]
7:         switch_index ← random(range)
8:         bit[index] ← switch_index
9:     end for
10:    Recoded_challenge ← Recode(Challenge)
11:    Challenge_copy ← Recoded_challenge
12:    R_0 ← PUF(Challenge)
13:    R_1 ← PUF(Challenge′)
14:    R_2 ← PUF(Challenge″)
15:    R_3 ← PUF(Challenge‴)
16:    if ({R_0, R_1, R_2, R_3} = {0, 0, 1, 1}) OR ({R_0, R_1, R_2, R_3} = {1, 1, 0, 0}) then
17:        for mask = 0 to 2^k − 1 do
18:            for index = 0 to k − 1 do
19:                if (mask >> index)&'1' then
20:                    if Recoded_challenge[bit[index]] = "+" then
21:                        Challenge_copy[bit[index]] ← "-"
22:                    else
23:                        Challenge_copy[bit[index]] ← "+"
24:                    end if
25:                    R_0^c ← PUF(Challenge_copy)
26:                    R_1^c ← PUF(Challenge_copy′)
27:                    R_2^c ← PUF(Challenge_copy″)
28:                    R_3^c ← PUF(Challenge_copy‴)
29:                    if ({R_0^c, R_1^c, R_2^c, R_3^c}) ≠ ({R_0, R_1, R_2, R_3}) then
30:                        return False
31:                    end if
32:                end if
33:            end for
34:        end for
35:    else
36:        return False
37:    end if
38:    return True
39: end procedure
```

Figure 4.5: Algorithm for filtering of stable challenges.

13. $bit[index] = bit[0] = switch\_index = 5$ (line 8).

14. $index = index{+}1 = 2$. Then algorithm proceeds to line 5 to check loop condition again (line 9).

15. $index = 2 > k - 1 = 1$. Then algorithm proceeds to line 10 as loop condition is not satisfied (line 5).

16. $Recoded\_challenge$ = Recode($Challenge$) = Recode($\{1, 0, 1, 1, 0, 0, 1, 0\}$) = $\{“-”, “-”, “+”, “-”, “-”, “-”, “+”, “+”\}$ (line 10).

17. $Challenge\_copy = Recoded\_challenge = \{“-”, “-”, “+”, “-”, “-”, “-”, “+”, “+”\}$ (line 11).

18. R_0 = PUF($Challenge$) = PUF($\{1, 0, 1, 1, 0, 0, 1, 0\}$) = 1 (line 12).

19. R_1 = PUF($Challenge'$) = PUF($\{1, 0, 1, 1, 0, 0, 1, 1\}$) = 1 (line 13).

20. R_2 = PUF($Challenge''$) = PUF($\{0, 0, 1, 1, 0, 0, 1, 0\}$) = 0 (line 14).

21. R_3 = PUF($Challenge'''$) = PUF($\{0, 0, 1, 1, 0, 0, 1, 1\}$) = 0 (line 15).

22. Note. If the response values differ from $\{1, 1, 0, 0\}$ or $\{0, 0, 1, 1\}$, the challenge is considered “weak” and function returns False after if statement (line 16). For illustrative purpose we keep challenge “strong” to illustrate all possible cases covered by this algorithm.

23. $\{R_0, R_1, R_2, R_3\} = \{1, 1, 0, 0\}$. Then algorithm proceeds to line 17 as if statement is satisfied (line 16).

24. $mask = 0 \leq 2^k - 1 = 3$. Then algorithm proceeds to line 18 as loop condition is satisfied (line 17).

25. $index = 0 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

26. $mask >> index = 0\&'1' = False$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 19).

27. $index = 1 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

28. $mask >> index = 0\&'1' = False$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 19).

29. $index = 2 > k - 1 = 1$. Then algorithm proceeds to line 17 as loop condition is not satisfied (line 18).

30. $mask = 1 \leq 2^k - 1 = 3$. Then algorithm proceeds to line 18 as loop condition is satisfied (line 17).

31. $index = 0 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

32. $mask >> index = 1\&'1' = True$. Then algorithm proceeds to line 20 as if condition is satisfied (line 19).

33. $Recoded\_challenge[bit[index]] = Recoded\_challenge[3] = \text{“}-\text{”} \neq \text{“+”}$. Then algorithm proceeds to line 23 as if condition is not satisfied (line 20).

34. $Challenge\_copy[bit[index]] = Challenge\_copy[3] =' +'$. As a result $Challenge\_copy$ $= \{\text{“}-\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}+\text{”}\} = \{1, 0, 1, 1, 1, 1, 1, 0\}$ (line 23).

35. $R_0^c = \text{PUF}(Challenge\_copy) = \text{PUF}(\{1, 0, 1, 1, 1, 1, 1, 0\}) = 1$ (line 25).

36. $R_1^c = \text{PUF}(Challenge\_copy') = \text{PUF}(\{1, 0, 1, 1, 1, 1, 1, 1\}) = 1$ (line 26).

37. $R_2^c = \text{PUF}(Challenge\_copy'') = \text{PUF}(\{0, 0, 1, 1, 1, 1, 1, 0\}) = 0$ (line 27).

38. $R_3^c = \text{PUF}(Challenge\_copy''') = \text{PUF}(\{0, 0, 1, 1, 1, 1, 1, 1\}) = 0$ (line 28).

39. Note. If the response values differ from $\{1, 1, 0, 0\}$, the challenge is considered “strong” (but not “persistently strong”) and function returns False after if statement (line 29). For illustrative purpose we keep challenge “persistently strong” to illustrate all possible cases covered by this algorithm.

40. $index = 1 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

41. $mask >> index = 0\&'1' = False$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 19).

42. $index = 2 > k - 1 = 1$. Then algorithm proceeds to line 17 as loop condition is not satisfied (line 18).

43. $mask = 2 \leq 2^k - 1 = 3$. Then algorithm proceeds to line 18 as loop condition is satisfied (line 17).

44. $index = 0 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

45. $mask >> index = 2\&'1' = False$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 19).

46. $index = 1 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

47. $mask >> index = 1\&'1' = True$. Then algorithm proceeds to line 20 as if condition is satisfied (line 19).

48. $Recoded\_challenge[bit[index]] = Recoded\_challenge[5] =$ " + " = "+". Then algorithm proceeds to line 21 as if condition is satisfied (line 20).

49. $Challenge\_copy[bit[index]] = Challenge\_copy[5] =' -'$. As a result $Challenge\_copy$ = {" − ", " − ", " − ", " − ", " − ", " − ", " + ", " + "} = {1, 0, 0, 0, 0, 0, 1, 0} (line 23).

50. $R_0^c = \text{PUF}(Challenge\_copy) = \text{PUF}(\{1, 0, 0, 0, 0, 0, 1, 0\}) = 1$ (line 25).

51. $R_1^c = \text{PUF}(Challenge\_copy') = \text{PUF}(\{1, 0, 0, 0, 0, 0, 1, 1\}) = 1$ (line 26).

52. $R_2^c = \text{PUF}(Challenge\_copy'') = \text{PUF}(\{0, 0, 0, 0, 0, 0, 1, 0\}) = 0$ (line 27).

53. $R_3^c = \text{PUF}(Challenge\_copy''') = \text{PUF}(\{0, 0, 0, 0, 0, 0, 1, 1\}) = 0$ (line 28).

54. $\{R_0^c, R_1^c, R_2^c, R_3^c\} = \{R_0, R_1, R_2, R_3\} = \{1, 1, 0, 0\}$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 29).

55. $index = 2 > k - 1 = 1$. Then algorithm proceeds to line 17 as loop condition is not satisfied (line 18).

56. $mask = 3 \leq 2^k - 1 = 3$. Then algorithm proceeds to line 18 as loop condition is satisfied (line 17).

57. $index = 0 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

58. $mask >> index = 3\&'1' = True$. Then algorithm proceeds to line 20 as if condition is satisfied (line 19).

59. $Recoded\_challenge[bit[index]] = Recoded\_challenge[3] = \text{“}-\text{”} \neq \text{“+”}$. Then algorithm proceeds to line 23 as if condition is not satisfied (line 20).

60. $Challenge\_copy[bit[index]] = Challenge\_copy[3] =' +'$. As a result $Challenge\_copy = \{\text{“}-\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}+\text{”}\} = \{1, 0, 1, 1, 1, 1, 1, 0\}$ (line 23).

61. $R_0^c = \text{PUF}(Challenge\_copy) = \text{PUF}(\{1, 0, 1, 1, 1, 1, 1, 0\}) = 1$ (line 25).

62. $R_1^c = \text{PUF}(Challenge\_copy') = \text{PUF}(\{1, 0, 1, 1, 1, 1, 1, 1\}) = 1$ (line 26).

63. $R_2^c = \text{PUF}(Challenge\_copy'') = \text{PUF}(\{0, 0, 1, 1, 1, 1, 1, 0\}) = 0$ (line 27).

64. $R_3^c = \text{PUF}(Challenge\_copy''') = \text{PUF}(\{0, 0, 1, 1, 1, 1, 1, 1\}) = 0$ (line 28).

65. $\{R_0^c, R_1^c, R_2^c, R_3^c\} = \{R_0, R_1, R_2, R_3\} = \{1, 1, 0, 0\}$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 29).

66. $index = 1 \leq k - 1 = 1$. Then algorithm proceeds to line 19 as loop condition is satisfied (line 18).

67. $mask >> index = 1\&'1' = True$. Then algorithm proceeds to line 20 as if condition is satisfied (line 19).

68. $Recoded\_challenge[bit[index]] = Recoded\_challenge[5] = \text{“}+\text{”} = \text{“+”}$. Then algorithm proceeds to line 21 as if condition is satisfied (line 20).

69. $Challenge\_copy[bit[index]] = Challenge\_copy[5] =' -'$. As a result $Challenge\_copy = \{\text{“}-\text{”}, \text{“}-\text{”}, \text{“}-\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}-\text{”}, \text{“}+\text{”}, \text{“}+\text{”}\} = \{1, 0, 0, 0, 1, 1, 1, 0\}$ (line 23).

70. $R_0^c = \text{PUF}(Challenge\_copy) = \text{PUF}(\{1, 0, 0, 0, 1, 1, 1, 0\}) = 1$ (line 25).

71. $R_1^c = \text{PUF}(Challenge\_copy') = \text{PUF}(\{1, 0, 0, 0, 1, 1, 1, 1\}) = 1$ (line 26).

72. $R_2^c = \text{PUF}(Challenge\_copy'') = \text{PUF}(\{0, 0, 0, 0, 1, 1, 1, 0\}) = 0$ (line 27).

73. $R_3^c = \text{PUF}(Challenge\_copy''') = \text{PUF}(\{0, 0, 0, 0, 1, 1, 1, 1\}) = 0$ (line 28).

74. $\{R_0^c, R_1^c, R_2^c, R_3^c\} = \{R_0, R_1, R_2, R_3\} = \{1, 1, 0, 0\}$. Then algorithm proceeds to line 18 as if condition is not satisfied (line 29).

75. $index = 2 > k - 1 = 1$. Then algorithm proceeds to line 17 as loop condition is not satisfied (line 18).

76. $mask = 4 > 2^k - 1 = 3$. Then algorithm proceeds to line 38 as loop condition is not satisfied (line 17).

77. As a result algorithm returns True, which means that $Challenge = \{1, 0, 1, 1, 0, 0, 1, 0\}$ is "persistently strong" (line 38).

No additional hardware is required to select persistently stable challenges except some timing overhead and a reduction of the effective length of an A-PUF. The algorithm described in Fig. 4.5 has a time complexity of $O(2^k)$. It was empirically determined from the FPGA implementation of 128-stage A-PUF that $k = 3 \sim 4$ is sufficient and rarely does it has to exceed 10% of the challenge bit length to asymptotically converge the reliability to 100%. Being a strong PUF, even if the effective length of A-PUF challenge is reduced from 128 bits to 116 bits, it still has an exponentially large CRP space. By using only persistently stable challenges for either secret key generation or device authentication, the sacrifice is trivial compared with the overheads of error correction code and helper data. These overheads are reduced substantially, if not completely eliminated by the proposed algorithm.

4.4 Experimental Results and Discussions

4.4.1 Simulation Results

A 16-stage classical A-PUF (with DFF based arbiter) design was coded in VHDL and simulated by Vivado Design Suite 2016.2 using the post-place and route model for Xilinx Artix-7 FPGA [109]. Due to the computation time of software simulator, $N = 16$ in order to exhaustively simulate all possible challenges and collect their responses with and without delay perturbance for analysis within a realistic time span. The propagation delay difference $\Delta_{A_N B_N}$ between the two paths traversed by the test pulse for each of the $2^N = 65536$ challenges are recorded. The complete set of challenge-response pairs (CRPs) is regrouped into quadruples of $((CH^{\Omega}, R_0), (CH^{\Omega'}, R_1), (CH^{\Omega''}, R_2), (CH^{\Omega'''}, R_3))$ as defined in Subsection 4.3.1 to analyze stable and unstable challenges.

The propagation delay difference $\Delta_{A_N B_N}$ plotted against the challenges sorted in ascending order of $\Delta_{A_N B_N}$ is shown in Fig. 4.6. Based on the response values, the challenges are partitioned into three regions: logical zero state (0), metastable

state (X) and logical one state (1). The arbiter enters into metastability when the propagation delay time difference falls within the range of $[-0.022\ ,\ 0.154]\ ns$, which marks the boundaries of metastable band (colored red). The response is non-linearly dependent on $\Delta_{A_N B_N}$ due to the varying magnitude and sign of δ_i for each arbiter stage. Three representative challenge quadruples from Fig. 4.6 are singled out to demonstrate the persistently stable, stable and unstable challenges.

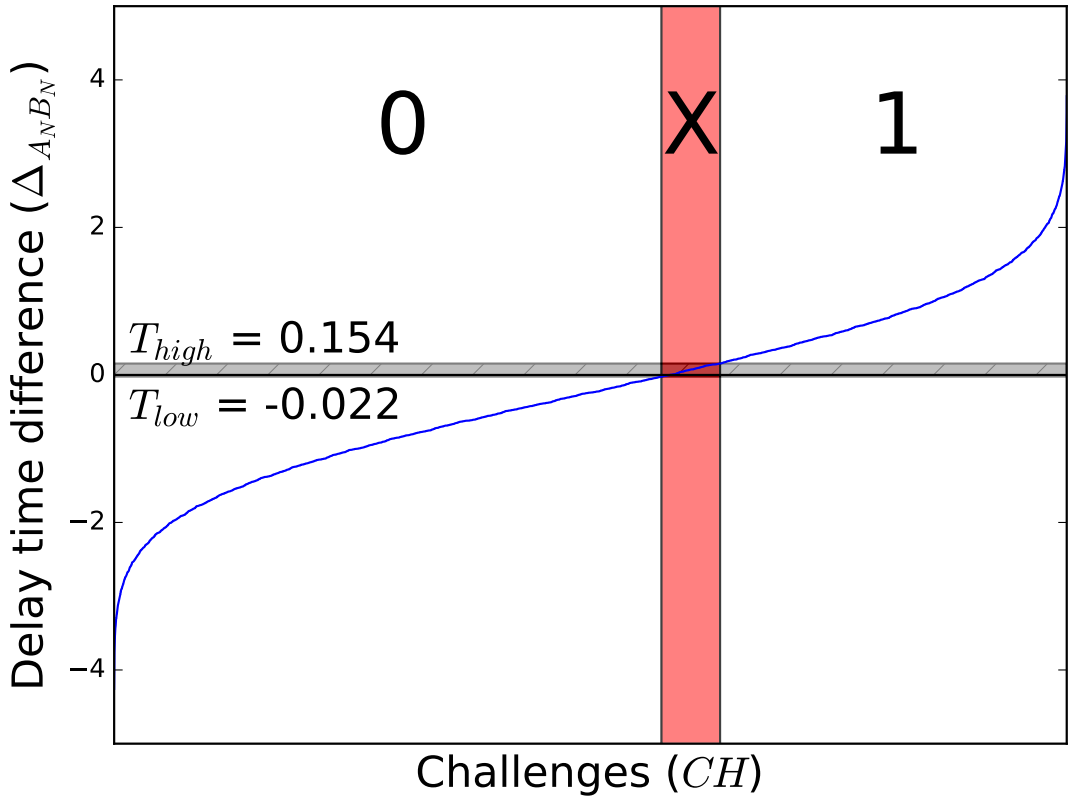
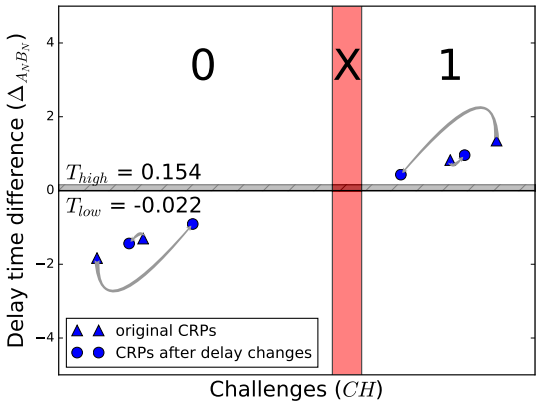
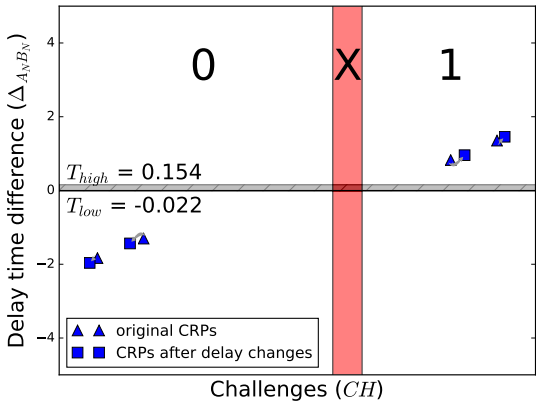


Figure 4.6: Challenge response pairs ordered by $\Delta A_N B_N$ value.

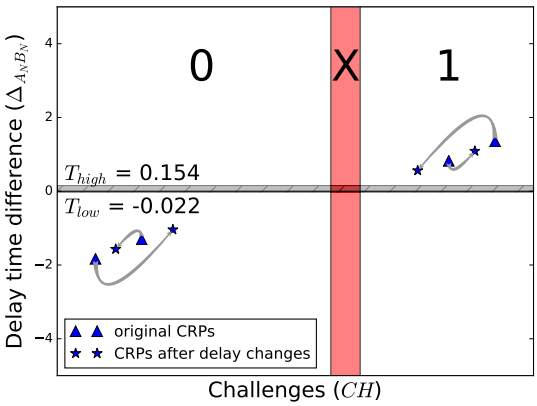
A persistently stable quadruple challenge {0000011011001000, 0000011011001001, 100001101100 1000, 1000011011001001} is illustrated in Fig. 4.7. Its corresponding quadruple propagation delay difference is $\{-1.304,\ -1.832,\ 0.825,\ 1.353\}\ ns$. This quadruple challenge is stable since its quadruple response is {0, 0, 1, 1}. Furthermore, its CRPs are located far from the metastable band. A small change in the delay differences of δ_1, δ_2 and both δ_1 and δ_2 will not change its stable response bit. The resultant $\Delta_{A_N B_N}$ due to these changes correspond to different combinations of change in CH_1 and CH_2 of the quadruple are shown in Fig. 4.7a, Fig. 4.7b and Fig. 4.7c.



(a) CRPs after changing δ_1



(b) CRPs after changing δ_2

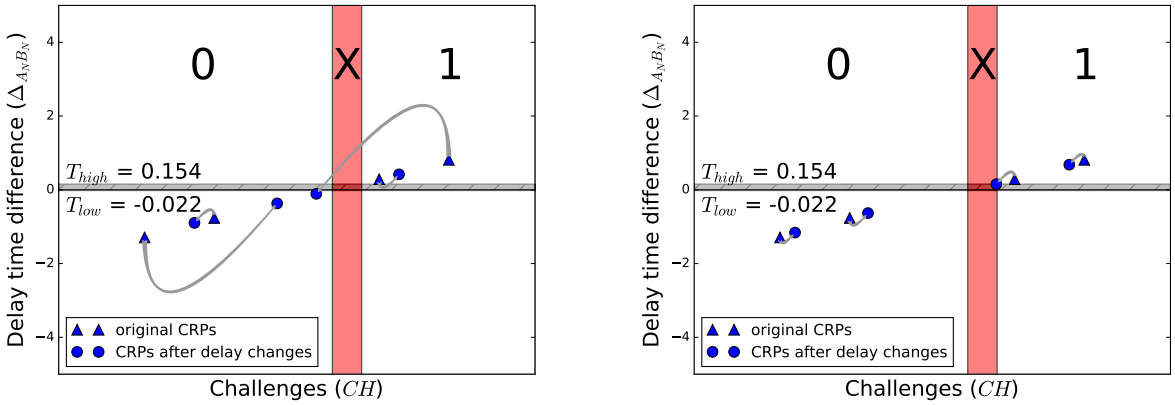


(c) CRPs after changing both δ_1 and δ_2

Figure 4.7: Small delay perturbance on a persistently stable challenge.

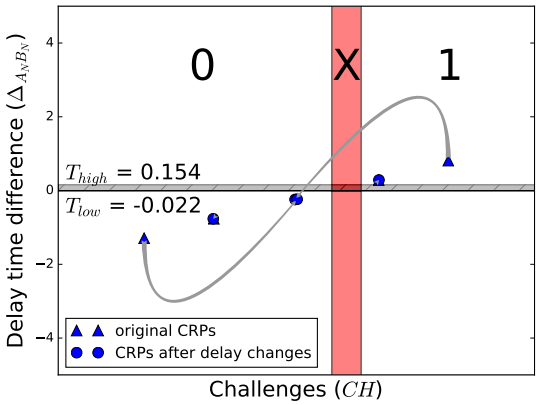
Another stable quadruple challenge {0000000000000110, 0000000000000111, 1000 000000000110, 1000000000000111} is illustrated in Fig. 4.8. It has a quadruple response of {1, 1, 0, 0}. Unlike the previous example, by introducing small variations into the propagation delay difference of some switching elements, the response bit to this challenge will flip. The response quadruple change to {1, **0**, 0, 0} with a slight deviation of either δ_1 or δ_2 (see Fig. 4.8a and Fig. 4.8b) or to {$\boldsymbol{X}$, 1, 0, 0} with a slight deviation of both δ_1 and δ_2 (see Fig. 4.8c). The propagation delay differences of this quadruple challenge are {0.285, 0.813, −0.764, −1.292} ns. The first challenge 0000000000000110 of the quadruple is close to the edge of metastable band, making its response bit susceptible to change when there is a polarity change in the differential delays of a small number of intermediate stages. These potentially unstable

challenges have been successfully detected and discarded by our algorithm in Fig. 4.5.



(a) CRPs after changing δ_1

(b) CRPs after changing δ_2

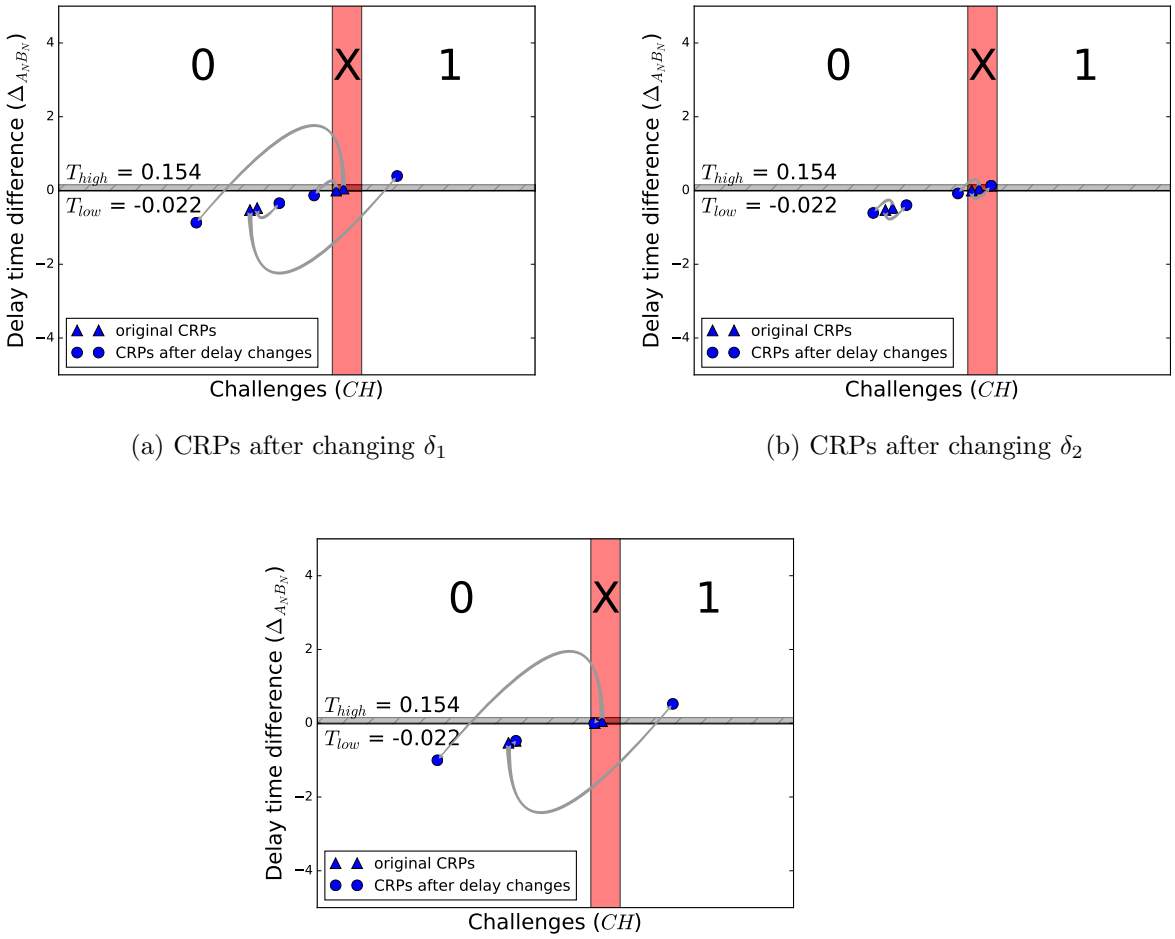


(c) CRPs after changing both δ_1 and δ_2

Figure 4.8: Response bit flipping due to small perturbance on a stable challenge.

The unstable challenge is illustrated in Fig. 4.9 by the challenge quadruple {0000000000001010, 0000000000001011, 1000000000001010, 1000000000001011}. Its corresponding differential delays are $\{-0.477, 0.051, -0.002, -0.530\}$ ns, which result in a response quadruple of $\{0, X, X, 0\}$. It is evident that a majority of response bits in this quadruple are prone to flipping when a small change in delay is introduced into one or more intermediate stages, as exemplified by the change of response quadruple to $\{0, \mathbf{0}, \mathbf{0}, \mathbf{1}\}$ in Fig. 4.9a, $\{0, \mathbf{0}, X, \mathbf{1}\}$ in Fig. 4.9b and $\{0, \mathbf{0}, X, 0\}$ in Fig. 4.9c due to the change in δ_1, δ_2 and both δ_1 and δ_2, respectively.

Out of the $2^{N-2} = 16384$ challenge quadruples, 67% are stable and 33% are unstable. So only 16.8% of the original challenges of a 16-stage A-PUF are useful for secret



(a) CRPs after changing δ_1

(b) CRPs after changing δ_2

(c) CRPs after changing both δ_1 and δ_2

Figure 4.9: Small delay perturbance on a unstable challenge.

key generation. This analysis further corroborates the practical limitation of short A-PUF due to the presence of unstable challenges with highly unreliable responses. In practice, A-PUF with at least 128 stages is used so that there are still a large number of useful challenges after filtering out the unstable and less reliable challenges. Some unstable challenges can be corrected by identifying and recoding the metastable states using the method in [110]. For example, the response quadruples $\{0, 0, X, 1\}$ and $\{0, 0, 1, X\}$ can be easily corrected to $\{0, 0, 1, 1\}$ and the response quadruples $\{X, 1, 0, 0\}$ and $\{1, X, 0, 0\}$ can be easily corrected to $\{1, 1, 0, 0\}$. Based on the raw data obtained from our simulation, around 20% of the response quadruples are located near the metastable band. These response quadruples include $\{0, 1, 0, 0\}$, $\{1, 0, 0, 0\}$, $\{0, 0, 1, 0\}$, $\{0, 0, 0, 1\}$, $\{0, 0, 0, X\}$, $\{0, 0, X, 0\}$, $\{0, X, 0, 0\}$, $\{0, X, X, 0\}$,

$\{X, 0, 0, 0\}$ and $\{X, 0, 0, X\}$. Apparently, there are more logical zero than logical one and metastable response bits (see Fig. 4.6). Therefore, these quadruple responses can be suitably corrected to $\{1, 1, 0, 0\}$ to make the response bit distribution more uniform. Unfortunately, the location of metastable band may vary due to the aperture time of physically implemented arbiter, which can complicate their detection. Currently, no approach has differentiated the apparently stable challenges located near the metastable band from the truly stable challenges that are located further away from the metastable band except our proposed algorithm in Fig. 4.5. Besides, to make full use of all quadruple challenges so classified, the unstable challenges can be utilized for true random number generator (TRNG) implementation.

4.4.2 FPGA implementation tests

4.4.2.1 Setup for data collection and analysis

A 130-stage A-PUF with SR latch based arbiter has been physically implemented in two different types of FPGA chip, namely the Zynq-7000 all programmable SoC XC7Z045 housed in ZC706 evaluation board [111] and the Xilinx Artix-7 FPGA housed in Nexys 4 DDR evaluation board [112]. The SR latch based arbiter has been chosen to enable identification of metastable states by the method proposed in [110]. Due to the cost of ZC706, one ZC706 and 20 Nexys 4 DDR boards are bought for the physical implementation tests. The high-end Zynq-7000 is used to demonstrate that the proposed challenge classification and filtering algorithms are applicable to A-PUF implemented on FPGA devices with more advanced manufacturing process technologies, configurable fabrics and denser interconnections. To test the uniqueness of the same A-PUF within the same FPGA chip, 8 identical 130-stage A-PUFs have been implemented on every FPGA chip.

To facilitate the application and classification of challenges and the collection of responses, the same efficient test setup as depicted in Fig. 4.10 [113] is adopted for the communication between a Host (PC) and the FPGA chip under test (CUT). A pseudo random M-sequence generator is implemented on the Host to produce weakly correlated 128-bit challenges. The initial challenge (C_{init}) generated by the Host is input to the Finite State Machine (FSM) of the CUT. The FSM wraps each challenge C_{init} by adding a leading binary bit and a trailing binary bit to make up a 130-bit challenge C_{ext} for input to the A-PUF. The quadruple $\{0 \parallel C_{init} \parallel 0, 0 \parallel C_{init} \parallel 1, 1 \parallel C_{init} \parallel 0, 1 \parallel C_{init} \parallel 1\}$ corresponds to $\{\Omega, \Omega', \Omega'', \Omega'''\}$ used for challenge classification, where $\parallel$ denotes string concatenation. The FSM also controls the number of times

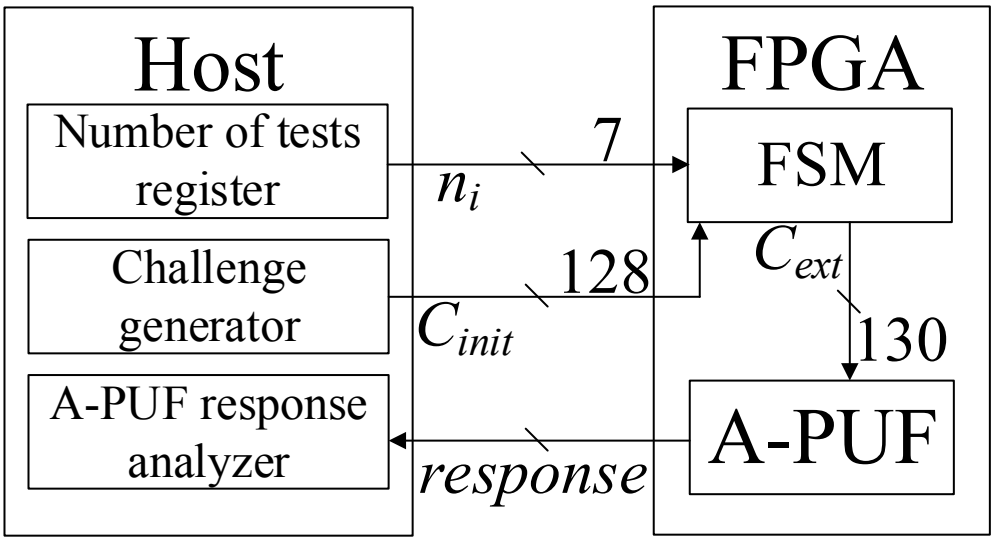


Figure 4.10: Overview of experimental setup.

each challenge is evaluated by the A-PUF for reliability test. The number of tests n_i stored in the register of the Host can be reprogrammed for different test requirements. The n_i quadruples of response $\{R_0, R_1, R_2, R_3\}$ produced by the A-PUF to the same challenge $\{0 \parallel C_{init} \parallel 0, 0 \parallel C_{init} \parallel 1, 1 \parallel C_{init} \parallel 0, 1 \parallel C_{init} \parallel 1\}$ will be transmitted to the Host. The response analyzer of the Host will classify the challenges as stable or unstable according to the set of response quadruples. The advantage of this setup is that no re-synthesis is required for each run even if there is a change in the rules for the generation of challenges, the number of challenges n_c or the value of k used for the filtering of challenges. Hence the configuration of A-PUF can remain the same for different combinations of parametric variation.

4.4.2.2 Distribution of stable and metastable states

n_c = 10,000 challenges were generated and the challenges were fed n_i = 100 times into the eight PUF instances implemented on the Zynq-7000 SoC. The results show that 92% of the quadruple responses are stable, i.e., they are {0, 0, 1, 1} and {1, 1, 0, 0} and only 8% of them are unstable and belong to one of { 1, 1, 0, 1}, {1, 1, 0, 0}, {1, 1, 1, 1}, {1, 1, X, 1}, {1, 1, X, X}, {1, X, 1, 1} and {X, X, 1, 1}. The response quadruple {1, 1, 1, 1} implies that logical one region is wider than logical zero region and the entire metastable band falls in the negative half plane of $\Delta_{A_N B_N}$ as in Fig. 4.3c.

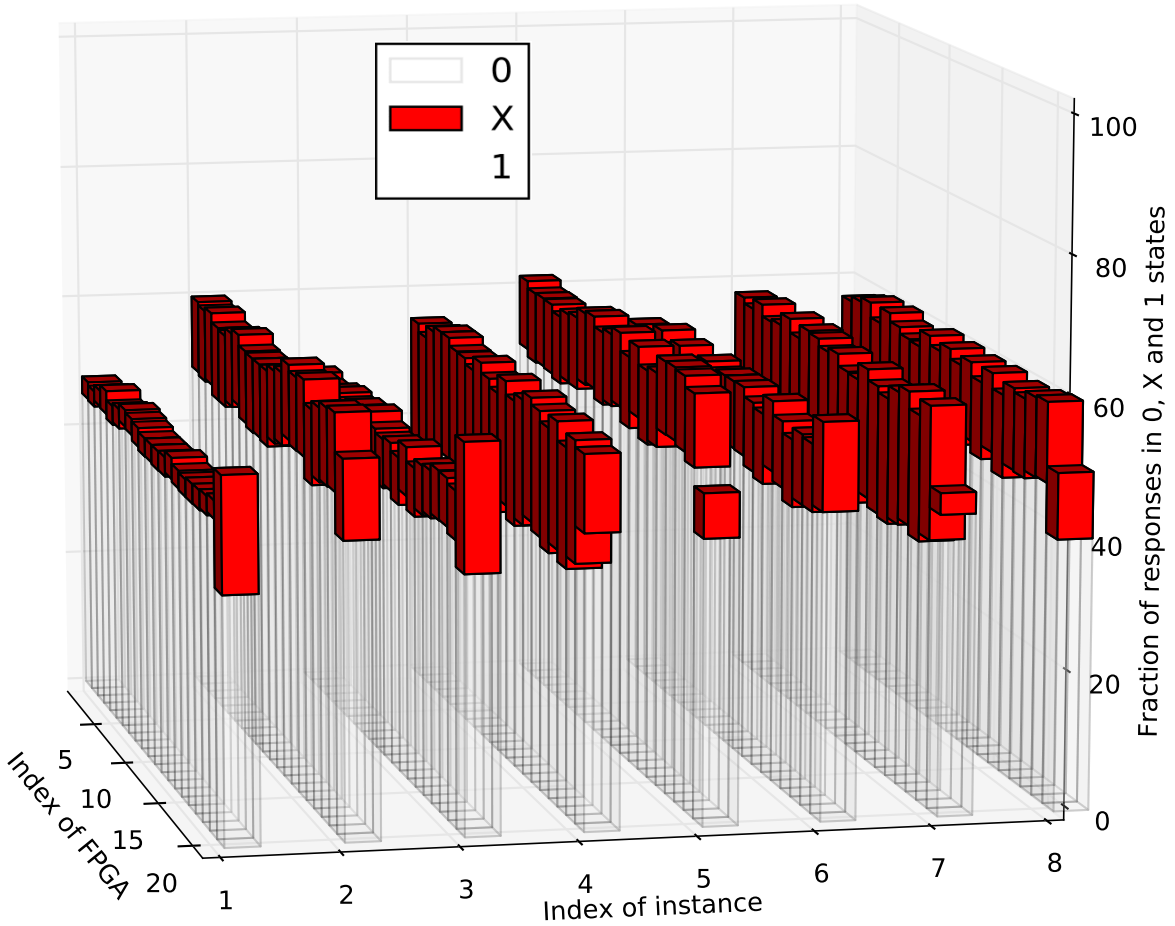


Figure 4.11: Distribution of 0, 1 and X bands for different FPGA and PUF instances.

Our experiment results show that the location of metastable band along the $\Delta_{A_N B_N}$ axis may not be fixed for different types of FPGA chip. Even different PUF instances implemented within the same FPGA chip may have different locations of metastable band. For example, of the eight instances of PUF implemented on Zynq-7000 SoC, three produce {0, 0, 0, 0}, two produce {1, 1, 1, 1} and three produce $\{X, X, X, X\}$ as response quadruples to some challenges. Fortunately, a specific instance of PUF implemented in the same type of FPGA chip has approximately the same location of metastable band. This is verified by the similar distribution of response quadruples among the eight instances of PUF implemented in 20 different Xilinx Artix-7 FPGA chips. For ZC706 the distribution is still uniform, but varies from component to component. This is shown in Fig. 4.11. To avoid cluttering the diagram, the logical one band in each bar is not shown. It is understood that it takes the fraction from the upper edge of metastable band to the maximum scale of 100. The first 20 FPGA indexes are Xilinx Artix-7 chips and index 21 is ZC706 chip.

For example, the second instance in Artix-7 FPGA has the entire metastable band falls in the positive half plane (see Fig. 4.3a) whereas the second instance in ZC706 has bipolar metastable band (see Fig. 4.3b). The distribution of response quadruples confirms the existence of different locations of metastable band and their locations and widths can be characterized by the type of FPGA chip.

4.4.3 Reliability test

It was confirmed by our earlier experiments [110] that FPGA implementation of classical 128-stage A-PUF has average reliability of only 0.577. This is far from the acceptable BER requirement of 10^{-6} based on the standard of BER [114] required for a 128-bit key generator.

Ostensibly, both minimum reliability (S_{min}) and average (S_{avg}) reliability are equally meaningful for benchmarking PUFs. However, for stable responses {1, 1, 0, 0} and {0, 0, 1, 1}, $S_{avg} = 0.999$ and $S_{min} = 0.762$, whereas for the most frequently appeared unstable responses {1, 1, X, X}, {1, 1, 1, 1} and {X, X, 1, 1}, $S_{avg} = 0.976$ and $S_{min} = 0.775$. This is counter-intuitive as the stable responses do not give significant enhancement in terms of S_{avg} and even worse in terms of S_{min}. In actual fact, S_{min} can be easily influenced by a few unreliable challenges even if the remaining challenges have 100% reliability. To account for this paradox, the following probability P_{stable} of stable response is also considered to discriminate the outliers.

$$P_{stable} = \frac{K_{stable}}{K} \tag{4.20}$$

where K is the total number of CRPs generated and K_{stable} is the number of responses with $S(CH) = 1$.

Table 4.1, which is also presented in [113], shows that response quadruples printed in bold font have much greater probability of being stable despite having comparable S_{min} values as unstable responses. Therefore, P_{stable} enables persistently stable challenges to be identified and additionally filtered to achieve an almost perfect A-PUF reliability of 1.0.

To use P_{stable} to gauge the reliability of filtered stable challenges, $K = 3000$ quadruple CRP sets have been generated and tested $E = 100$ times. To check the effectiveness of the proposed challenge filtering algorithm, all combinations of challenge bit flips to $k = 0$, 1, 2, 3 and 4 arbitrary intermediate stages have been checked. As shown in Fig. 4.12, both S_{avg} and S_{min} reach 1.0 along with P_{stable} when k is increased to 4. Thus, $2^k = 16$ additional checks are required to generate a set of 100% reliable

Table 4.1: Reliability of different A-PUF response quadruples.

$\{R_0, R_1, R_2, R_3\}$	S_{avg}	S_{min}	P_{stable}
{1, 1, 0, 0}	**0.999**	**0.758**	**0.991**
{0, 0, 1, 1}	**0.999**	**0.768**	**0.991**
$\{1, 1, X, X\}$	0.972	0.753	0.651
$\{X, X, 1, 1\}$	0.976	0.775	0.669
$\{1, 1, 1, 1\}$	0.944	0.758	0.279
$\{0, 1, 1, 1\}$	0.778	0.778	0.000
$\{1, 1, 0, 1\}$	0.740	0.733	0.000
$\{1, 1, X, 1\}$	0.774	0.763	0.000
$\{1, 1, 1, 0\}$	0.753	0.753	0.000
$\{1, 1, 1, X\}$	0.776	0.735	0.000
$\{X, 1, 1, 1\}$	0.766	0.755	0.000
$\{1, X, 1, 1\}$	0.741	0.718	0.000

responses by pruning around 30% of challenges. This result shows that the proportion of 100% reliable challenges obtained by physical implementation is comparable to the number of persistently stable challenges obtained from our simulation.

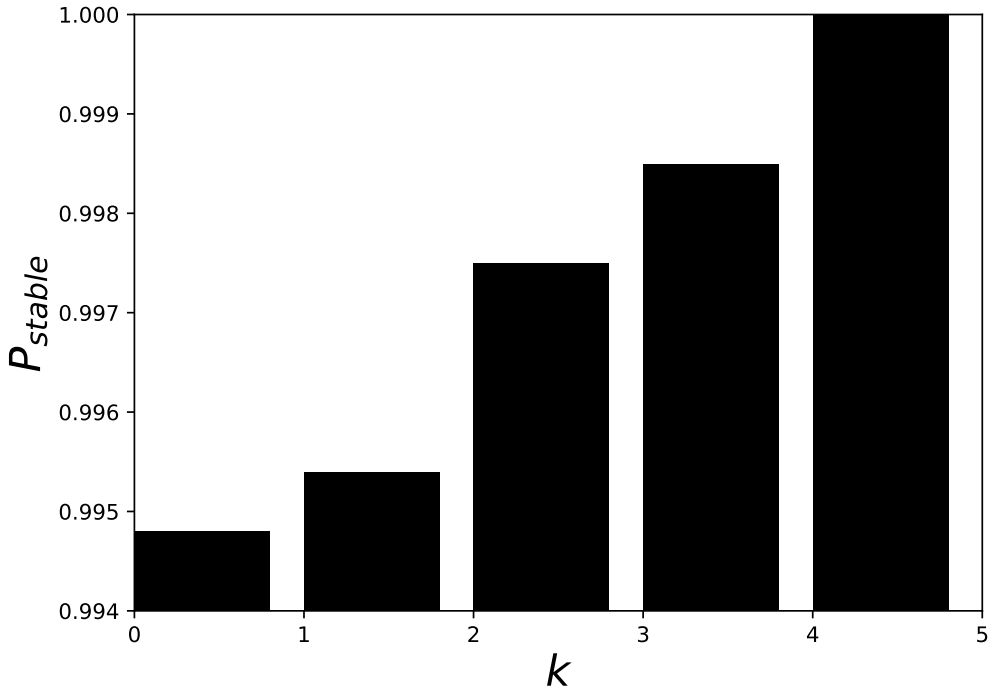
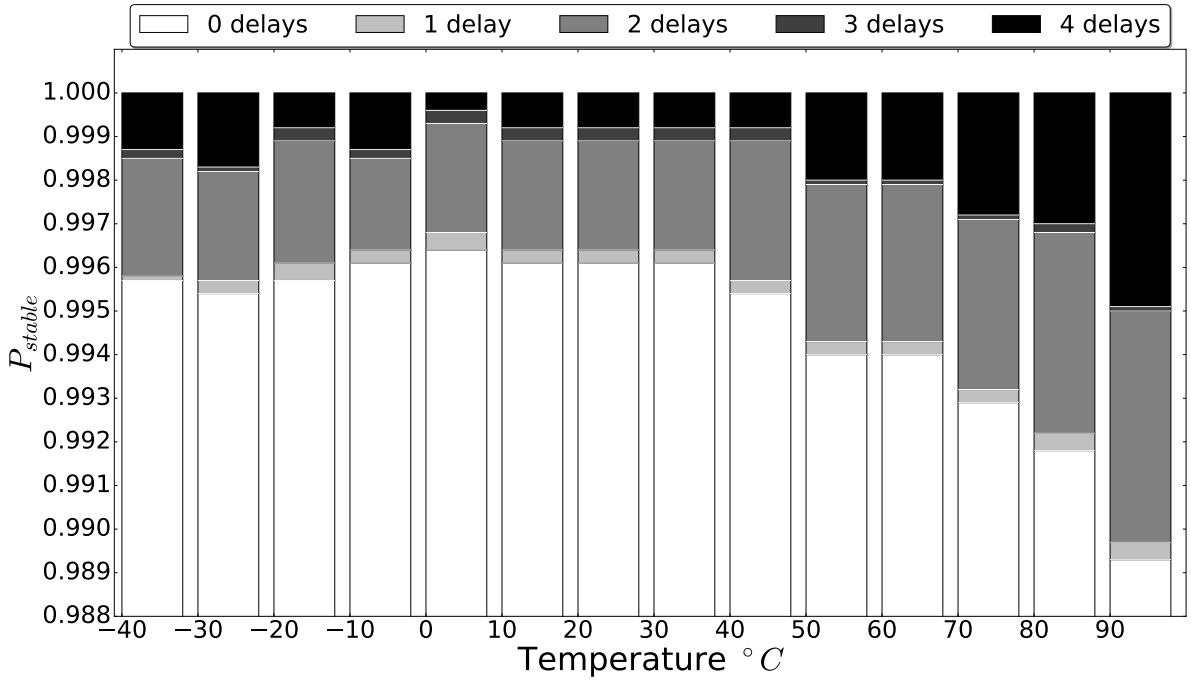
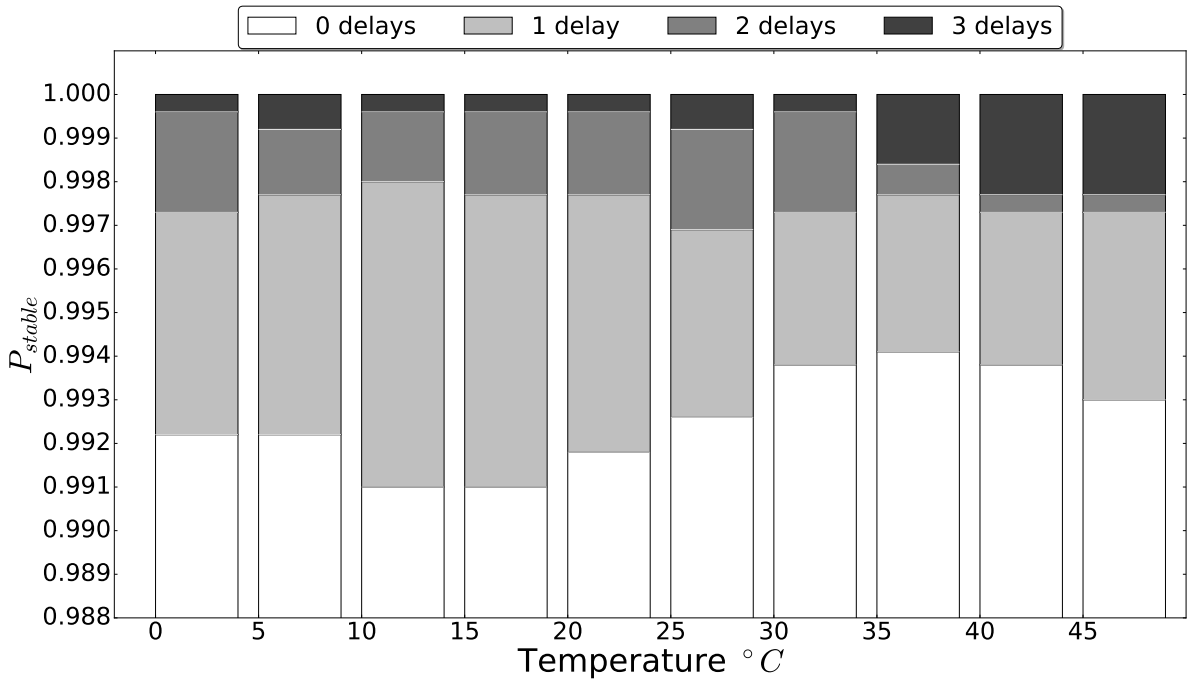


Figure 4.12: Reliability enhancement by challenge filtering algorithm of Fig. 4.5 with different values of k.



(a) Nexys 4 board



(b) ZC706 board

Figure 4.13: Results of temperature test.

The responses to these 70% of retained challenges were further tested under

varying operational conditions. The vendor specified allowable temperature range is [−40°C, +90°C] for Nexys-4 evaluation board and [0°C, +45°C] for ZC706 SoC board. All temperature tests were performed with Thermotron ®8800 temperature chamber [115]. The results are shown in Fig. 4.13. It takes only $k = 4$ for Xilinx Artix-7 FPGA and $k = 3$ for Zynq-7000 SoC to achieve the reliability of 1.0. P_{stable} for the unstable challenges in both cases fall within the range of [0.6, 0.75].

The dependency of P_{stable} on the parameter k is investigated using the Zynq-7000 SoC. The results are depicted in Fig. 4.14. To keep the test time reasonable, only all combinations of challenge bit flips to k contiguous stages are considered. From Fig. 4.14, only 12 out of 127 combinations for $k = 4$ has $P_{stable} \neq 1$. As k increases, the reliability also increases. Based on this experiment, to achieve a perfect reliability of 1 for a 128-bit key, $k = 12$ (∼10% of challenges). If the k stages are randomly chosen instead of adjacent, there is a high probability (above 90%) that $k = 4$ is enough to obtain a set of challenges that produce 100% reliable responses.

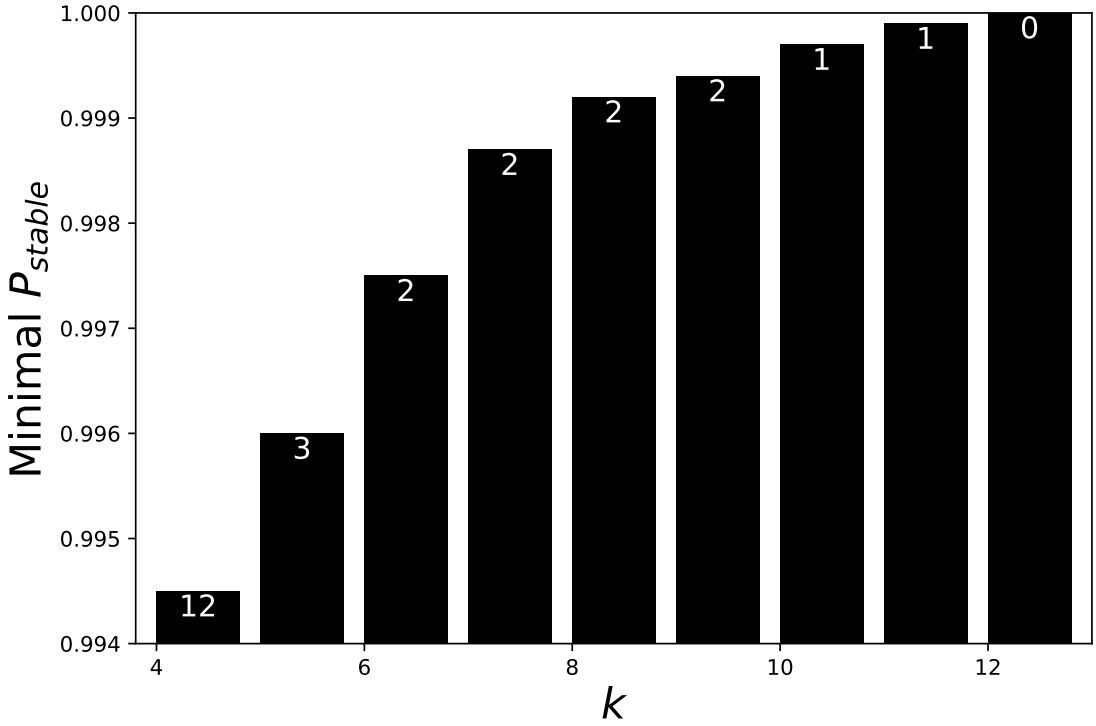
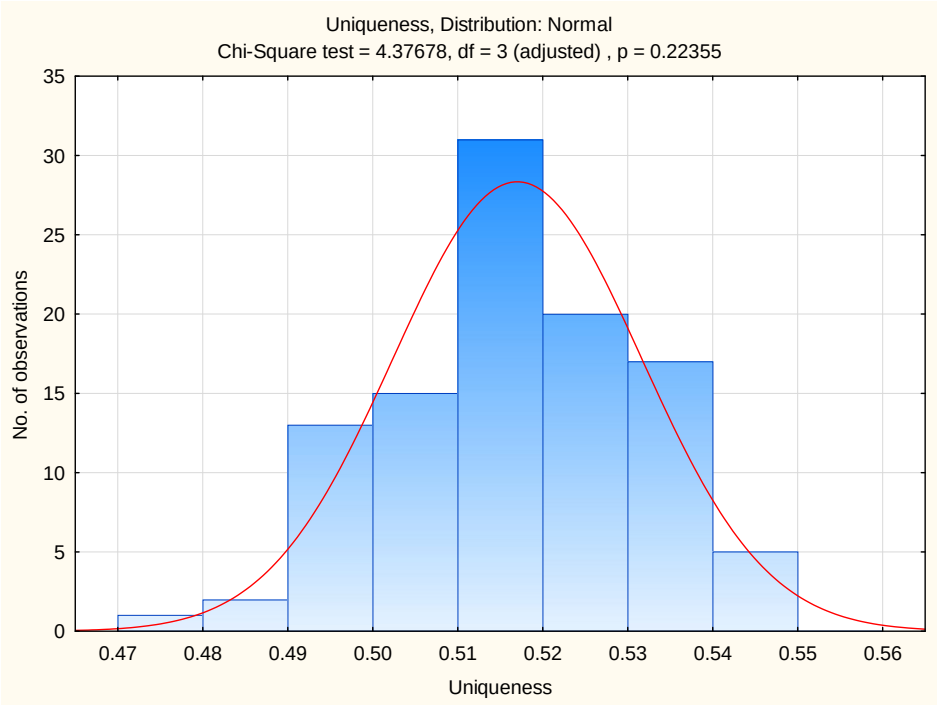


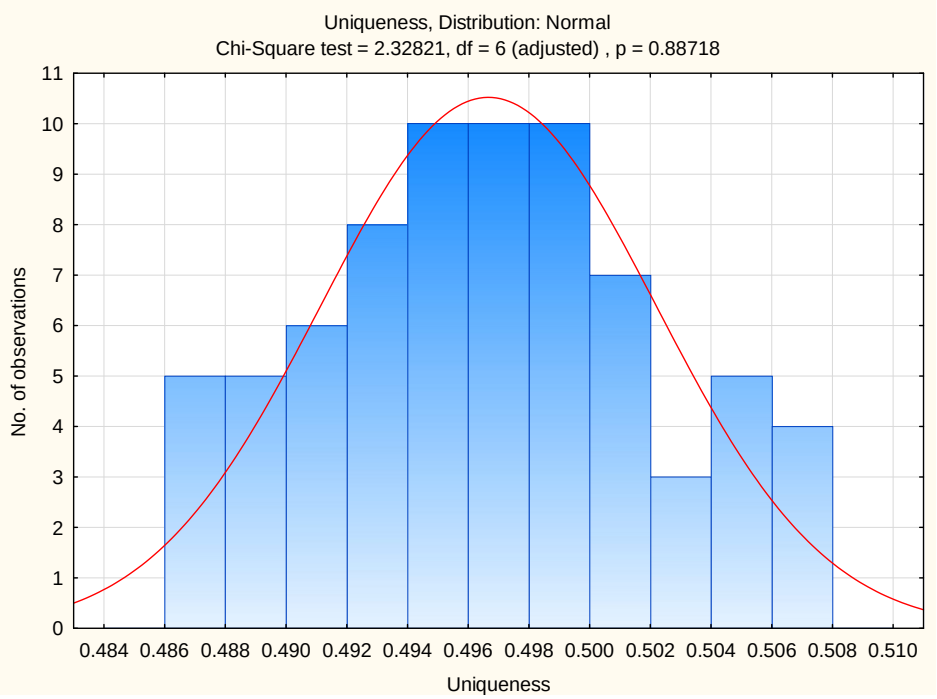
Figure 4.14: Minimum values of P_{stable} for different k.

4.4.4 Uniqueness test

$K = 10{,}000$ challenges have been generated for the uniqueness test.



(a) Between PUF instances across FPGA chips



(b) Between PUF instances within a single chip

Figure 4.15: Uniqueness of reliability-enhanced A-PUF.

All challenges have been filtered by the algorithm in Fig. 4.5. $m = 20$ Xilinx Nexys 4 FPGA chips are used for the implementation of different PUF instances. Each FPGA chip has a set of reliable stable challenges. To obtain the same challenges for each tested instance, a set of 1074 common and reliable challenges for all 20 FPGA chips is extracted. The same experiment for PUF instances across 20 different FPGA chips was also repeated for $D = 8$ A-PUF instances implemented within a single FPGA chip. The results are shown in Fig. 5.18. In both cases, the uniqueness exhibits a normal distribution, with a mean of 0.511 and standard deviation of 0.015 for inter-chip HD and a mean of 0.496 and standard deviation of 0.006 for intra-chip HD. They are both equal to the ideal uniqueness of 0.5 with accuracy up to one decimal place. This implies that the zeros and ones in the responses to a persistently stable challenge are evenly distributed among different PUF instances.

4.4.5 Randomness test

The cryptographic randomness of a TRNG can be validated by the 15 statistical tests provided by the National Institute of Standard and Technology (NIST) statistical test package [88]. A sequence of 10 million bits was generated. These bits are divided into a set of 5.95 million response bits from persistently stable challenges with reliability of 1.0 and a set of 1.65 million response bits from unstable challenges. The remaining 2.4 million response bits to strong challenges were pruned by the algorithm in Fig. 4.5. The final bit sequence is further divided into 100 blocks, each of 59,500 bits for persistently stable challenges and 16,500 bits for unstable challenges, respectively.

The NIST test results in Table 4.2 show that the response bits generated from persistently stable challenges failed some of the tests, which are marked in bold print. On the other hand, the response bits generated from unstable challenges passed all the NIST tests successfully. As expected, the response instability to unstable challenges makes a good entropy source for TRNG. It turns out that randomness and reliability are two conflicting design criteria for challenge space selection, and the parameter k of Fig. 4.5 provides a good tuning knob to trade randomness for reliability.

Nevertheless, the bit sequence produced by A-PUF with enhanced reliability of 1.0 still possesses good entropy except that it is 1-skewed. Because it has 45.9% of zeros and 54.1% of ones, it is not able to pass the frequency based tests (Frequency, Frequency block, Runs, The longest run and Cumulative sums). This 1-skew problem can be mitigated by detecting and recoding the metastability states into stable logic 0 states [110]. Alternatively, the persistently stable response set of A-PUF can be post-processed by some functions like Von Neumann corrector, Linear Feedback

Table 4.2: NIST randomness test results.

Test Description	Passed/Total		P-value	
	stable	unstable	stable	unstable
Frequency (Monobit) Test	0/100	95/100	0.00	0.02
Frequency Test within a Block	20/100	100/100	0.00	0.53
Runs Test	10/100	100/100	0.00	0.74
Test for the Longest Run of Ones in a Block	39/100	100/100	0.00	0.53
Binary Matrix Rank Test	100/100	99/100	0.35	0.33
Discrete Fourier Transform (Spectral) Test	98/100	100/100	0.46	0.21
Non-overlapping Template Matching Test	93/100	98/100	0.51	0.57
Overlapping Template Matching Test	83/100	95/100	0.03	0.35
Maurer's "Universal Statistical" Test	100/100	100/100	0.02	0.03
Linear Complexity Test	91/100	95/100	0.03	0.05
Serial Test	98/100	100/100	0.55	0.73
Approximate Entropy Test	100/100	100/100	0.35	0.62
Cumulative Sums (Cusum) Test	1/100	93/100	0.00	0.03
Random Excursions Test	10/10	10/10	—	—
Random Excursions Variant Test	10/10	10/10	—	—

Shift Register, Multiple Input Signature Register, Adaptive Signature Analyzer, hash functions, etc. as for the raw responses of RO-PUF, SRAM PUF and classical A-PUF before it is used as a true random sequence [95].

4.4.6 Identifier generation

The most important application of A-PUF in FPGA implementation is to generate a reliable and unique identifier (ID) for each FPGA chip. Based on the "persistently strong" challenges, unique and reliable FPGA IDs can be generated by the algorithm proposed in Fig. 4.16.

The uniqueness (dissimilarity) of a set of IDs can be determined by:

$$U_{id} = 1 - \frac{ID_{avg}}{K_{id}} \tag{4.21}$$

where ID_{avg} is the average number of identical IDs and K_{id} is the total number of IDs generated. For example, for an ID with length $L = 2$, there are four possible ID types ("00", "01", "10", "11"). If there are $K_{id} = 57$ IDs (15 of "00", 7 of "01", 10 of "10" and 25 of "11"), then the $ID_{avg} = 57/4 = 14.25$.

Generation of 2 (K_{id}) L-bit ($L = 4$) IDs by the proposed algorithm can be illustrated as follows.

```
1: procedure GENERATE_ID(K_id)
2:     N ← number of stages
3:     LFSR ← Init_LFSR(seed, N)
4:     k ← number of delays
5:     Set_id ← ∅
6:     L ← required ID length
7:     Current_L ← 0
8:     Current_ID ← “”
9:     while |Set_id| < K_id do
10:        challenge ← LFSR.next_challenge()
11:        if Check(challenge) then
12:            Current_ID ← Current_ID + PUF(challenge)
13:            Current_L ← Current_L + 1
14:            if Current_L = L then
15:                Set_id ← Set_id∪ {Current_ID}
16:                Current_L ← 0
17:                Current_ID ← “”
18:            end if
19:        end if
20:    end while
21:    return Set_id
22: end procedure
```

Figure 4.16: Algorithm for unique and reliable ID generation.

1. $K_{id} = 2$ (line 1, parameter of function).
2. $N = 8$ (line 2).
3. $LFSR = Init\_LFSR(seed, N) = Init\_LFSR(\text{“}11111111\text{”}, 8)$ (line 3).
4. $k = 2$ (parameter for function Check, line 4).
5. $Set_{id} = \emptyset$ (line 5).
6. $L = 4$ (line 6).
7. $Current\_L = 0$ (line 7).
8. $Current\_ID =$ “” (line 8).
9. $|Set_{id}| = |\emptyset| = 0 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).
10. $challenge = LFSR.next\_challenge() =$ “01111111” (line 10).

11. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

12. $Current\_ID = Current\_ID$ + PUF($challenge$) = “” + PUF(“01111111”) = “” + ’1’ = “1” (line 12).

13. $Current\_L = Current\_L + 1 = 1$ (line 13).

14. $Current\_L = 1 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

15. $|Set_{id}| = |\emptyset| = 0 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

16. $challenge = LFSR.next\_challenge()$ = “10111111” (line 10).

17. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

18. $Current\_ID = Current\_ID$ + PUF($challenge$) = “1” + PUF(“10111111”) = “1” + ’1’ = “11” (line 12).

19. $Current\_L = Current\_L + 1 = 2$ (line 13).

20. $Current\_L = 2 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

21. $|Set_{id}| = |\emptyset| = 0 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

22. $challenge = LFSR.next\_challenge()$ = “01011111” (line 10).

23. $Check(challenge)$ = False. Then algorithm proceeds to line 9 as if condition is not satisfied (line 11).

24. $|Set_{id}| = |\emptyset| = 0 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

25. $challenge = LFSR.next\_challenge()$ = “10101111” (line 10).

26. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

27. $Current\_ID = Current\_ID$ + PUF($challenge$) = “11” + PUF(“10101111”) = “11” + ’0’ = “110” (line 12).

28. $Current\_L = Current\_L + 1 = 3$ (line 13).

29. $Current\_L = 3 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

30. $|Set_{id}| = |\emptyset| = 0 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

31. $challenge = LFSR.next\_challenge()$ = “01010111” (line 10).

32. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

33. $Current\_ID = Current\_ID$ + PUF($challenge$) = “110” + PUF(“01010111”) = “110” + ’1’ = “1101” (line 12).

34. $Current\_L = Current\_L + 1 = 4$ (line 13).

35. $Current\_L = 4 = L = 4$. Then algorithm proceeds to line 15 as if condition is satisfied (line 14).

36. $Set_{id} = Set_{id} + Current\_ID = \emptyset + \{$“1101”$\} = \{$“1101”$\}$ (line 15).

37. $Current\_L = 0$ (line 16, this variable is cleared as current ID is stored).

38. $Current\_ID$ = “” (line 17, this variable is cleared as current ID is stored).

39. Then algorithm proceeds to line 9 to check loop condition again (line 18).

40. $|Set_{id}| = —\{$“1101”$\}— = 1 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

41. $challenge = LFSR.next\_challenge()$ = “10101011” (line 10).

42. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

43. $Current\_ID = Current\_ID$ + PUF($challenge$) = “” + PUF(“10101011”) = “” + ’0’ = “0” (line 12).

44. $Current\_L = Current\_L + 1 = 1$ (line 13).

45. $Current\_L = 1 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

46. $|Set_{id}| = —\{\text{“1101”}\}— = 1 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

47. $challenge = LFSR.next\_challenge()$ = “11010101” (line 10).

48. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

49. $Current\_ID = Current\_ID$ + PUF($challenge$) = “0” + PUF(“11010101”) = “0” + ’1’ = “01” (line 12).

50. $Current\_L = Current\_L + 1 = 2$ (line 13).

51. $Current\_L = 2 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

52. $|Set_{id}| = —\{\text{“1101”}\}— = 1 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

53. $challenge = LFSR.next\_challenge()$ = “11101010” (line 10).

54. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

55. $Current\_ID = Current\_ID$ + PUF($challenge$) = “01” + PUF(“11101010”) = “01” + ’1’ = “011” (line 12).

56. $Current\_L = Current\_L + 1 = 3$ (line 13).

57. $Current\_L = 3 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

58. $|Set_{id}| = —\{\text{“1101”}\}— = 1 < K_{id} = 2$. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

59. $challenge = LFSR.next\_challenge()$ = “01110101” (line 10).

60. $Check(challenge)$ = False. Then algorithm proceeds to line 9 as if condition is not satisfied (line 11).

61. $|Set_{id}|$ = —{"1101"}— = 1 < K_{id} = 2. Then algorithm proceeds to line 10 as loop condition is satisfied (line 9).

62. $challenge = LFSR.next\_challenge()$ = "00111010" (line 10).

63. $Check(challenge)$ = True. Then algorithm proceeds to line 12 as if condition is satisfied (line 11).

64. $Current\_ID = Current\_ID$ + PUF($challenge$) = "011" + PUF("00111010") = "011" + '0' = "0110" (line 12).

65. $Current\_L = Current\_L + 1 = 4$ (line 13).

66. $Current\_L = 3 \neq L = 4$. Then algorithm proceeds to line 9 as if condition is not satisfied (line 14).

67. $Current\_L = 4 = L = 4$. Then algorithm proceeds to line 15 as if condition is satisfied (line 14).

68. $Set_{id} = Set_{id} + Current\_ID$ = {"1101"} + {"0110"} = {"1101", "0110"} (line 15).

69. $Current\_L = 0$ (line 16, this variable is cleared as current ID is stored).

70. $Current\_ID$ = "" (line 17, this variable is cleared as current ID is stored).

71. Then algorithm proceeds to line 9 to check loop condition again (line 18).

72. $|Set_{id}|$ = —{"1101", "0110"}— = 2 = K_{id} = 2. Then algorithm proceeds to line 21 as loop condition is not satisfied (line 9).

73. As a result algorithm returns set of 2 unique IDs Set_{id} = —{"1101", "0110"}— (line 21).

To check the dissimilarity of the identifiers, $K_{id} = 200$ of each identifier length $L = 1, \ldots, 128$ bit are generated by the proposed algorithm. The reliability of generated IDs is 1.0 with $k = 4$. For $L = 2$, $ID_{avg} = 50$ and $U_{id} = 0.75$. U_{id} increases exponentially with the ID length until $L = 16$. When $L > 16$, all the IDs are unique, as shown in Fig. 4.17. Moreover, the normalized HD among the generated IDs is within the range of [0.479, 0.495]. Therefore, the generated IDs are also unique according to (2.5).

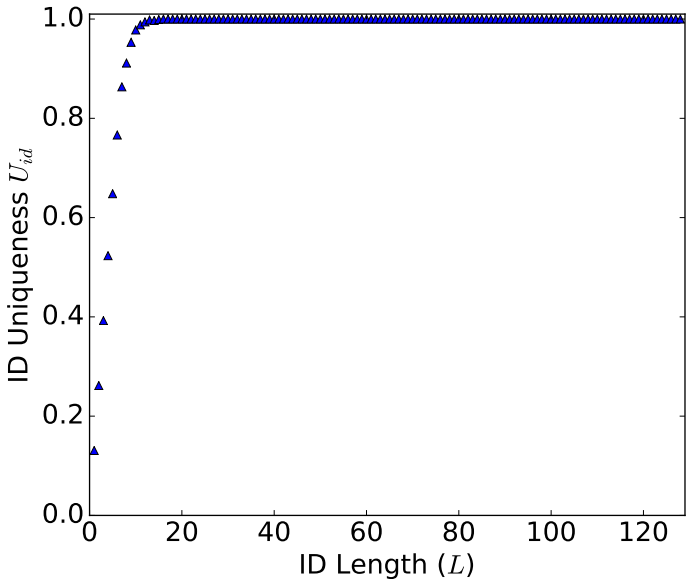


Figure 4.17: Uniqueness of 200 identifiers of each length L generated by Algorithm in Fig. 4.16.

4.5 Summary

This section presents an approach to generate highly reliable CRPs using SR latch based A-PUF in FPGA implementation. The method is based on classifying the challenges into stable and unstable types. Our experiments have shown that around 90% of the challenges can be classified as stable. However, around 20% of the responses are located near the metastable region which are prone to have lower reliability in the face of large environmental variation. To enhance the stability of stable responses, additional changes in intermediate stage delay signs is tested. It has been demonstrated that the combinations of changes in $k = 4$ additional delay signs can sift the persistently stable CRPs to achieve a perfect reliability of 1. The persistently stable CRPs filtered by the proposed algorithm have also been tested for uniqueness and randomness. Ideal uniqueness of 0.5 with accuracy up to one decimal place is also achieved from both inter-chip and intra-chip instances and the randomness is similar to the classical A-PUF. The proposed challenge classification, filtering and unique ID generation algorithms can be applied to generate 128-bit 100% reliable unique FPGA IDs with reasonable processing time.

The cost of the proposed algorithm is the time taken for challenge classification and additional tests for delay sign changes. For $k = 4$, it requires $2^6 = 64$ more challenge application time than for classical A-PUF design. It takes 10 minutes for

Xilinx Artix-7 FPGA to generate K = 40,000 CRPs. 60 unique and reliable IDs of 128-bit each can be created by our method in approximately 10 minutes. Thus, the average time for a single ID generation is around 10 seconds. For Xilinx Zynq-7000 SoC, the time taken is 4 times shorter. The performance can be further boosted by several times with more advanced FPGA chip. Therefore, even without specific design optimization, the ID generation rate is still acceptable for most security protocols.

The classification of stable and unstable challenges as well as the proposed filtering algorithm for persistently stable CRPs do not, however, fully address other inherent limitations and pitfalls of existing A-PUF designs such as imperfect entropy and modeling attack vulnerability. Although the reliability enhanced outputs of the A-PUF do not meet the stringent requirement for true random number sequence, the generated output sequence does possess good entropy except that its distribution is not uniform. This is not a unique problem created by our algorithm but a common limitation of the "raw" outputs generated by any existing PUF designs. The difference is the unstable challenges identified by our algorithm makes a better A-PUF based TRNG.

Notwithstanding the ability to boost the reliability to 1.0, the proposed challenge sifting algorithm makes no change to the underlying structure of A-PUF and its fundamental delay-based mechanism for generating the CRPs, and hence its resistance against modeling attacks. What the proposed A-PUF delay model provided is a better understanding of the cause of low reliability (such as three different type of metastable regions) and the trade-off between reliability and randomness. This formal modeling of the relation between selective bit flips of input challenge and delay sign changes could be useful for maintaining the reliability in devising countermeasure, as presented in our preliminary study [113]. Our future direction is to further leverage on the proposed delay model to temporally obfuscate the challenge or partially reconfigure the architecture without compromising uniqueness and reliability to resist modeling attacks.

Chapter 5

Machine Learning Resistant Authentication Protocol with Enhanced Reliability Arbiter PUF

5.1 Introduction

As the earliest and most well-explored silicon PUF implementations [60], A-PUF is also a popular target of attack. A-PUF makes use of the difference $\Delta(CH)$ between the arrival time of the same test pulse traversing in two symmetrical paths, which is determined by a digital challenge CH, to an arbiter. Thus, the response of an arbiter can be expressed as:

$$R = \begin{cases} 0 & \text{if } \Delta(CH) < 0, \\ 1 & \text{if } \Delta(CH) > 0. \end{cases} \quad . \tag{5.1}$$

Fig. 5.1 shows the distribution of $\Delta(CH)$ for the entire set of challenges of a 16-stage A-PUF. The challenges (CH) are represented and sorted by their decimal equivalent values in the abscissa. The differential delay values ($\Delta(CH)$) are obtained from the simulation results based on the mapping to Xilinx ZC706 FPGA chip. Apparently, the challenges can be separated into two classes (zero and one) by their $\Delta(CH)$ values.

Recent research has shown that attacker can predict the whole CRP set of a 64-stage A-PUF with an accuracy of 98∼99% with the knowledge of three to four hundreds CRPs [106] using linear classifier such as support vector machine (SVM) or logistic regression (LR). Modifications of the A-PUF output by additional XOR gates or feedback loops to mask the raw responses are also vulnerable with slight modifications of the attacking model or more advanced machine learning methods

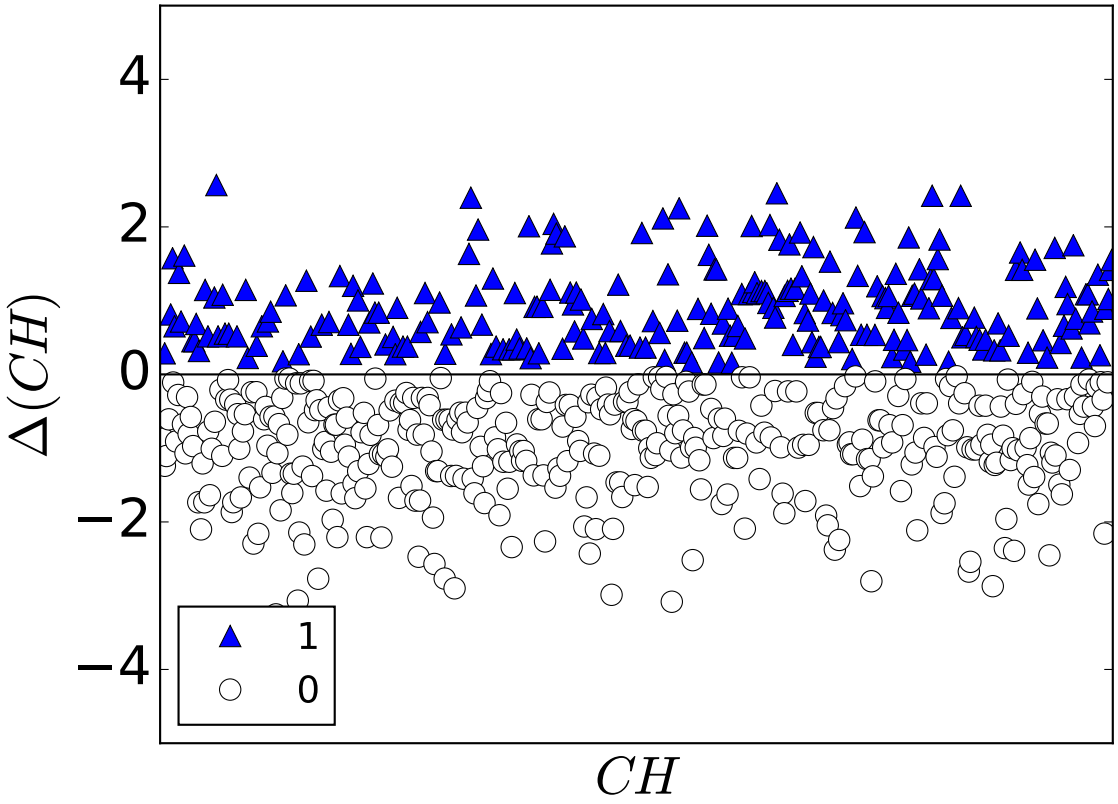


Figure 5.1: Distribution of differential delay values (in ns) for all possible challenges to an A-PUF.

such as gradient boosting or evolutionary algorithms [67]. Post-processing the output response bits is not as effective as tackling the source, i.e., the input challenge bits, directly. More effective countermeasures are usually accomplished by applying some pseudorandom bit obfuscation to the real challenges such that a good quality binary classifier cannot be built. Unfortunately, most of these nonlinear transformations, such as SHA-256 hash function [116], Geffe generator [117], non-linear feedback shift registers [118], alternating step generators [119], etc., require a lot more hardware resources than the implementation of the A-PUF itself.

5.2 Related Works

There are two main ways to address the machine learning vulnerability of a strong PUF. The first approach involves modifications to the internal structure of a PUF to increase the complexity of each basic building block that is replicated to complete

the mapping of an input challenge to an output response. One such example is the Double Arbiter PUF (DA-PUF) [120], which is designed to increase the number of possible symmetric routes. Although this PUF decreases the prediction accuracy of support vector machine (SVM) from 86.3 to 50%, its reliability also degrades from 0.95 to 0.89. Alternatively, the linearity of cascaded blocks can be broken by using non-linear tables to make the transfer function block nonlinear [67]. The lookup table can be designed using SRAM cells, which are similar to the SRAM PUF used to avoid the bias of zeros or ones. It was claimed in [67] that 20K of SRAM cells are sufficient to reduce the predictability of Bagging, SVM, Logistic Regression (LR) and Gradient Boosting (GB) to 70%. The obvious disadvantage of the latter approach is the unacceptably high hardware overhead, which is around 5000 times higher than the original A-PUF design. The non-linearity of the A-PUF circuit can also be increased by feed-forward loops [121]. All the seven modifications of this method have demonstrated an increase in randomness by 10 to 20% with a decrease in uniqueness and reliability by 5 to 10%. Thus, increasing the randomness by partially collapsing the linear addition effectively increases the resistance against machine learning attacks at a lower hardware cost. Owing to the trade-off between reliability and randomness, additional cost, which is dependent on the type of PUF and reliability enhancement technique used, is needed to restore the lost reliability. The transfer function for the CRP mapping of the PUF can be designed based on the inherently non-linear physical property of analog circuits such as the non-linear VTC PUF [122], nonlinear current PUF [123], SCA PUF [124]), etc., but these analog voltage and current functions are difficult to be implemented efficiently on FPGA. As evaluated in [67], these analog transfer function based strong PUFs offer good resistance only against attacks by basic machine learning algorithms such as SVM or LR, but not to advanced attacks using Bagging, Boosting and Evolutionary methods.

The second approach to fortify the PUF against machine learning attacks is to reduce the leakage of challenge-response mapping by controlled strong PUF design [125]. The main idea of this technique is to randomize the challenge and/or response values with a hash function. This scheme was initially applied to weak PUFs to protect their challenge-response interface from being directly accessible by the attacker. Such protection was deemed unnecessary for strong PUFs due to the intractability to exhaustively measure and collect the exponentially huge CRPs, making it relatively safe to leave the CRP interface "as is" for lightweight authentication. As A-PUF has later been successfully modeled with accuracy of 95 to 98% by various machine learning methods with training set size ranges from a few hundred [106] to a few

thousand CRPs [66], controlled implementation of challenge-response interface has been extended to strong PUF. However, secure hash function in hardware implementation consumes much more hardware resources than the PUF itself. The first and probably the most well-known implementation of controlled strong A-PUF was due to XOR-PUF [19]. A recent study [126] has shown that depending on the training set size, the prediction accuracy of A-PUF by LR has been reduced to 60-65% by n-input XOR ($n \geq 11$). This phenomenon can be explained by the exponentially decreasing reliability of A-PUF circuit with the number of inputs n. If $n = 10$, the reliability becomes smaller than 0.1. Albeit resilient against basic A-PUF modeling attack, XOR-PUF has also found to be inadequate against modified mathematical model [66] and more advanced algorithms [67].

Challenge obfuscation can also be used to increase the modeling attack resistance of A-PUF. A pattern based challenge processing algorithm is proposed in [127]. The main idea is to apply part of the challenge to multiple A-PUF instances and choose a response based on a random number. This method is appealing as it does not affect A-PUF reliability. A robust and reverse-engineering resistant approach is made based on the comparison of the substrings of multiple responses [128]. The prover generates a random substring of an original response word and pads it with additional random bits to create a bit string of the required length. Then the verifier matches the transmitted padded string with the response obtained from a PUF model. Since the accuracy of the PUF model is very high (at least 90%), the verifier is able to find the correct substring based on their Hamming distance. This solution was tested to be secure against a number of different attacks, including modeling, replay, seed compromise, etc. However, this protocol can still be attacked by Covariance Matrix Adaptation Evolution Strategy (CMA-ES) with an accuracy of 96.9% using a training set of 4×10^5 CRPs and a total training time of around 30 hours.

In short, controlled strong PUF designs are by far a more practical approach to raise the barrier of modeling attack. It does not directly affect PUF reliability when the obfuscation is applied to the challenge and does not degrade uniqueness and randomness more than other methods, except that they are still attackable by advanced machine learning algorithms like CMA-ES and require non-trivial extra hardware resources. In this paper, we propose a lightweight A-PUF based device authentication method that is practically secure against CMA-ES, Bagging and Boosting techniques.

5.3 Proposed Approach

Three possible implementations of controlled A-PUF designs are proposed, namely SHA-256, T Flip-Flop (TFF) register and Multiple Input Signature Register (MISR) challenge value obfuscation. Their resistance against modeling attacks and overhead are analyzed and compared in this section.

5.3.1 SHA-256 challenge value obfuscation

The most simple and reliable approach to randomize the mapping between the challenge and response is by cryptographic hash function. This technique is inspired by similar approaches used to protect the CRPs of weak PUFs or build a TRNG based on PUF (e.g. [93]). The block diagram of possible implementation is depicted in Fig. 5.2.

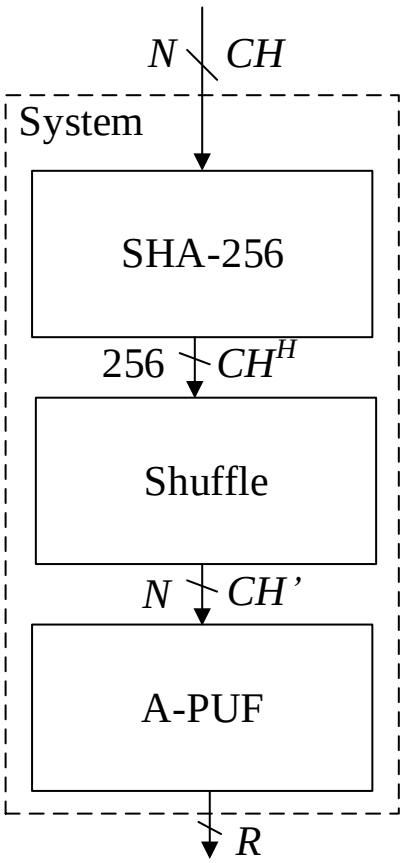


Figure 5.2: SHA-256 challenge value obfuscation scheme for A-PUF.

The system's input is an N-bit challenge, which is fed to a SHA-256 module and transformed to a 256-bit hash value (CH^H). Since the output of SHA-256 has a fixed length of 256 bits, which may be larger than the N-bit input of the A-PUF module. The hashed output is reduced to the size of N and shuffled. This can be done by a decoder or similar circuit. This processing method has been implemented in Xilinx Artix-7 FPGA ($N = 128$ for A-PUF) and tested on a simple machine learning attack using Support Vector Machine (SVM) algorithm. The results of this attack are shown in Fig. 5.3.

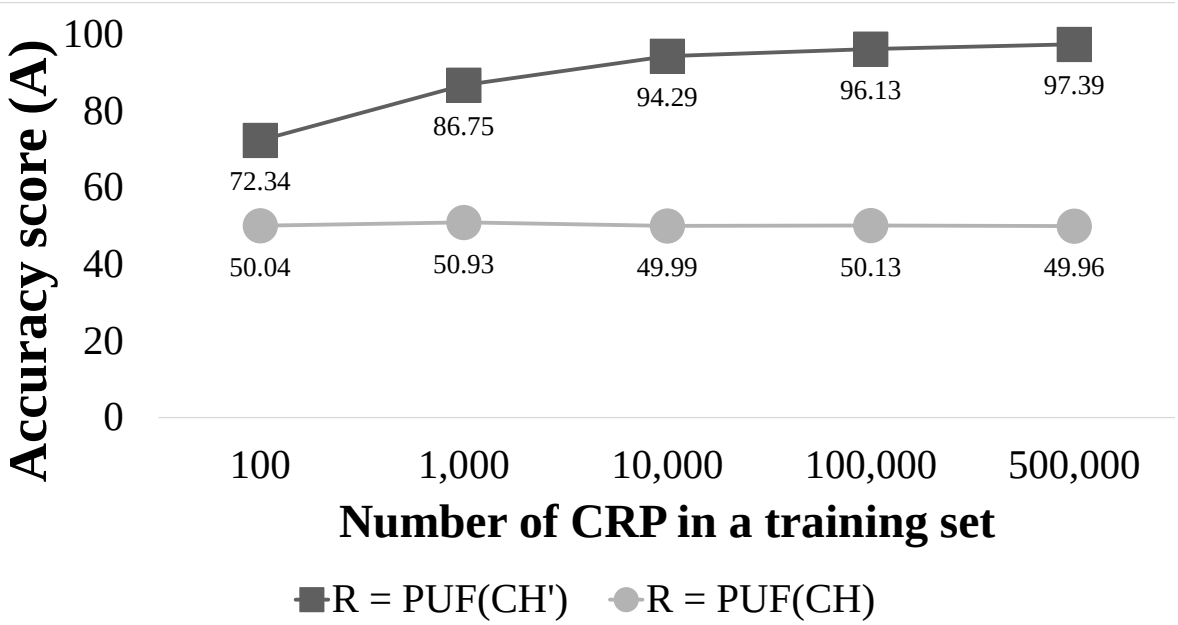


Figure 5.3: SVM based machine learning attack on SHA-256 pre-processing.

The reported accuracy scores on external CRPs (CH, R) are fluctuating around 50%. This probability is as good as a random guess for a binary response. On the other hand, if the attacker has access to the true CRP values (CH', R), the SVM is able to achieve a high accuracy of 97.39% on a large set of CRPs. Despite good resistance against SVM attack, the obvious disadvantage of this technique is its significant hardware overhead, which will be further analyzed in Subsection 5.3.4.

5.3.2 TFF register based transformation scheme

A register (C-REG) containing N T flip-flops (TFF) is proposed for processing the externally input challenge P (see Fig. 5.4). According to (4.9), the differential delay $\Delta_{A_N B_N}(f(P))$ will be modified to:

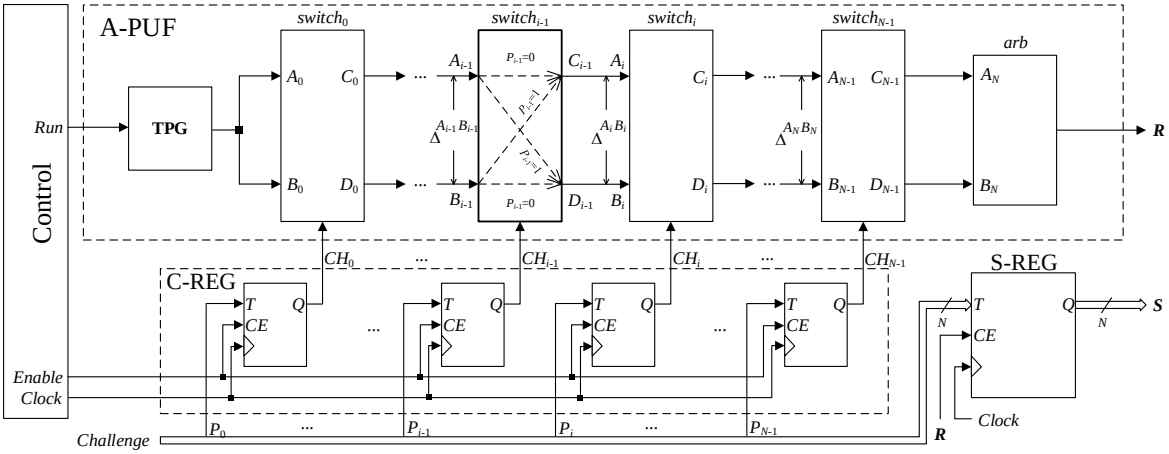


Figure 5.4: A-PUF design modified by adding TFF register.

$$\begin{aligned}\Delta_{A_N B_N}(f(P)) &= \sum_{j=0}^{N-1}[\delta_j \prod_{i=j}^{N-1}[1-2\cdot f(P_i)]] = \\ &= \sum_{j=0}^{N-1}[\delta_j \prod_{i=j}^{N-1}[1-2\cdot (P_i \oplus CH_i)]] = \\ &= \Delta_{A_N B_N}(P \oplus CH).\end{aligned} \tag{5.2}$$

where $f(\cdot)$ is the obfuscation function and CH is the current state of the C-REG.

As a result, besides the individual switch characteristics ($\delta_i \forall i \in [0, N-1]$) and the challenge P, the A-PUF delay model is also dependent on the current state CH of the C-REG, which is in turn dependent on its initial state (seed value). The obfuscation makes $\Delta_{A_N B_N}$ a non-linear function of the external input challenges that can be collected by the attackers, and a different classifier has to be built for each seed, which dramatically increases the search space to mount an attack with the already obtained CRPs.

Consider a two-party authentication protocol for a trusted party and a PUF device. During the enrolment phase, the trusted party collects a sufficiently large number of responses. First, the trusted server classifies the challenges and filters out the weak challenges. Since the actual challenges can be computed by XORing P and CH, the server will apply a sequence of challenges P to a PUF to achieve 100% reliability before storing the actual CRPs of the PUF in a database.

Upon deployment of the device, each authentication session is performed by the device sending an authentication request to the server of the trusted party. This

request should include the signature S (see S-REG block in Fig. 5.4) to identify the device [129]. Assuming that K challenges have been sent during the previous authentication sessions, the signature can be expressed as:

$$S = \oplus_{i=0}^{K} P^{\Omega_i} \ \forall i \ s.t. \ R = PUF(P^{\Omega_i}) = 1. \tag{5.3}$$

In other words, S is a Boolean ring sum of all K challenges that produce a logic one response. This checksum validates that the device used only CRPs sent by the server to avoid abusing the A-PUF.

In each authentication session, the server sends only persistently stable challenges to the PUF device. Since the trusted party knows the current state CH of the device and the next challenge from its database D_{CH}, the challenge to be sent can be computed by $P = CH \oplus D_{CH}$. As the same set of challenges applied to the C-REG in different order can produce different responses, the server can respond by sending an order set of challenges to the device. By computing the obfuscated challenges, the server can verify the responses received from the device against the correct responses stored in its database to confirm the authenticity of the device. Even if the attacker can access the observed CRPs (P, R) from previous authentication sessions, it will not give him an advantage in the prediction of future responses to any input challenges.

To test the resistance of the proposed challenge obfuscation method, $n_{pstrong} = 10^6$ persistently stable challenges have been selected out of $n_c \approx 5{\times}10^6$ challenges. Then, all of the selected challenges are obfuscated by the proposed method (see Fig. 5.4) with an initial seed value of all zeros.

The original $\text{CRP}_{pstable}$ set is divided into training and testing sets as before. The size of the training set varies from 100 CRPs to 500,000 CRPs and the size of the corresponding testing set varies from 999,900 CRPs to 500,000 CRPs. As shown in Fig. 5.5, the prediction accuracy increases from 70.49% for the smallest training set of 100 CRPs to 98.52% for the largest training set of 500,000 CRPs.

On the other hand, if the SVM is trained by the CRP set obfuscated by our proposed technique, their prediction accuracy fluctuates around 50% and is independent of the training set size. This prediction accuracy is as good as random prediction based binary classifier. This proves that the challenge obfuscation by C-REG makes modeling attacks notoriously more difficult to perform, because it reshuffles the challenge bits to increase the nonlinearity of the A-PUF model. Furthermore, the current challenge becomes dependent on the historical input and obfuscated challenges, which reduces the correlation between cascaded stage delays and greatly increases the complexity of the learning model.

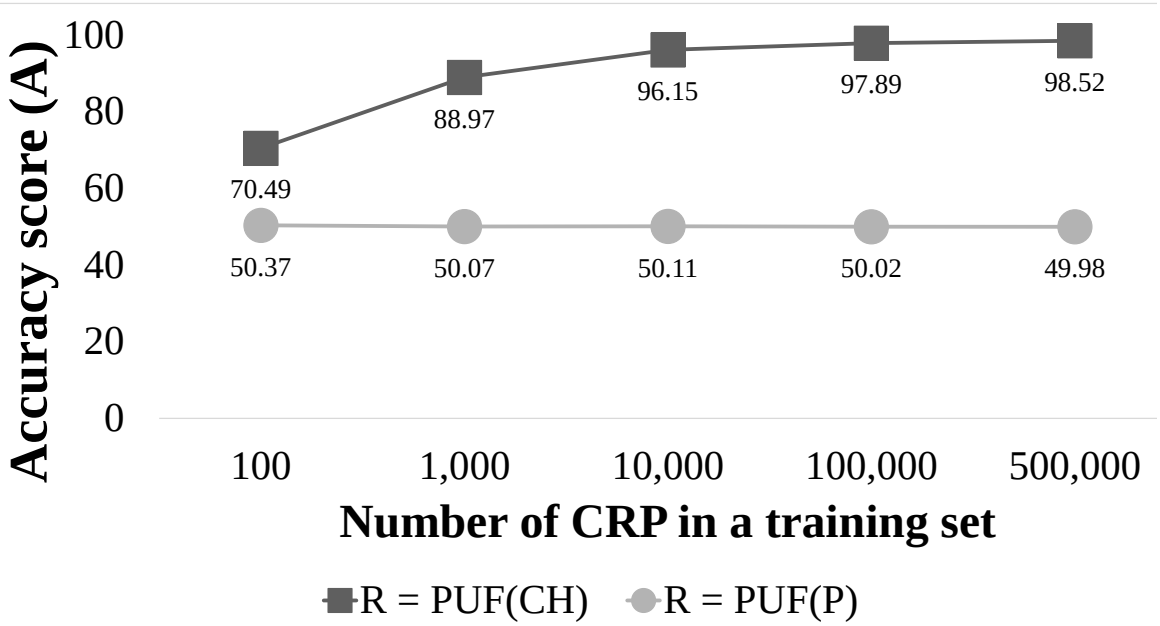


Figure 5.5: Prediction accuracy of SVM trained using data sets generated by original and obfuscated with TFF register challenges.

The proposed countermeasure is resilient against typical machine learning attacks that use similar strategies as SVM and Logistic Regression. It has not been attested against more sophisticated emerging machine learning attacks that use very different strategies like evolutionary strategy [68]. Although it may not be completely shielded from more advanced latest attacks, it increases the attack barrier with smaller hardware overhead compared with the conventional countermeasures. This approach is dedicated to A-PUF design, because it relies on the use of persistently stable challenges, which are not applicable for other strong PUFs. Generic approaches such as error correction codes require much more hardware resources to protect A-PUF against typical machine learning attacks. Currently, it is assumed that a trusted party can help to collect the CRPs. If practical or cost constraint forbids CRP measurements for a large number of devices, this requirement can be relaxed by building the A-PUF model based on the original CRP set using much less number of challenges.

5.3.3 MISR circuit for challenge pre-processing

In this proposal, a MISR circuit is used for the pre-processing of challenge before it is applied to the A-PUF. A MISR requires less hardware resources to perform the non-linear transformation than other challenge obfuscation techniques. Since MISR is an indispensable block for the computation of compact signature for multichannel

input test data when design-for-testability is incorporated in an ASIC or a FPGA design where the A-PUF is embedded, the MISR used for test compaction can be reused with no hardware overhead.

A MISR can be represented by an initial value (*seed*) in memory, a GF(2) feedback polynomial $\phi(X) = \oplus_{i=0}^{N} \alpha_i x^i$, where $\alpha_i \in \{0, 1\}$ are the polynomial coefficients, N is the degree of the characteristic polynomial $\phi(X)$ and X is an N-bit input vector. The coefficients α_i can be fixed at the design phase. For test purpose, these values can be changed by reconfiguration to generate different signatures. From security point of view, changing α_i corresponds to applying a different non-linear transformation to the same challenge, which can greatly enhance its resistance against modeling.

MISR converts an input challenge (CH) to a new challenge (CH') based on its current state. The actual challenge applied to the A-PUF $CH'(k) = f(seed, CH(k-1), CH'(k-1), \phi(X))$ at an arbitrary instant k is dependent on the non-linear mixing of all previously input and converted challenges. The actual challenge $CH'(k)$ applied to the PUF instance can be obtained by the following equation.

$$\begin{bmatrix} CH'_0(k) \\ CH'_1(k) \\ CH'_2(k) \\ \vdots \\ CH'_{N-1}(k) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & \alpha_0 \\ 0 & 1 & 0 & \dots & \alpha_1 \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \dots & 1 & \alpha_{N-1} \end{bmatrix} \times \\ \times \begin{bmatrix} CH'_0(k-1) \\ CH'_1(k-1) \\ CH'_2(k-1) \\ \vdots \\ CH'_{N-1}(k-1) \end{bmatrix} \oplus \begin{bmatrix} CH_0(k-1) \\ CH_1(k-1) \\ CH_2(k-1) \\ \vdots \\ CH_{N-1}(k-1) \end{bmatrix} \tag{5.4}$$

where $CH_i(k-1)$ is the i-th bit of the externally applied input challenge currently used for the authentication and $CH'_i(k-1)$ is the i-th bit of the current MISR state. The actual challenge applied to the A-PUF $CH'_i(k)$ will become the next MISR state.

As an example, consider an 8-stage A-PUF design augmented by a MISR with a feedback polynomial $\phi(X) = x^8 \oplus x^6 \oplus x^5 \oplus x^4 \oplus 1$. If the present MISR state $CH'(k-1)$ is 11001010 when the verifier applies a challenge word $CH(k-1) =$ 10010011 externally to the A-PUF through the MISR, then the actual challenge $CH'(k)$ of the A-PUF will be 01110110, which is very different from the user applied challenge $CH(k-1)$. The hamming distance between these two challenges is 5.

The architecture of the proposed A-PUF is shown in Fig. 5.6. An ordinary BILBO can operate in four different register modes. It is configured as a memory in normal

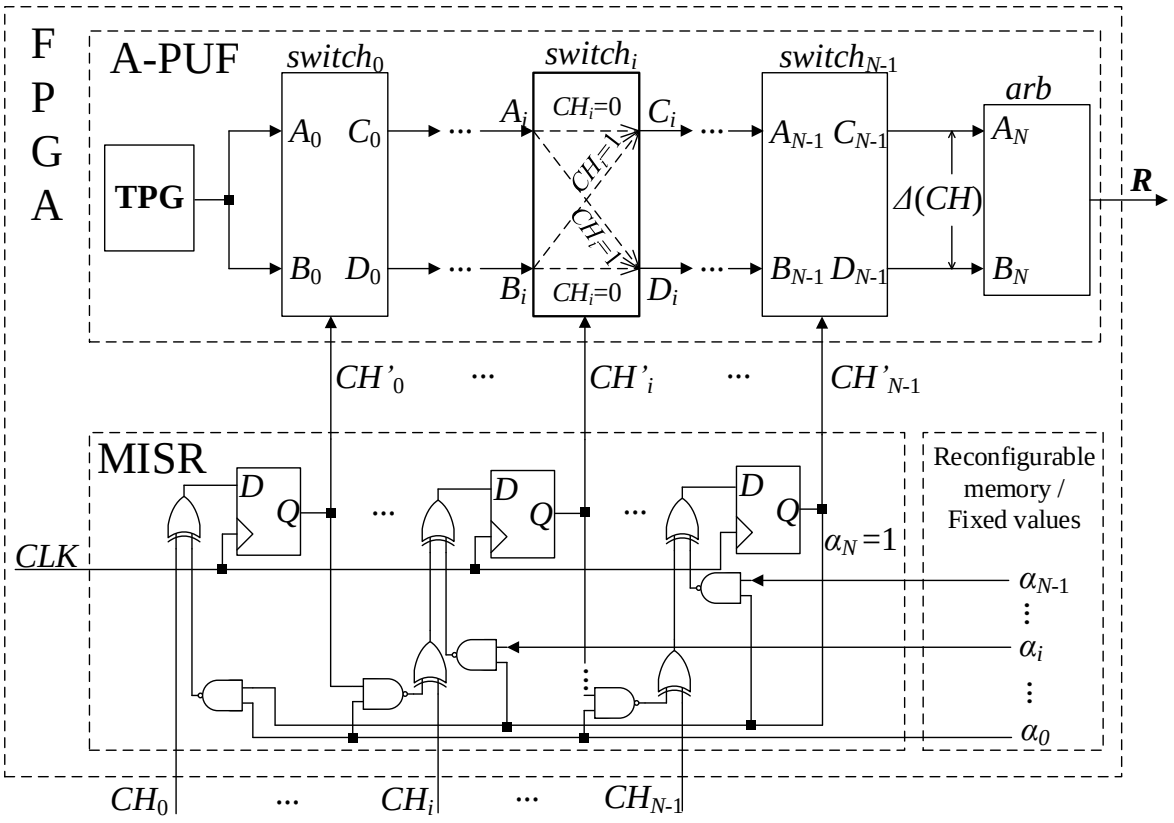


Figure 5.6: Proposed MISR fortified A-PUF architecture.

mode, as a shift register in scan mode and as a MISR in test generation or signature analysis mode, and its content can be cleared in reset mode. Only two of the four modes of BILBO are needed in this case. The initial state (*seed*) is applied to the A-PUF in normal mode ($\alpha_0 = 0$) and all subsequent challenges are applied in MISR mode ($\alpha_0 = 1$).

The results of machine learning attack using SVM algorithm are shown in Fig. 5.7. If machine learning model is trained by CRP′ data set, the accuracy score is high with scores of [81.92, 98.34]%.

On the other hand, if these algorithms are trained by CRP set, their prediction accuracy fluctuates around 50% and is independent of the path length of A-PUF. The scores correspond to a random prediction for binary classification model. Therefore, the use of challenge transformation by MISR achieves the same effect of reshuffling the challenge bits to increase the nonlinearity of A-PUF model, making modeling attacks also notoriously more difficult to perform. Furthermore, the current challenge becomes dependent on the historical input and obfuscated challenges, which

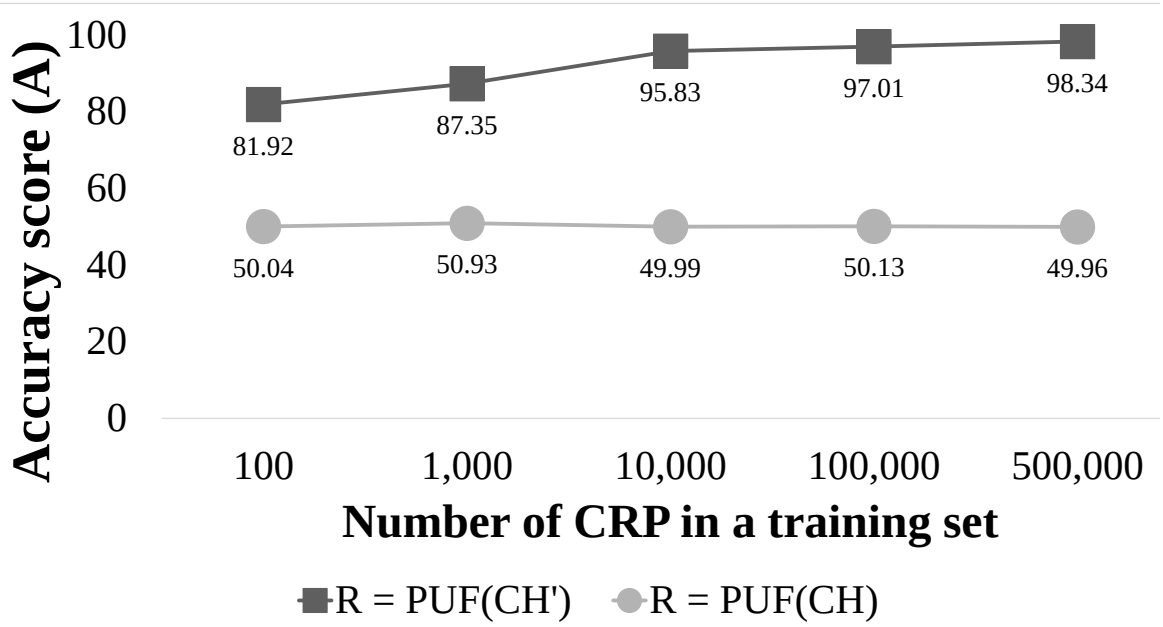


Figure 5.7: Prediction accuracy of SVM trained using data sets generated by original and obfuscated with MISR register challenges.

eliminates the correlation between cascaded stage delays and greatly increases the complexity of learning model.

5.3.4 Comparison

Converting A-PUF design to a controlled PUF has shown increase in resistance to modeling attacks by simple machine learning methods such as SVM, Logistic Regression, shallow artificial neural networks, etc. The accuracy of the model drops to around 50%, which is unacceptable for binary classifier. However, these methods can still be successfully attacked by using advanced techniques as Evolutionary Strategy (CMA-ES). Furthermore, their implementations incur hardware overheads which may not be acceptable when weak PUFs or cryptographic algorithms can be used instead to provide the same level of security. The results of the comparison are shown in Table 5.1.

SHA-256 can be a very secure solution against modeling attacks, but it requires 10 times more resources than the other two methods. On the other hand, MISR based obfuscation provides much better level of security with reasonable overhead than the insecure TFF Register, which has only limited improvement against modeling attacks. Therefore, MISR obfuscation is chosen to build an authentication protocol for A-PUF.

Table 5.1: Comparison of proposed challenge obfuscation methods

Method	**# LUT blocks**	**# of Flip-Flops**	**SVM attack, A**	**CMA-ES attack, A**
SHA-256	3756	2609	50%	50%
TFF Register ($N = 128$)	0	128	50%	95%
MISR ($N = 128$)	380	128	50%	60%

5.4 Precise A-PUF Machine Learning Model

While machine learning can be used for attacking a PUF, it can also be used advantageously to build a precise model to build secure authentication protocol with a highly reliable strong PUF so that no raw CRP needs to be transmitted in clear or accessible externally. The reliability enhancement algorithm of A-PUF is based on the approach described in Chapter 4. In the experimental results shown in Fig. 5.8, six major classes of quadruple responses were generated from the 24-bit A-PUF implemented in Xilinx Artix-7 FPGA. This set contains 2^{22}, which covers the complete CRP set of 24-bit A-PUF ($4 \times 2^{22} = 2^{24}$) quadruple responses. The distribution was obtained by repeating the experiment for $E = 100$ times. All classes shown in Fig. 5.8 are stable except the category labelled "others", which contains following quadruples, {1, 1, 1, 0}, {1, 1, 0, 1}, {1, 0, 1, 1}, {0, 1, 1, 1}, {1, 1, 1, X}, {1, 1, X, 1}, {1, X, 1, 1}, and {X, 1, 1, 1}. Therefore, the complete CRP set can be merged into three bigger classes "stable zero" ({1, 1, 0, 0} and {1, 1, X, X}), "stable one" ({0, 0, 1, 1} and {X, X, 1, 1}) and "metastability state" ({1, 1, 1, 1} and "others"). Every quadruple from the "others" category can be corrected into {1, 1, 1, 1}. These results agree with the experiments conducted on 128-stage A-PUF implemented in the same FPGA chip as the response distribution is the same in terms of the percentage of complete CPR set (of 10^{10} quadruples) generated.

From Fig. 5.8, the quadruple responses of an A-PUF can be divided into six different classes. In contrast to the existing models with accuracy of 95-98%, a more precise ($\simeq$100% accuracy) model can be built based on the stable quadruple responses defined in Chapter 4 and experimentally validated over allowable range of operating voltage and temperature variations specified by the FPGA device or board used for the A-PUF implementation. An accurate A-PUF model is built by using a tripartite classification algorithm to segregate the CRPs into five different classes, namely unstable, metastable zero, metastable one, stable zero and stable one. As illustrate

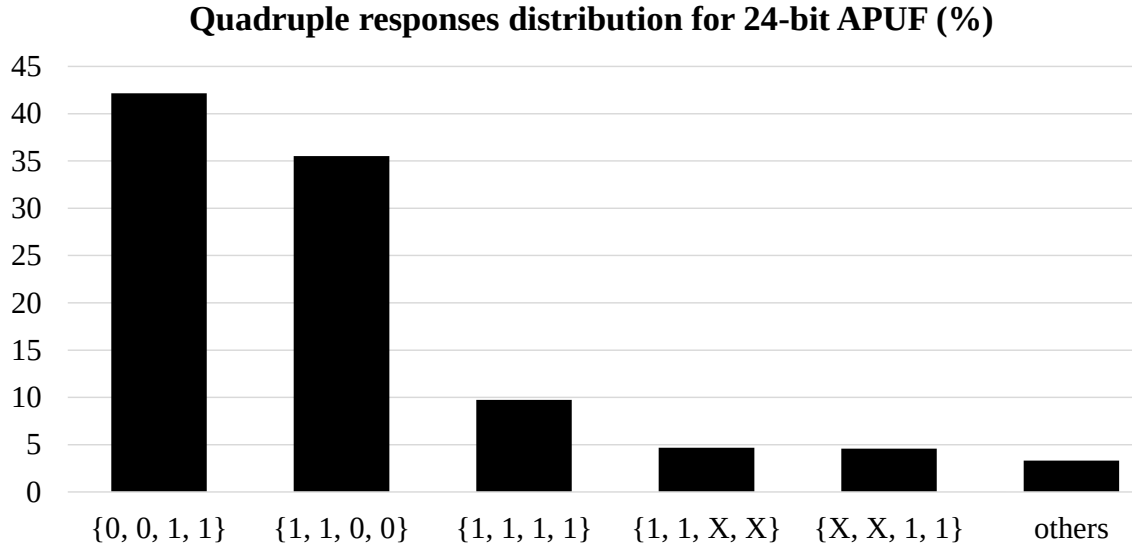


Figure 5.8: Quadruple response distribution for 24-bit A-PUF.

in Fig. 5.9, the classes of response before processing in each stage are enclosed in rectangular blocks and the final classified responses are enclosed in circular blocks.

The first stage of classification is the most complicated one as the classifier has to differentiate between stable and metastable quadruple responses from the unstable quadruple responses. Therefore, a fully connected Deep Neural Networks (DNN) is used for the first classifier. The proposed DNN architecture is shown in Fig. 5.10. The input-layer contains an N-bit parity vector, which determines the sign of each delay element. These inputs are further propagated using 20 layers (L_1, ..., L_{20}) of 2048 Rectified Linear Units (ReLU) followed by a softmax layer for binary classification, 0 for unstable quadruple response ({1, 1, 1, 1} and responses from the class "others") and 1 for stable and metastable quadruple responses ({0, 0, 1, 1}, {1, 1, 0, 0}, {X, X, 1, 1} and {1, 1, X, X})). There is no necessity to further segregate the responses {1, 1, 1, 1} from those under "others" as these unreliable responses can be dumped together with {1, 1, 1, 1} as one unreliable class to separate it from the other four response classes without loss of accuracy.

The DNN is trained to model two different path lengths of A-PUF, $N = 24$ for short A-PUF and 128 for standard A-PUF. The quadruple CRP set generated is divided into three datasets, namely training (80% of the data), validation (or development – 10%) and test (10%). Each quadruple challenge is repeated 100 times on the physical PUF device to determine its response stability before feeding it into the DNN for training. If the response quadruple has the same value each time, it is labeled as stable and belongs to one of the followings, {0, 0, 1, 1}, {1, 1, 0, 0}, {X,

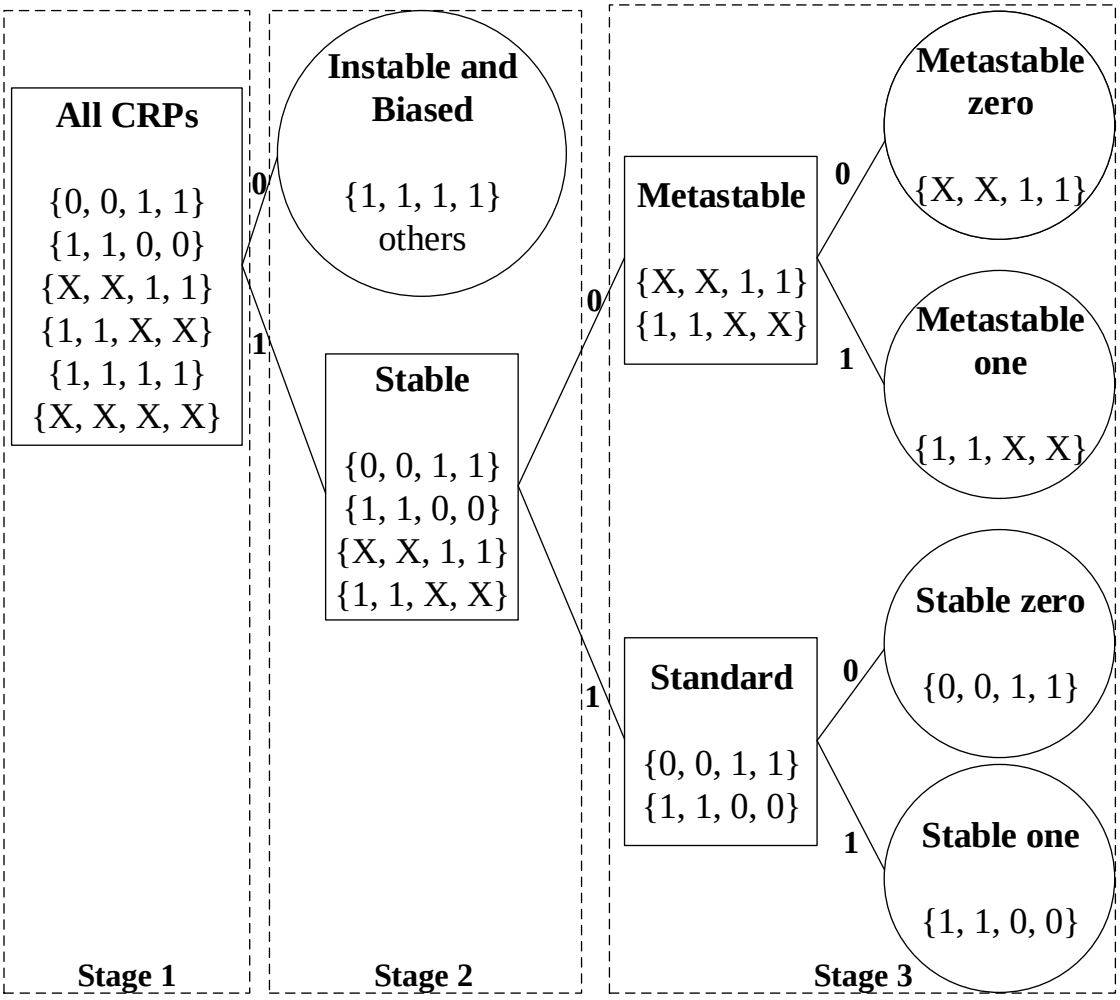


Figure 5.9: Response classification algorithm diagram.

X, 1, 1} or {1, 1, X, X}. Otherwise, it is labeled as "others" in the training set. The trained models are tested with temperature and voltage variations to determine its stability. To avoid overfitting, two techniques have been applied, L2-regularization and dropout ($p = 0.5$). The CRP sets containing 2^{22} and 10^{10} quadruples have been generated to build the models for $N = 24$ and 128, respectively. An accuracy score of 100% has been achieved consistently on training, validation and test datasets for both 24-stage and 128-stage A-PUF implementations.

The classifier in the second stage does not require to be as powerful because separating the stable quadruple responses from the metastable quadruple responses is a much simpler task. Therefore, the neural network in the previous stage is converted to a shallow version with the number of layers reduced to three including the softmax. This simplified classifier is reused and tested to be capable of providing 100% accuracy

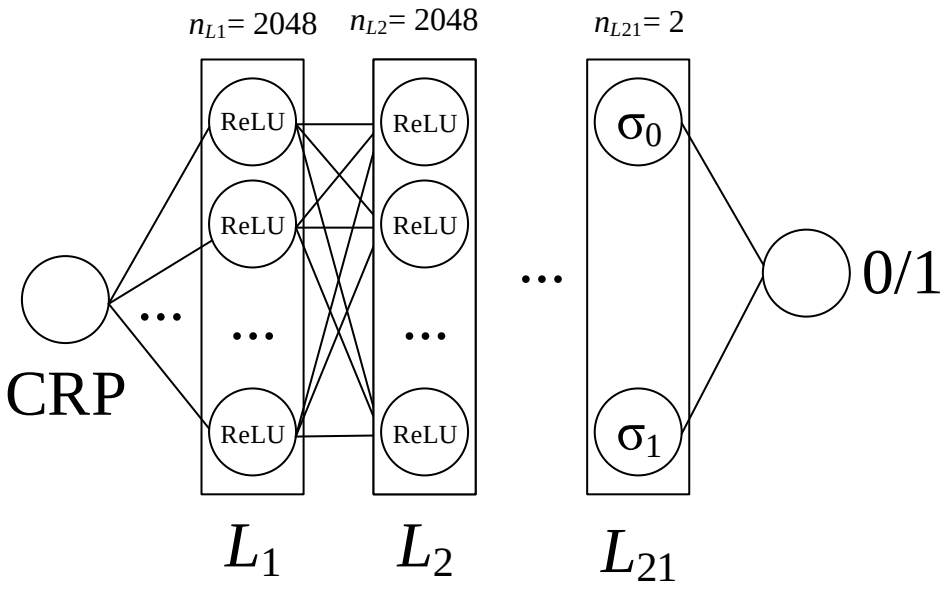


Figure 5.10: Deep neural network architecture for the classifier in stage 1.

in classifying the quadruples as stable ({0, 0, 1, 1} and {1, 1, 0, 0}) and metastable ({X, X, 1, 1} and {1, 1, X, X}).

The last stage utilizes the same model with different parameters to separate the one from zero for both stable and metastable classes of quadruples (see the classifiers "Metastable" and "Stable" in Fig. 5.9). The classifier in this level can be implemented by any linear classification algorithm such as Logistic Regression or SVM. Since majority of the quadruples are stable (86.94%), it is reasonable to use a simple model for this classification.

The constructed precise A-PUF model is stored at the authentication server. Since all responses can be reliably generated internally by the server A-PUF model and the physical device, no helper data or "raw" CRP needs to be stored or transmitted in clear or accessible externally. Also, no quadruple CRP needs to be excluded or forfeited from being used by the protocol as every one of them can be reproduced on both the device and the server sides.

5.5 Authentication Protocol

The proposed protocol involves exchange of messages among a trusted party, an authentication server and an A-PUF embedded in the device to be authenticated.

5.5.1 Auhtenticable device

The proposed PUF is implemented based on an N-stage A-PUF block and a Built-In Logic Observer (BILBO) [130]. Its block diagram is shown in Fig. 5.12. The BILBO block is used primarily in Built-In Self-Test (BIST) applications. It can operate in four modes, which are determined by the value of M. When M = "00", it is used as a shift register in scan test; when M = "01", it is used as a Linear Feedback Shift Register (LFSR) for test pattern generation; when M = "10", it is used as a memory register; when M = "11", it is used as a Multiple Input Signature Register (MISR) for compact test signature generation. These functions can be used advantageously to aid the segregation and obfuscation of CRPs. BILBO in memory mode is used to feed the input challenges directly to the A-PUF for building the A-PUF model. MISR mode is used to obfuscate the true challenge, hence increases the complexity of machine learning attack. The functions of all components are summarized in Table 5.2.

Table 5.2: Components of authenticable device.

Component	Description
LFSR	Generates $(N-2)$-bit challenge CH with low correlated M-sequence.
BILBO	As memory register (M = "10") during enrollment to feed C from LFSR directly to A-PUF module. As MISR in authentication mode to obfuscate (lossy data encoding) CH into $(N-2)$-bit CH'.
A-PUF	Arbiter PUF with SR-latch based arbiter, which takes N bits of MSB + CH' + LSB as input and outputs a 2-bit response R for three states, "00" (logical zero), "10" (logical one), "11" (metastable state).
SREG	Collect four responses from A-PUF to form an 8-bit quadruple response R_Q.
ECC	Stabilize quadruple response R_Q to R_{Corr} by majority voting.
Encoder	Compress R_{Corr} by extracting the first four bits into R_{Enc} as the remaining bits of R_{Corr} can be decoded from R_{Enc}.
FIFO	Store R_{Enc} $\frac{K}{4}$ times to make up R_{Final}.
FSM	Control LFSR, BILBO, SREG, FIFO and Encoder blocks, and padding challenge CH' with MSB and LSB.

The $N-2$ bits of CH (which are the effective middle bits of the quadruple challenge) input into the BILBO block are generated by a separate LFSR block. This $(N-2)$-bit word is fed into the BILBO circuit when M = "11" is sent from the Finite State Machine (FSM) to the BILBO block to set it into MISR mode. Each $(N-2)$-bit obfuscated challenge CH' generated by the BILBO is padded with four

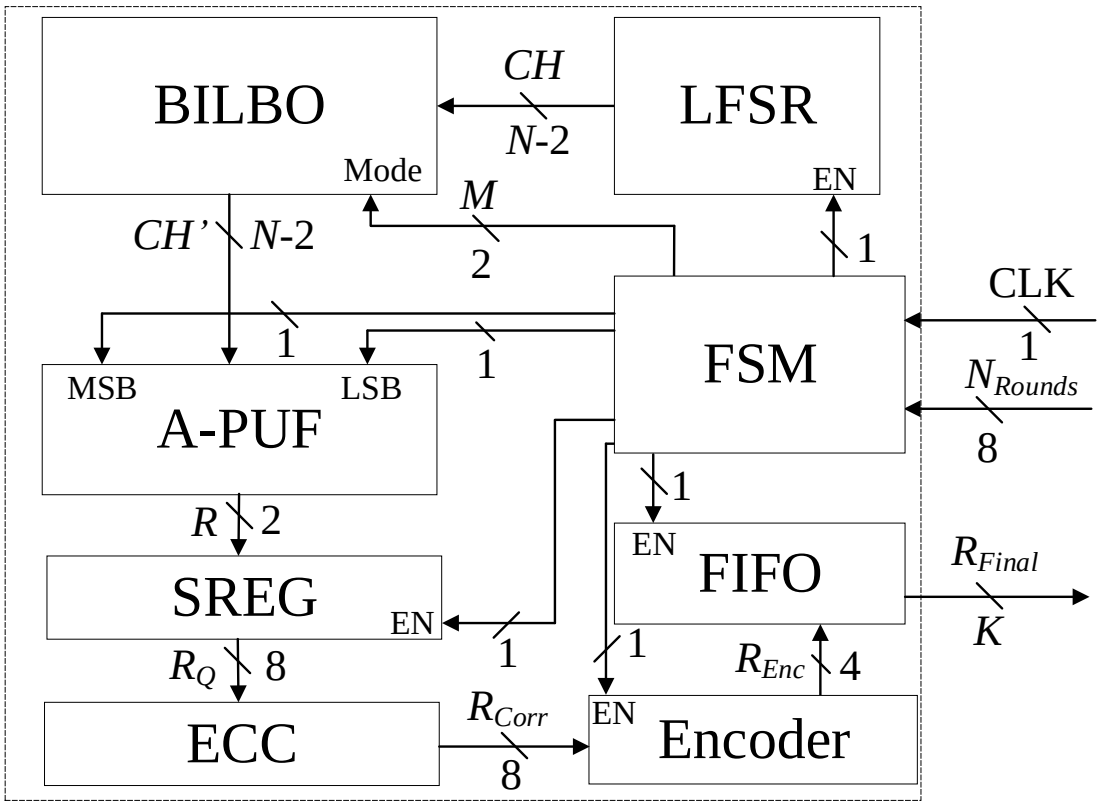


Figure 5.11: Block diagram of authenticable device.

possible combinations of MSB and LSB provided by the FSM to form a quadruple challenge to the A-PUF block. The A-PUF outputs a two-bit binary word R to encode the three trits ({0, 1, X}) as shown in Fig. **??**. The quadruple response due to CH' is stored in an 8-bit shift-register SREG in Fig. 5.12. Although the reliability of the A-PUF has been significantly enhanced by considering quadruple response, there is still a very small probability of bit error. Since every CRP has been repeated 100 times to build an A-PUF model at the server side, it has been experimentally verified that repeating every quadruple challenge $k = 5$ times at the device side to vote for the trits of the quadruple response is sufficient to achieve a reliability of 1.0 over the specified operating temperature and voltage range. The $k = 5$ collected quadruple responses R_Q to the same CH' are fed to the ECC block that implements the majority voting. If the result of majority voting of R_Q is equal to {0, 0, 1, 1}, {1, 1, 0, 0}, {X, X, 1, 1}, {1, 1, X, X} or {1, 1, 1, 1}, the output R_{Corr} of ECC will be one of these quadruples according to the majority votes. Otherwise, R_{Corr} will be corrected to {1, 1, 1, 1} which falls into the unstable quadruple response class. The

five unique quadruples of R_{Corr} is symmetrical, i.e., the second and third trits are the same as the first and fourth trits. This undesirable redundancy is removed by an Encoder block to compress R_{Corr} to only four binary bits that uniquely encode the two non-redundant trits for each quadruple response class. The quadbit R_{Enc} to each CH' is stored in a first-in-first-out (FIFO) buffer. The final K-bit response R_{Final} is produced by making the BILBO performing N_{Rounds} of iterations $\frac{K}{4}$ times to generate $\frac{K}{4}$ different obfuscated quadruple challenges to the A-PUF.

5.5.2 Authentication server

Unlike the physical device, the server contains a software implementation of BILBO, LFSR, FSM and A-PUF model built using the algorithm described in Section 5.4. The blocks labeled LFSR and BILBO perform exactly the same functions as those described for the authenticable device in Fig. 5.12. ECC or majority voting is not needed as the A-PUF model has already been trained with CRP quadruples with known stability status. It is capable of accurately predicting the class of quadruple response based on the $N-2$ middle bits of a challenge. The FSM model in the server also generates a random 8-bit number N_{Rounds}, which is sent to the device to be authenticated. The response analyzer in the server compares the response R_{Final} received from the device, and the response R_{Model} generated by it's A-PUF model for the same N_{Rounds} in each authentication session to determine if the device is authentic. The descriptions of server's software modules are summarized in Table 5.3.

Table 5.3: Software components of authentication server block.

Component	Description
LFSR	Python script for generating pseudorandom $(N-2)$-bit challenge CH using the same seed as the device to be authenticated.
BILBO	Python script for lossy compression of CH to CH', using the same seed and polynomial coefficients as the device to be authenticated.
A-PUF	Deep neural network modeled in TensorFlow (Python) for generating an 8-bit R_{Model}. As a precise software model, SREG and ECC is not required.
FSM	Python script for generating an 8-bit N_{Rounds} for both authenticable device and authentication server.
Response analyzer	Python script to encode and assemble $\frac{K}{4}$ R_{Model}. It also compares R_{Final} received from the device with R_{Model} generated by the server.

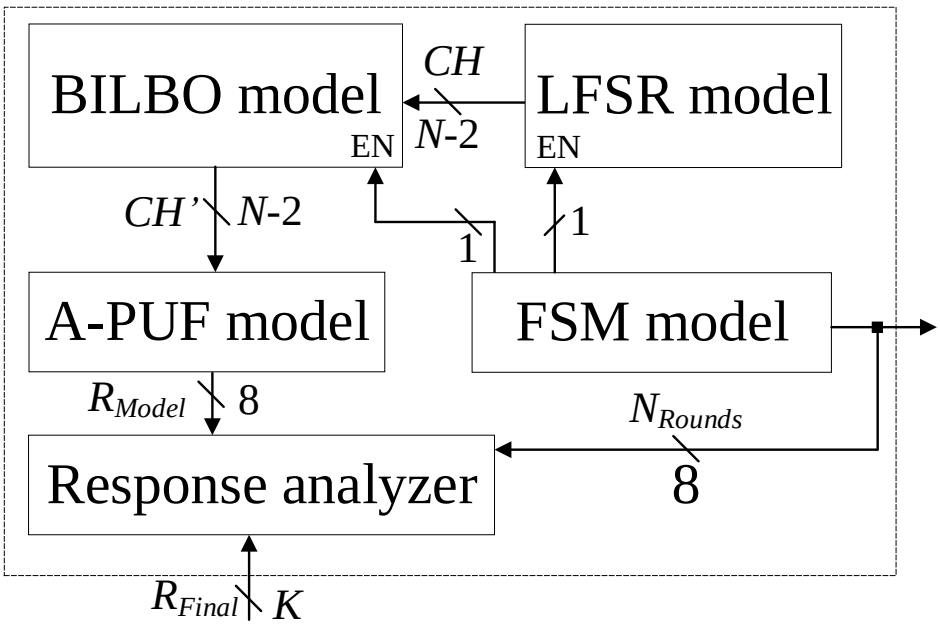


Figure 5.12: Authentication server block diagram.

5.5.3 Authentication Protocol

As shown in Fig. 5.13, the communication protocol consists of three phases, which are enrollment, authentication and update.

The first phase involves a trusted party, who provides the initialization data for both the device and the server. During this phase, the device (BILBO block) is set to memory mode (M = "10") to enable the FSM to feed any challenge CH directly to the A-PUF block without obfuscation. In this mode, the FSM can feed an arbitrary quadruple challenge to the A-PUF to produce an 8-bit response R_Q. Each N-bit quadruple challenge CH is repeated many times to gather a majority voted quadruple response R_Q. This way a large number of reproducible quadruple CRPs can be collected from the physical device. This will enable the machine learning algorithm presented in Section 5.4 to create an accurate A-PUF model to precisely predict the encoded quadruple response (one of the five response classes) to any input quadruple challenge specified by their middle bits CH'. The number of quadruple CRPs required to build an accurate A-PUF model varies from few millions for N = 24 to few billions for N = 128. Therefore, this phase can take a significant amount of time (from hours to weeks). The built model is uploaded to the server.

The device and server share some common parameters of BILBO, which deter-

mine the actual challenges to the A-PUF and its final responses. These parameters are namely the *seed* value or initial value of MISR circuit and polynomial coefficients, which are used to generate the real challenge to the physical A-PUF or its mathematical model at the server. The feedback polynomial and seed value of the LFSR block are different from those of the BILBO block and they are set once during the design phase. As shown in our preliminary work [131], the key parameters of MISR circuit can be fixed or stored in the reconfigurable memory for further updating. The last step of this phase is the MISR mode activation after all the parameters have been set and the A-PUF model has been created. The memory mode of MISR will be permanently disabled by the FSM after the enrollment phase to prevent the attacker from accessing the "raw" CRPs during other phases.

The authentication phase is the main mode of operation that is invoked most frequently. The server responds to the device's request for service by sending a random number N_{Rounds} to initiate a request to authenticate the device before the service can be granted. Based on the value of N_{Rounds}, the device uses the LFSR block to generate the challenge CH, stores it and feeds it to the BILBO in MISR mode. N_{Rounds} of iterations are executed in MISR mode to generate an obfuscated challenge CH' to the PUF to produce one quadruple response. During the authentication phase, $\frac{K}{4}\ N_{Rounds}$ are generated at the server and sent to the device. A K-bit R_{Model} is computed at the server to compare with the K-bit R_{Final} received from the device. R_{Model} is calculated in almost the same manner as that on the device except that the computations are performed in software by the A-PUF and BILBO models instead of physically by the hardware. The authenticity of the device is confirmed by $R_{Final} = R_{Model}$ and denied by $R_{Final} \neq R_{Model}$.

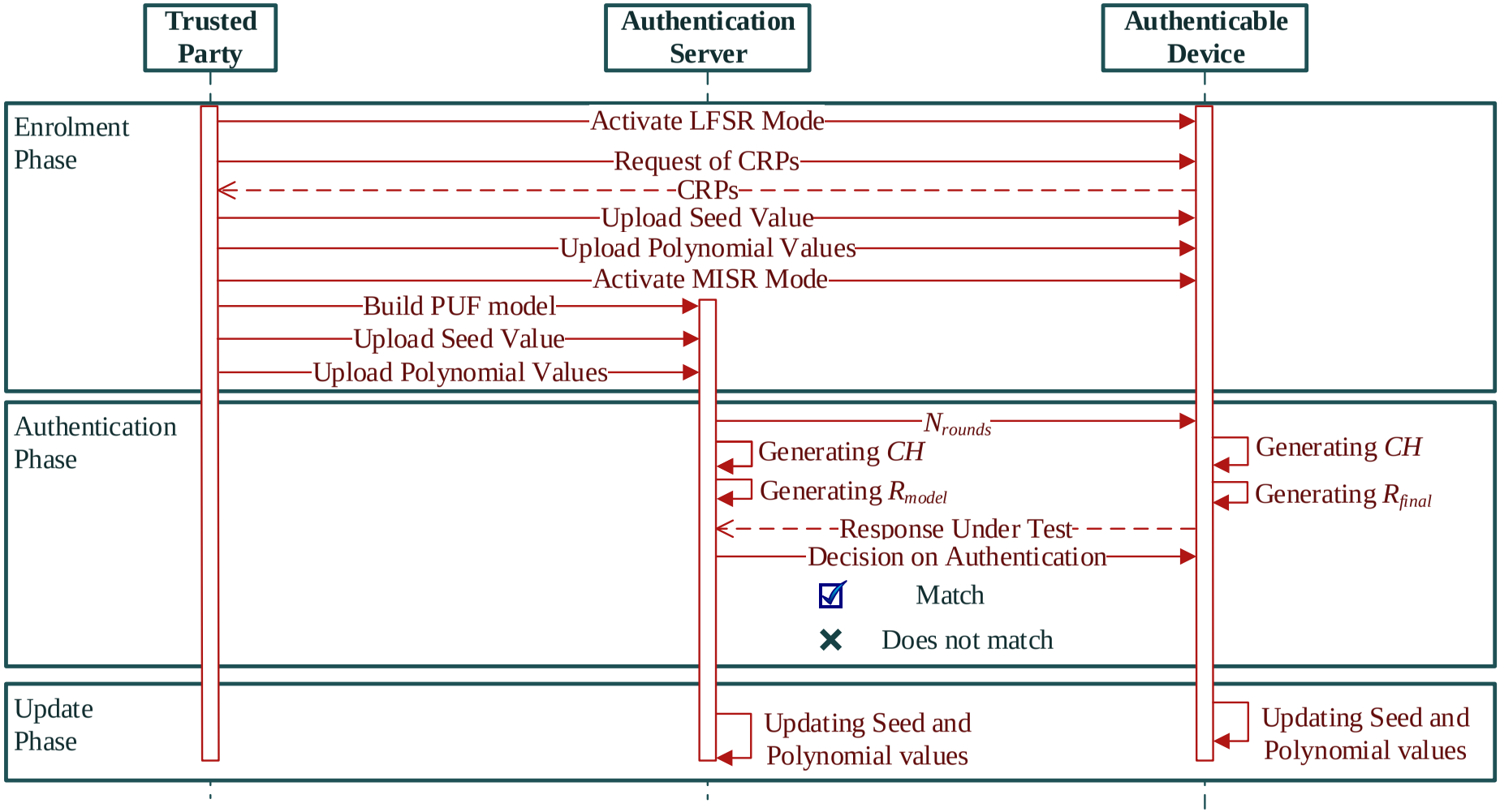


Figure 5.13: Authentication protocol.

If the parameters of BILBO are stored in reconfigurable memory, then an update phase can also be invoked in the proposed protocol. The parameters can be updated to new values from the memory regularly or after certain number of successful authentications. The total number of available polynomials can be estimated using the Euler's totient function. For $N = 128$, the number of GF(2) polynomials is $\approx 1.3 \times 10^{36}$. The number of possible seed values is also exponentially large (2^N). Thus, these values can be randomly chosen and the number of different combinations will not be exhausted throughout the device life cycle for N bigger than 64. The choice of exact number of parameters to be stored in the memory depends on the application and on the trade-off between hardware overhead and system security.

5.6 Security Analysis

5.6.1 Attacker Model

According to Kerckhoffs' principle [132], attacker knows everything about our system except the keys. In our proposed authentication protocol, the following five secrets are shared between the authenticable device and authentication server: seed value of LFSR, polynomial values of LFSR, seed value of BILBO, polynomial values of BILBO, challenge-response map of A-PUF.

We assume that the attacker has access to all data transmitted between the device and the server. The attacker can exploit the collected data to produce correct responses to the N_{Rounds} initiated by the server to gain successful authentication. The attacker is assumed to have access to only the device external IO ports to directly apply authentication request or measure any side-channel signals by direct wired connection in attempt to recover the secret parameters of LFSR or BILBO. The incentive of the attack vanishes if the device is damaged or the tamper can be conspicuously detected before the attack is successful. The protocol is considered secure if the attacker is not able to predict the correct responses under known approaches notwithstanding the full access to the transmitted data and IO pins of authenticable device. We further assume that the profitability of an attack diminishes rapidly if the attacker has to continuously deploy significant computing power to perform the attack and analysis beyond a realistic time span. Hence the protocol can be considered secure if the time required for a successful attack grows exponentially with the number of stages N of A-PUF.

Side-channel attacks require expensive equipment, and well-timed and accurate measurements. They rely on known exploitable implementation weaknesses of LFSR

and BILBO, which can be easily circumvented. Since the ultimate goal of the attacker is to successfully predict the response to unknown and unused challenge of the embedded A-PUF, which can be more effectively achieved by modeling the propagation delays of the parallel and swapped paths of each switch of the A-PUF than side-channel attacks, machine learning attacks [77] are emphasized in our security analysis.

5.6.2 Modeling attack

This attack requires the attacker to collect a reasonable subset of CRPs to build an accurate model of the PUF. One of the most efficient methods to perform this kind of attack is machine learning. Since an A-PUF can be approximated by a linear classifier (e.g., LR or SVM), the proposed MISR based challenge obfuscation creates a non-linear dependency of the response on the naked challenge.

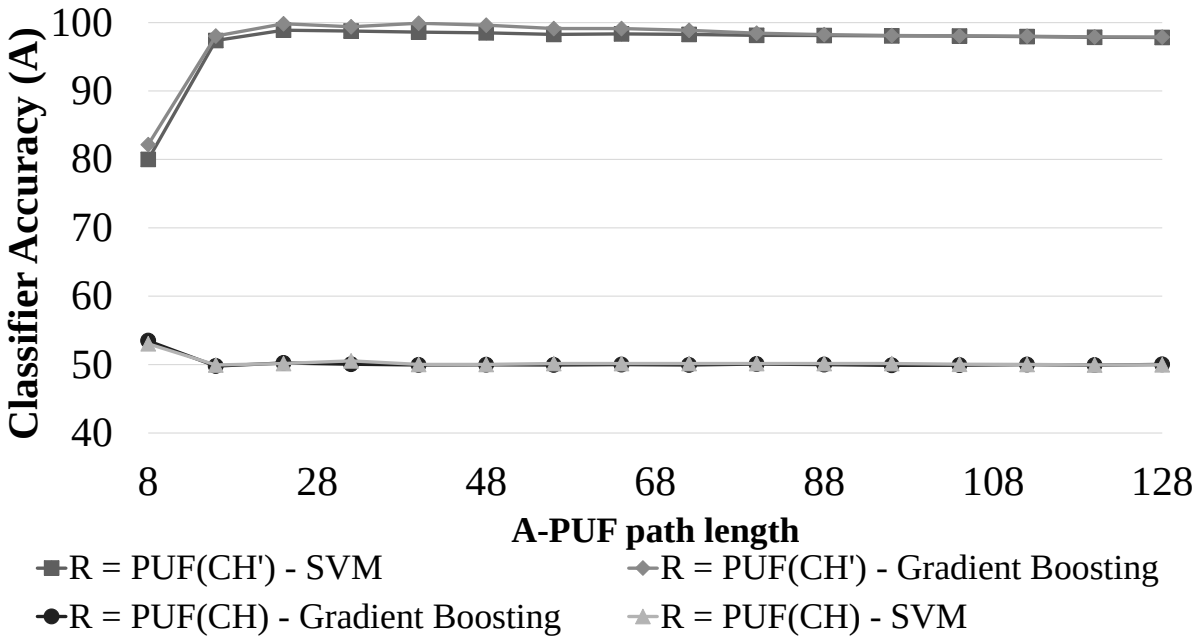


Figure 5.14: Efficiency of SVM and Gradient Boosting modeling attacks.

To estimate the effectiveness of the proposed challenge transformation algorithm, a 128-stage A-PUF with SR latch based multi-arbiter [110] is implemented in Xilinx Artix-7 FPGA. This circuit has 16 arbiters uniformly allocated to sections of 8 multiplexer pairs to emulate different lengths of A-PUF path. Meantime, a MISR circuit with feedback polynomial $\phi(X) = x^{128} \oplus x^{127} \oplus x^{126} \oplus x^{121} \oplus 1$ is implemented on the same FPGA chip to transform an arbitrary 128-bit input challenge CH to an obfuscated challenge CH'. Both CRP (CH, R) and CRP' (CH', R') are split

into 16 data sets according to the emulated A-PUF path length to generate a 16-bit response from each arbiter instance. A host computer is used to generate a uniformly distributed sequence of 128-bit challenges, i.e., $C \in [0, 2^{128} - 1]$, for input into the MISR of the FPGA chip. To test the resistance of the proposed MISR augmented A-PUF against modeling attacks, SVM with radial basis function (RBF) kernel and gradient boosting with 10,000 trees and learning rate of 0.1 are implemented on the host computer using Sklearn library of Python. 10% of the randomly chosen CRPs are used as the training set, which is cross-validated using 5 blocks K-fold method. The remaining 90% are used as the test set. Both algorithms are capable of discriminating two classes of responses using non-linear surface. The attack resistance is evaluated by an accuracy score in terms of the percentage of successful classifications.

The results are shown in Fig. 5.14. If the machine learning model is trained by the CRP' data set, which is inaccessible to the attacker, high accuracy scores of [97.38, 98.34]% for SVM and [97.88, 99.83]% for gradient boosting are obtained. The only exception is found in the 8-stage A-PUF emulation, where the scores are 80% for SVM and 82.17% for GB due to the limited number of CRPs (only 256). This is because A-PUF with less than 16 stages has very unstable behavior [110]. For path longer than 8 stages, the prediction accuracy reduces slightly from 98.89% to 97.83% for SVM and from 99.83% to 97.88% for GB and for 128 stages, the predication accuracy is almost the same for both attack algorithms. Since the attacker can only access to the CRP data set before the challenge C is transformed into $\hat{C}$, the prediction accuracy fluctuates around 50% for both SVM and GB, as shown in Fig. 5.14.

Recent research [68] shows that secure designs based on A-PUF [128] can be successfully attacked using powerful multi-core CPU and GPU servers even if the attacker does not know the system, i.e., by treating the PUF system as a black box. Therefore, we have simulated the CMA-ES attack using 24-core Intel Xeon CPU server with 32 GBytes of RAM on both 24 and 128-bit A-PUF circuits with MISR based challenge obfuscation algorithm. CMA-ES algorithm was implemented using PyBrain library. The results are shown in Fig. 5.15 (solid circle and triangular points are measured experimental results and the hollow points are extrapolated results). The extrapolation is justifiable as the longest physical experiment (the fourth solid triangle point in Fig. 5.15) conducted on 128-bit A-PUF took more than two months. It is made based on the four physically conducted experiments for each of the 24-bit and 128-bit A-PUFs. When the size of the training CRP set is increased by tenfold in each experiment, the accuracy of machine learning algorithm increases only marginally by 2.79% to 2.97% for the 24-bit A-PUF and 0.89% to 1.08% for

the 128-bit A-PUF, but the training time increases exponentially by 2.07x, 5.17x to 10.76x for the 24-bit A-PUF and 1.43x, 5.38x and 10.96x for the 128-bit A-PUF. For larger training sets, we deliberately underestimate the training time and overestimate the training accuracy by conservatively assuming a maximum of 3% and 1% in our extrapolation of the increase in accuracy for 24- and 128-bit A-PUFs, respectively, as well as a lower than expected flat 10x growth rate in training time for every decade increase in the size of the training CRP set.

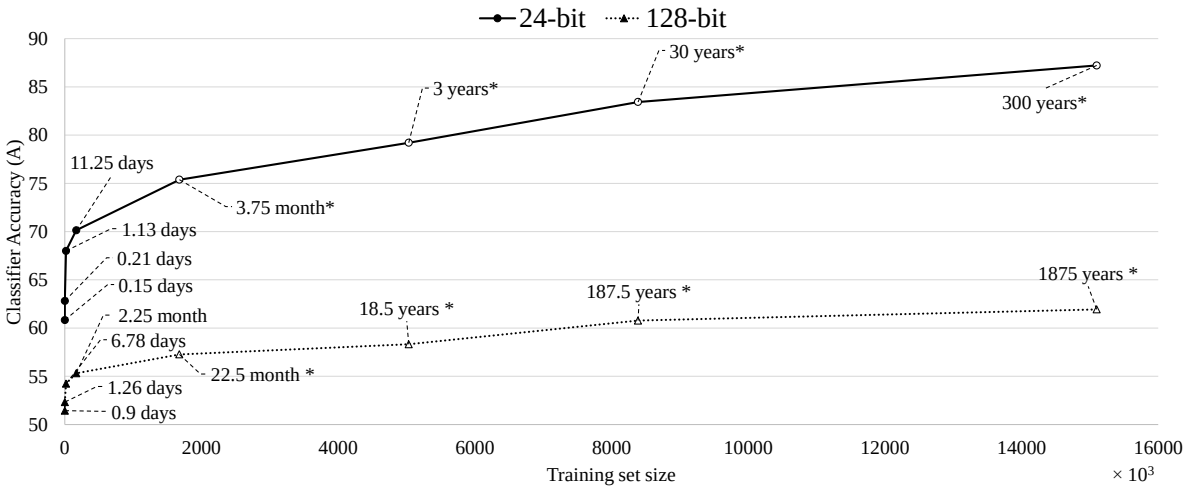


Figure 5.15: Evolution strategy attack results on both 24 and 128-bit A-PUF.

Although the classifier accuracy increases with training set size, this attack cannot succeed for two practical reasons. First, the number of CRPs required to build a reasonably accurate model is significantly greater than that required for authentication throughout the device lifetime. To achieve an accuracy of 60% for 128-stage A-PUF, the number of required CRPs exceeds 16 millions. If the training data is to be collected within a year, then 44943 CRPs have to be extracted per day. If each authentication requires a response of 64 bits or less for validation, then no more than 32 CRPs (as each challenge produces a two-bit trinary response) can be collected with every authentication request. The device will have to make at least 1404 authentications a day, which is close to one authentication per minute. This has yet taken into consideration the training time. If the prediction accuracy of the model is to be increased by just another 5%, the attacker will have to continuously query the device at sub-millisecond rate for one year just to collect enough CRPs for training. It is hard not to burnout the device or remain stealthy at this rate of authentication. Secondly and most importantly, the time required for a successful attack has exceeded the lifetime of an ordinary person. Even if the computing power can be increased by

100 times, it will still require a couple of years to achieve a poor prediction accuracy of only 60%.

Thus, the proposed algorithm is practically secure against different machine learning algorithms as it requires huge subset of CRPs with training time exceeding the device lifetime. If any of the BILBO parameters is reconfigured, the attack will have to start all over again as the successfully learned parameters have been changed.

5.6.3 Random Guessing

There are two parameters in the proposed design that the attacker may determine by trial and error. They are the *seed* value and the number of possible polynomial $\phi(X)$ coefficients. The probability of guessing both values correctly in any one attempt can be estimated by:

$$P_{Guess} = \frac{N}{2^{2\times N} - 2^N}. \tag{5.5}$$

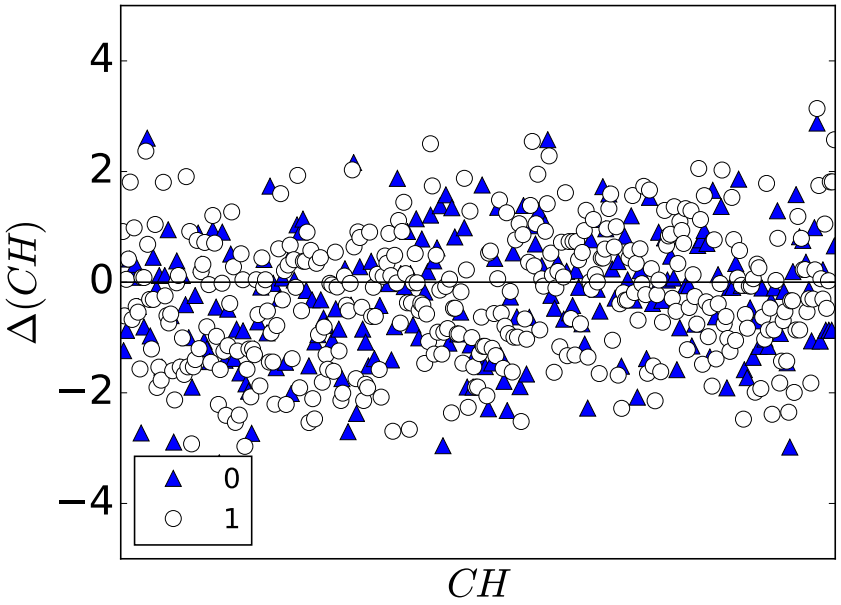
If $N = 24$, $P_{Guess} = 8.52 \times 10^{-14}$, which is equivalent to more than billion years of computation to break the system assuming that it takes one second to make and apply a guess. It is probably easier to perform a brute force attack by exhausting the whole challenge response set (2^{24} CRPs) to compromise these parameters. Since the actual CRP set is inaccessible to an attacker, this attack is not possible.

5.6.4 Compromising seed and polynomial coefficients of BILBO

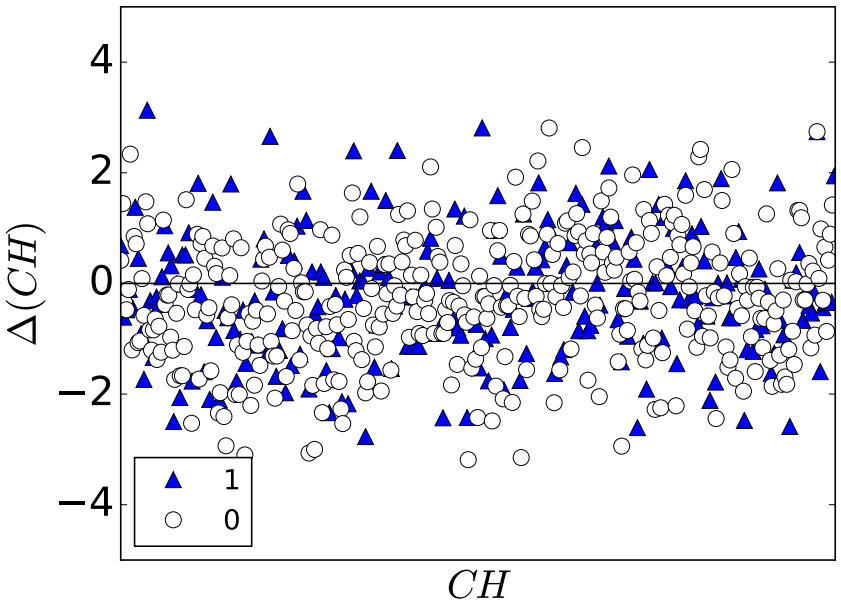
Performant physical and side channel attacks on strong PUFs have been reported [133]. These attacks assume that the attacker has access to the device to perform controlled measurements to collect a sufficient number of side-channel traces without being discovered. The attacker is also assumed to have complete understanding of the device protection to determine the most appropriate analysis for the exploitable side-channel signal, provided that such side-channel leakage can be accurately measured. Side channel attacks, even if feasible, help only to derive the approximate but not the exact values of the key parameters of BILBO circuit. It can at best reduce the search space for machine learning algorithm to boost modeling attack. According to [133], the training time can be reduced by 200 to 300 times, which means that in the worst case, the training time for the proposed 128-bit A-PUF design will be shortened to around 5-6 years to achieve an accuracy of 60% assuming that the attacker is able to measure exploitable side-channel leakage with similar accuracy. The attacker may, however, independently attack LFSR and MISR by side channel analysis [134] and

then use the information to model the A-PUF. To successfully deduce four of the five secret parameters (seed and polynomial values of LFSR and MISR), without access to the internal nets, a large number of accurate power measurements is required due to the interplay between these components and PUF. As the computations are controlled by the internal FSM and triggered by authentication request, the power traces have to be collected during authentication with proper synchronization between the device and assumed server, which make the attack challenging.

To demonstrate the demand of accurate measurements required for a successful side channel attack, two different seed values, 00...0 and 00...1 are used to simulate a 16-stage A-PUF with its challenges pre-processed by a MISR with feedback polynomial $\phi(X) = x^{16} \oplus x^{15} \oplus x^{13} \oplus x^4 \oplus 1$. In both cases, $M = 65534$ challenges are applied to the MISR in the same order. The distribution of the two sets of CRP obtained are shown in Fig. 5.16, where CH denotes the challenge and $\Delta(CH)$ denotes the delay difference in ns between the two paths selected by CH. Around 99% of the path delay values have changed, even with only a LSB change in the MISR *seed*. As a small error in the estimation of BILBO parameters has an avalanche effect on the CRP set, the cost of side-channel analysis increases significantly with the increased resolution of measurements required for an accurate modeling while countermeasures against side-channel attacks are made easier by reducing the magnitude of measurable side-channel leakage. For example, additional T Flip-Flop register can be used to mirror the LFSR. Each T Flip-Flop toggles when the corresponding D Flip-Flop in the LFSR is not switching and vice versa to eliminate any measurable difference in switching power. Changing the polynomial coefficients in reconfigurable memory will have similar effect. The collected traces are useless when the coefficients of the MISR has been reconfigured. Taking into consideration the training time required, 2-3 updates of polynomial coefficients or *seed* of MISR is adequate during the device lifetime to thwart combined modeling and side-channel attacks. The secret parameters can be securely refreshed by encrypting them using symmetric cipher before they are sent to the device. A simple secure protocol to refresh these parameters making use of the existing A-PUF is described as follows. The authentication server generates a challenge C and applies the hash CH_H of CH to the A-PUF model to obtain a response $R = \text{PUF}_{Model}(CH_H)$. Then, server calculates $h_1 = p_{New} \oplus R$ and the hash value h_2 of p_{New}, where p_{New} is the value of the secret parameter to be updated. The values h_1, h_2 and CH_H are transmitted to the device. The device applies CH_H to its A-PUF to obtain the response $R = \text{PUF}(CH_H)$. Thus, p_{New} can be recovered by $R \oplus h_1$. Meantime, the updated parameter is also authenticated by checking if h_2 is equal to



(a) *seed* is $0 \cdots 00$



(b) *seed* is $0 \cdots 01$

Figure 5.16: Distribution of differential delays (in ns) for MISR obfuscated challenges.

the hash value of the recovered p_{New} before it is updated on the device. To confirm the successful update of p_{New}, the device calculates a signature $S = \text{PUF}(p_{New})$ and

sent it back to the server. The server verifies if S equals to $\text{PUF}_{Model}(p_{New})$ before switching to the new parameter for future authentication.

5.6.5 Spoofing attacks

During the authentication phase, the only values transmitted between the server and the device are N_{Rounds} from the server and R_{Final} from the device. Knowledge of these two information does not allow the attacker to obtain the real CRPs of the PUF. The attacker cannot falsify authentication by eavesdropping the communications to replay R_{Final} to N_{Rounds}. This is because the state of MISR depends not only on the current input challenge but also the past obfuscated challenges. A different output will be generated by the BILBO block each time even for the same N_{Rounds}. Knowing only the number of BILBO loops required to obfuscate the challenge from each authentication session does not allow the attacker to generate the correct response similar to a real device. Even if the attacker manages to apply the same challenge to the BILBO block, the responses from the device will be vastly different as the MISR transformation is lossy and cannot be uniquely recovered. Therefore, the only chance for an attacker to spoof the server is to build a mathematical clone of the device, which amounts to the same cost and difficulty as a successful machine learning attack. Any other manipulation of the original messages passing between the server and device may compromise the device from being successfully authenticated but cannot gain illegitimate privilege in accessing the services offered by the server as the authentic device.

5.6.6 Exploiting PUF instability

Recent research [69] has pointed out that the use of only selective stable CRPs for device authentication can be hazardous [127, 135, 136, 137]. These protocols validate the response based on the Hamming distance between the received and enrolled responses. This may allow the attacker to correlate the distance between the noisy response and the private parameters of the protocol. For example, FSM PUF [136] relies only on 1% of the CRP set to avoid the reliability issues of PUF but this gives the attacker an opportunity to deduce the path delay difference from the transmitted response. The proposed protocol utilizes the entire CRP set but corrects the unstable responses internally according to their response classification to achieve a reliability of 1.0 without having to transmit helper data that will reveal sensitive information. Furthermore, the MISR transformation is lossy and after a few iterations, it is irreversible

and non-reproducible from the same input challenge. Thus, the proposed authentication protocol is practically secure against reliability-based side-channel modeling attacks. Its ability to refresh configurable private parameters makes it even harder for the attacker.

5.7 Experimental Results and Discussions

The quality of the PUF design is evaluated using three widely adopted figures of merit, which are reliability, uniqueness and randomness.

5.7.1 Reliability test

To achieve a reliability of 1.0 using the proposed design, ECC codes are required. The proposed A-PUF implementation contains repetition codes to determine the class of a quadruple response value. The responses were further tested under varying operational conditions. The vendor specified allowable temperature range is [-40° C, +90° C] for Nexys-4 evaluation board. All temperature tests were performed with Thermotron ®8800 temperature chamber. The results are shown in Fig. 5.17. It takes only $k = 5$ repetitions for the ECC to achieve the reliability of 1.0.

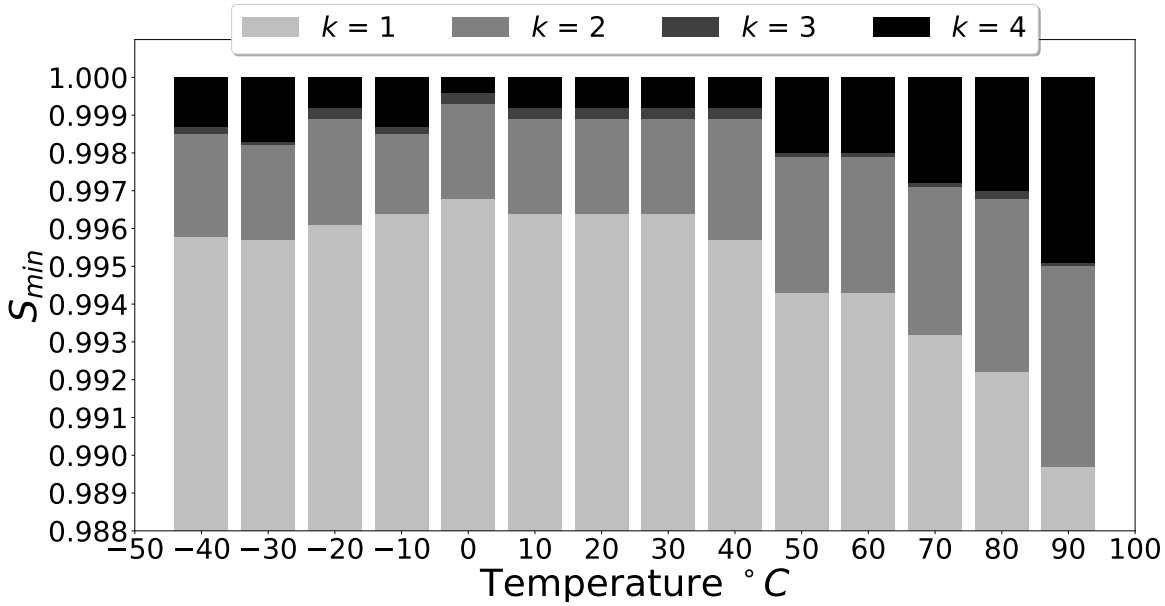


Figure 5.17: Results of temperature test.

5.7.2 Uniqueness test

K = 10,000 challenges have been generated for the uniqueness test. All challenges have been generated in a MISR mode and repeated in the memory mode. m = 20 Xilinx Nexys 4 FPGA chips are used for the implementation of different PUF instances. The results are shown in Fig. 5.18. The uniqueness exhibits a normal distribution, with a mean of 0.497 and a standard deviation of 0.011 for inter-chip HD. This is equal to the ideal uniqueness of 0.5 with accuracy up to one decimal place. This implies that the zeros and ones in the MISR based challenge transformation has no negative effect on the uniqueness of the original A-PUF design.

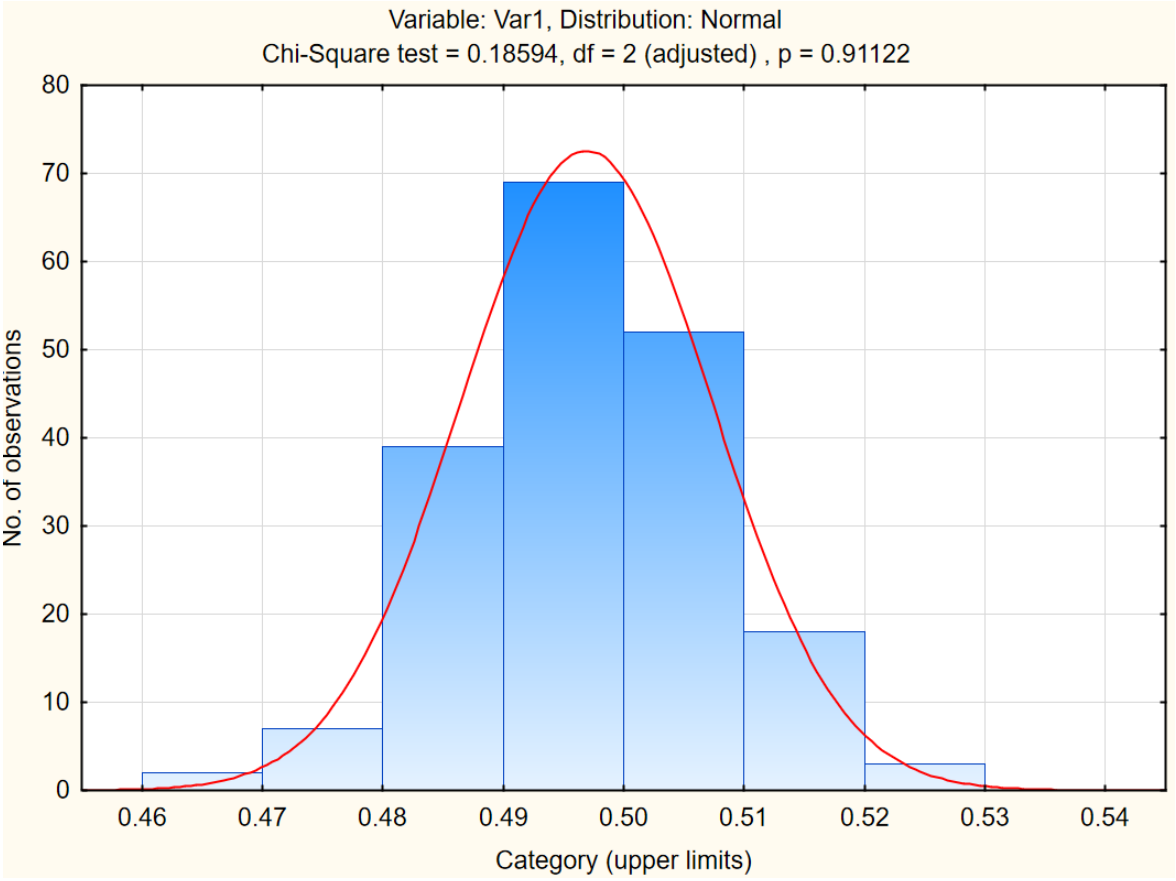


Figure 5.18: Uniqueness of proposed A-PUF design.

5.7.3 Randomness test

The cryptographic randomness of a True Random Number Generators (TRNG) can be validated by the 15 statistical tests provided by the National Institute of Standard and Technology (NIST) statistical test package [88]. A sequence of 10 million bits

has been generated in MISR mode of operation. This bit sequence has been divided into 100 blocks, each of 100,000 bits. The NIST test results in Table 5.4 show that the response bits generated from the processed challenges passed all the NIST tests successfully.

Table 5.4: NIST randomness test results.

Test Description	**Passed/Total**	**P-value**
Frequency (Monobit) Test	98/100	0.04
Frequency Test within a Block	100/100	0.53
Runs Test	100/100	0.74
Test for the Longest Run of Ones in a Block	100/100	0.53
Binary Matrix Rank Test	100/100	0.35
Discrete Fourier Transform (Spectral) Test	100/100	0.46
Non-overlapping Template Matching Test	98/100	0.57
Overlapping Template Matching Test	96/100	0.35
Maurer's "Universal Statistical" Test	100/100	0.05
Linear Complexity Test	97/100	0.08
Approximate Entropy Test	97/100	0.05
Serial Test	100/100	0.73
Cumulative Sums (Cusum) Test	98/100	0.04
Random Excursions Test	10/10	—
Random Excursions Variant Test	10/10	—

Besides, the bit sequence produced by the A-PUF with enhanced reliability of 1.0 still possesses good entropy, which matches the results obtained earlier with multiple A-PUF using MISR circuit [95]. Furthermore, it can also be post-processed by hash function similar to what has been done with the raw responses of RO-PUF, SRAM PUF and other designs before it is used as a true random sequence.

5.7.4 Hardware overhead

The proposed protocol can be viewed as an obfuscated strong PUF based authentication. Therefore, it is compared with several approaches from this category. Table 5.5 compares the FPGA resources (LUT and DFF blocks) consumed by different 64- and 128-stage A-PUF based protocols with the authenticable device implemented on Xilinx Artix-7 FPGA chip.

The proposed design is efficient in terms of the overall utilization rate of LUT blocks and FFs. It uses more LUTs than PolyPUF and slightly more LUT than RPUF but significantly less FFs than both of them. Although CMOS PUF uses less FFs, it has a smaller CRP set after filtering out the unreliable ones. Since the

Table 5.5: Comparison of FPGA resource usages.

Approach	N	#LUT (% of available)	#FF(% of available)
PolyPUF [135]	64	213 (1.34)	450 (2.84)
Slender PUF [128]	64	652 (4.11)	1400 (8.83)
OB-PUF [127]	64	680 (4.29)	360 (2.27)
RPUF [137]	128	350 (2.21)	389 (2.45)
PUF-FSM [136]	128	960 (6.06)	1500 (9.46)
CMOS PUF [138]	128	395 (2.49)	176 (1.12)
This approach	128	380 (2.39)	264 (1.67)
This approach	64	192 (1.19)	132 (0.84)

proposed A-PUF leads to a more secure protocol with increased resistance against modeling attack by both basic and advanced machine learning algorithms, the sacrifice of around 2% of hardware resources of a FPGA chip is well justifiable. Moreover, if the BILBO block already exists in the system for BIST, it can be reused for this purpose to make the overhead twice smaller.

As machine learning attack can also be prevented by encrypting challenge and/or response values as stated in [139], the hardware resources required for the implementation of two standard cryptographic techniques, namely symmetric encryption (fast and compact versions of 128-bit Advanced Encryption Standard (AES)) and Secure Hash Algorithm (SHA-256 of SHA-2 mentioned in [139]) on the same Xilinx Artix-7 FPGA are also evaluated and compared. The number of LUTs and Flip-Flops used, and accuracies of SVM and CMA-ES based modeling attacks on these implementations are compared in Table 5.6. The results show that existing cryptographic approaches to enhance ML-resistance of PUF require around twice to ten times more FPGA fabrics than the proposed MISR obfuscation method.

Table 5.6: Comparison with standard cryptographic techniques.

Technique	#LUT	#FF	SVM	CMA-ES
This	380	264	50%	56%
SHA-256 [140]	3756	2609	50%	50%
AES (small) [141]	888	444	50%	50%
AES (fast) [142]	1642	820	50%	50%

5.8 Summary

This chapter reviews three methods (SHA-256, TFF Register and MISR circuit) to convert an A-PUF into a controlled PUF to thwart modeling attacks. These tech-

niques are analyzed for hardware overhead and resistance to attacks by simple and advanced machine learning algorithms. The MISR based obfuscation algorithm is chosen due to its acceptably low resource consumption and high security level. This chapter also presents an approach to generate highly reliable CRPs using SR latch based A-PUF in FPGA implementation. The method is based on both metastability detection and CRP classification to detect relatively stable and unstable challenges. It has been demonstrated that combining this approach with repetition codes ($k = 4$) can achieve a perfect reliability of 1.0 over the temperature range of $-40°$C to $90°$. On the other hand, using only highly stable CRPs can increase the vulnerability of A-PUF design to machine learning attacks. Therefore, the proposed design is augmented with the BILBO module to obfuscate the linear dependency between raw challenge and response. An authentication protocol is then proposed based on the construction of a 100% precise A-PUF model, which is safely stored on the server side and the BILBO block with reconfigurable parameters. This approach significantly increases the resistance against CMA-ES modeling attack as it demands huge CRP set (more than billions) and high computational time (hundreds of years) for the attack to be successful. Thus, the proposed authentication scheme has overcome the practical challenge of meeting concurrently the desiderata of high reliability, high level of security and lower hardware resource overhead.

Chapter 6

Conclusion and Future Works

6.1 Conclusions

Being a very new and fast growing concept, IoTs has a lot of unresolved problems such as but not limited to legislation, standardization, privacy and security, maintenance and resources utilization. Lack of proper security standards in IoT carries many possible threats, which can be hazardous for human health via stealing or modifying medical information. Biomedical sensors have rigorous requirement on chip area and power efficiency. Consequently, privacy protection and security assurance demand lightweight authentication protocols and security primitives. For the last 15 years, PUFs have found their place in security device market, which is confirmed by the emergence of commercial companies like Verayo. Furthermore, flourishing FPGA market, and power and performance efficiency gap between modern FPGA and ASIC chips bring major vendors such as Xilinx, Intel, Microsemi, etc. into competition for modern IoT solutions. This is one of the reasons that embedding PUF into the high-end FPGA chips for mass production has become a new normal for the industry. Nowadays, PUF research is dominated by constant search for new architectures that can harness greater variability from nanoscale fabrication while efficiently address post-manufacturing reliability problems in rugged operating environments and new threats. On the other hand, as mature and well known "classical" architectures are either ready to be implemented or have already been deployed in some industrial designs, they too deserve decent attention and continuous enhancement.

This research aims to improve existing problems of Arbiter PUF design in two aspects: increasing the reliability of its deployment and fortifying the device and protocol to overcome their vulnerability to modeling attacks. The first problem is especially severe in FPGA implementation as the designer has much less control of the placement and routing process and can only use existing unused fabric or building

blocks. Thus, a new way of addressing the reliability of classical Arbiter PUF design on FPGA is needed given that such restrictions are inevitable. One main reason for the temporal instability of some of its responses is attributed to the metastable state of the arbiter circuit. Unresolved metastable state is indetermistic and constitutes to noisy response due to its sensitivity to perturbation. Our experiments have shown that filtering out of these unreliable responses reduces the random components, making Arbiter PUF even more vulnerable to modeling attack as the linear additive mathematical model can be more readily derived from a smaller subset of the challenge-response space.

In a nutshell, this thesis presents techniques for reliability and unpredictability improvement of Arbiter PUF in FPGA implementation. A new authentication protocol is proposed to efficiently fuse the high reliability PUF with lightweight obfuscation method to protect the authentication against modeling attacks by common as well as more complex advanced machine learning algorithms. The main contributions of this thesis can be summarized as follows:

- Metastability detection techniques have been proposed for Arbiter PUF implemented in FPGA. Existing classical arbiter circuit design based on single D Flip-Flop is more susceptible to metastability. If the output falls into metabstable state, the response bit (logic 0 or 1) generated is unpredictable with approximately equal probability. Being metastable, it is highly susceptible to small change in operating conditions. The proposed techniques use 4 DFFs or SR latch to detect response bits that fall into metastable region of arbiter. The metastable states are assigned a ternary logic of A-PUF response alphabet, which can be converted to one of the two stable logic state upon their detection. These solutions provide improved reliability of A-PUF circuit. The proposed circuits have been tested for standard PUF figures of merit (reliability, uniqueness and randomness). Despite reliability improvement, the other PUF figures or merit are not degraded. Thus, these techniques can be used efficiently in FPGA implementation of Arbiter PUF.

- A challenge classification algorithm is developed from the known linear additive mathematical model of Arbiter PUF. The challenges are grouped into quadruples. The classification that distinguishes their relative response reliability is supported by additional analysis of metastable state according to the locations of metastability regions. Technique has also been developed to identify and filter out persistently reliable challenge-response pairs. The proposed method has

been evaluated by simulation using Xilinx Artix-7 FPGA family and physically implemented in both Xilinx ZC706 and Digilent Nexys-4 evaluation boards containing different FPGA chip families. The experimental results have corroborate the precision of such concise classification of CRPs. The stable and unstable CRP sets so classified can be used for different purposes. For example, key generation for stable CRPs and True random number generation for unstable CRPs. The A-PUF with unreliable CRP filtered out by the proposed classification algorithm has achieved a reliability score of 1.0 over a wide temperature range without uniqueness degradation.

- Achieving high reliability score for A-PUF design immediately opens the possibility for modeling attack due to the increased linearity for mathematical modeling with reduced CRP space. Basic machine learning attacks based on Support Vector Machine algorithm have been addressed by using general approach of controlled PUF but utilizing three different circuits, SHA-256, T Flip-Flop (TFF) register and Multiple Input Signature Register (MISR). These three techniques have been analyzed for their hardware overhead, resistance against basic machine learning attacks and potential vulnerability to advanced methods such as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). The choice of obfuscation scheme for A-PUF design is a trade-off between the level of security and required resources. SHA-256 method is found to provide the highest level of security but requires 10 times more hardware overhead than the other two obfuscation methods. Basic level of security and low overhead is provided by TFF register while MISR obfuscation strive a good balance with a practically acceptable security level and reasonably low resource consumption that can still support lightweight authentication.

- For further improvement of FPGA based A-PUF design, machine learning attack resistant authentication protocol has been designed. Proposed protocol relies on two factors, the high reliability of A-PUF at the device side and a precise modeling to avoid storing the CRP set at the server side. This is achieved by applying the two techniques presented earlier in this thesis, i.e., metastability detection and challenge classification augmented by lightweight error correction codes to avoid the reduction of CRP set. The A-PUF with reliability of 1.0 can be precisely modeled using Deep Neural Network, which is further used on the server side to avoid transmitting the actual challenges and responses. Thus, the proposed protocol is shown to possess practically high resistance against

CMA-ES attack as the attack demands billions of CRPs and hundred of years to achieve a prediction accuracy of only 0.6, which is not attractive for the attacker. The design has also reconfigurable memory to change the key parameters of the obfuscation circuit to invalid all the past successful trainings of the machine learner.

6.2 Future Works

Since PUF designs are an essential part of hardware metering methods, two directions can be considered for future research efforts. First, the Multi-Arbiter design can be imporved to enlarge the number of response bits for utilization in active metering schemes.

6.2.1 Enhancement of the proposed MA-PUF

MA-PUF design presented in Chapter 3 has significant potential to increase the number of bits in a response word with reasonably low overhead comparing to the A-PUF with a single arbiter. However, some pairs of arbiter outputs have high correlation to each other, which can be potentially used by an attacker to gain additional information about the PUF. Therefore, some study can be made to develop an algorithm for selecting multiple uncorrelated arbiters during the design or operation time by partial dynamic reconfigurability.

There are multiple ways to add arbiters to a classical A-PUF design:

1. **Incremental approach**. Choose random position in multiplexer chain and add the arbiter circuit. If new arbiter is correlated to some of the others, it should be excluded. The process is to be repeated until significant amount of uncorrelated arbiters are added into the design.

2. **Fixed approach**. Designer should add arbiter on every i-th stage, e.g., after every 8 stages for a 128-bit A-PUF, or $i = 8, 16, 24, \ldots, 128$. The parameter of i is determined during the design phase.

3. **Mask approach**. Unlike the above two methods, this method works during operation. All possible arbiters are associated with a mask, which allows a selective subset to be activated during operation. Choosing a correct subset of arbiters is complex task and can be formulated as an optimization problem. The objective function can be the minimization of the number of significantly correlated pairs of arbiters with the binary masks as control parameters.

If this problem can be addressed, the number of response bits for an arbiter PUF design can be increased without compromising security. Furthermore, it may increase the complexity of modeling attack as attacker will have to build a model for each arbiter circuit. Partial dynamic reconfiguration can be exploited to add dynamic obfuscation in changing the CRP space.

6.2.2 Re-licensing scheme for IP protection

A complex system can contain many licensed hardware IP components. Some of them can be compromised by an unauthorized user activity, or the license can be expired due to obsolescence or revoked due to the key being stolen by a third party. Under these circumstances, the license key should be immediately disabled and a compromised IP component should locked again upon expiry or revocation of the key for whatever reasons. An overview of the suggested re-licensing scheme is shown in Fig. 6.1.

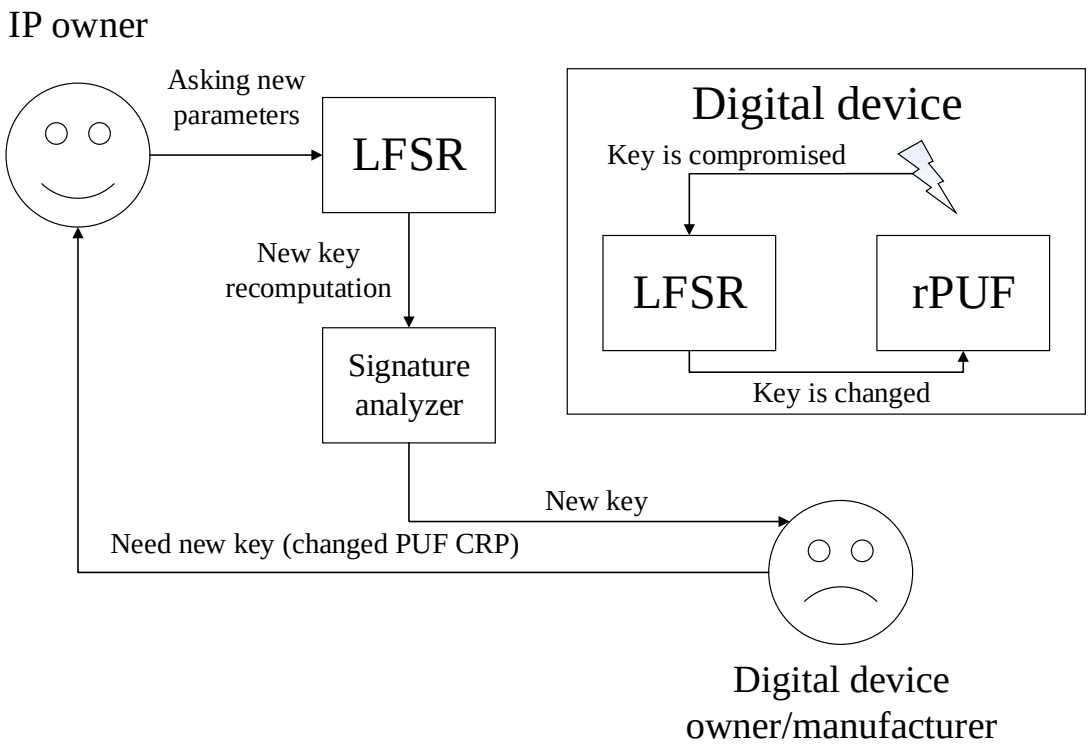


Figure 6.1: Re-licensing active metering modification.

The mechanism for renewing the key is based on a reconfigurable PUF and an LFSR. When the key is compromised, the IP component is automatically locked and

shifted to a next state with a code generated by the LFSR. The IP owner who has access to the LFSR with the seed to generate the key can unlock the IP component. Based on the reconfigured CRPs of the PUF and an LFSR state, the IP owner can generate a new key to the end-customer. Optionally the key could be processed by signature analyzer to prevent it from being used openly as is.

The main difficulty in the implementation of this scheme is a good locking mechanism that have equal flexibility to change its dependency to the renewed CRP space of the RPUF as needed. For the active IC metering approaches based on FSM modification, it is impossible to reconfigure the FSM after synthesis. Therefore, one of the extensions of this research is to find a flexible (reconfigurable) locking mechanism to work with the target RPUFs besides the development of a good RPUF. At present, it seems promising to turn the proposed multi-arbiter into a RPUF for FPGA-based active metering scheme.

List of Author's Publications

Book Chapter

[1] **S. S. Zalivaka**, L. Zhang, V. P. Klybik, A. A. Ivaniuk, and C. H. Chang, "Design and implementation of high quality PUF for hardware-oriented cryptography," in *Secure System Design and Trustable Computing*, C. H. Chang and M. Potkonjak Ed., Springer International Publishing AG, Switzerland, pp. 39–81, Oct. 2015.

Journal Paper (published)

[2] Y. Cao, L. Zhang, **S. S. Zalivaka**, C. H. Chang and S. Chen "CMOS image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 11, pp. 2629–2640, Oct. 2015.

Journal Paper (revised paper submitted)

[3] **S. S. Zalivaka**, A. A. Ivaniuk, and C. H. Chang "Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response," *IEEE Trans. Inf. Forens. and Secur.*.

Conference Papers

[4] Y. Cao, **S. S. Zalivaka**, L. Zhang, C. H. Chang, and S. Chen, "CMOS image sensor based physical unclonable function for smart phone security applications," in *Proc. Int. Symp. on Integr. Circuits (ISIC'14)*, Marina Bay Sands, Singapore, pp. 392–395, 10-12 Dec. 2014.

[5] **S. S. Zalivaka**, A. V. Puchkov, V. P. Klybik, A. A. Ivaniuk, and C. H. Chang, "Multi-valued Arbiters for Quality Enhancement of PUF Responses on FPGA implementation," in *Proc. Asia and South Pacific Design Automat. Conf. (ASP-DAC'16)*, Macau, China, Jan. 2016, pp. 533–538 (Invited paper).

[6] **S. S. Zalivaka**, A. A. Ivaniuk, and C. H. Chang, "FPGA Implementation of Modeling Attack Resistant Arbiter PUF with Enhanced Reliability," in *Proc. IEEE Int. Symp. on Quality Electronics Design (ISQED'17)*, Santa Clara, USA, Mar. 2017, pp. 313–318 (Invited paper).

[7] **S. S. Zalivaka**, A. A. Ivaniuk, and C. H. Chang, "Low-cost fortification of arbiter PUF against modeling attack," in *Proc. IEEE Int. Symp. on Circ. and Syst. (ISCAS'17)*, Baltimore, USA, May. 2017, pp. 1600–1603.

Bibliography

[1] R. Insights. (2015, Oct.) FPGA market size, share, analysis, research report, 2020. [Online]. Available: http://www.radiantinsights.com/research/fpga-market

[2] I. Bolsens, "All Programmable SoC FPGA for networking and computing in big data infrastructure (Keynote I)," in *Proc. Asia and South Pacif. Design Automat. Conf. (ASP-DAC'14)*, Singapore, Jan. 2014, pp. 1–1, http://img.deusm.com/eetimes/zob-3pi-0023-01-lg.jpg.

[3] S. Sikand, "IP Reuse – design and verification report 2013," IC Manage, Tech. Rep., 2013.

[4] U. Government. (2003–2016) IPR annual seizure statistics. U.S. Customs and Border Protection. [Online]. Available: https://www.cbp.gov/trade/priority-issues/ipr/statistics

[5] R. Das, V. Markovich, J. McNamara, and M. Poliks, "Anti-tamper microchip package based on thermal nanofluids or fluids," USA Patent US20 120 068 326 A1, Mar. 22, 2012. [Online]. Available: http://www.google.com/patents/US20120068326

[6] R. S. Chakraborty and S. Bhunia, "RTL hardware IP protection using key-based control and data flow obfuscation," in *Proc. Int. Conf. on VLSI Design (VLSI'10)*, Bangalore, India, Jan. 2010, pp. 405–410.

[7] N. Couture and K. B. Kent, "Periodic licensing of FPGA based intellectual property," in *Proc. IEEE Int. Conf. on Field Progr. Techn. (ICFPT'06)*, Bangkok, Thailand, Dec. 2006, pp. 357–360.

[8] E. Peterson, "Developing tamper resistant designs with xilinx virtex-6 and 7 series FPGAs," Xilinx Inc., Tech. Rep., Oct. 2013.

[9] W. Liang *et al.*, "A chaotic IP watermarking in physical layout level based on FPGA," *Radioeng.*, vol. 20, no. 1, pp. 118–125, Apr. 2011.

[10] C. H. Chang and L. Zhang, "A blind dynamic fingerprinting technique for sequential circuit intellectual property protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 1, pp. 76–89, Jan. 2014.

[11] F. Koushanfar, "Hardware metering: A survey," in *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. New York, USA: Springer, 2011, pp. 103–122.

[12] P. Tuyls *et al.*, *Security with Noisy Data*, P. Tuyls *et al.*, Eds. New York, USA: Springer, 2007.

[13] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'07)*, Vienna, Austria, Sep. 2007, pp. 63–80.

[14] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy, "DRAM-based intrinsic physically unclonable functions for system-level security and authentication," *IEEE Trans. on Very Large Scale Integr. (VLSI) Syst.*, no. 99, pp. 1–13, 2016.

[15] S. S. Kumar *et al.*, "Extended abstract: The butterfly PUF protecting IP on every FPGA," in *Proc. IEEE Int. Worksh. on Hardw.-Orient. Secur. and Trust (HOST'08)*, Anaheim, USA, Jun. 2008, pp. 67–70.

[16] D. Yamamoto *et al.*, "Uniqueness enhancement of PUF responses based on the locations of random outputting RS latches," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'11)*, Nara, Japan, Sep. 2011, pp. 390–406.

[17] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA," in *Proc. Int. Symp. Reconfig. Comput.: Architect., Tools and Applic. (ARC'10)*, Berlin, Germany, Mar. 2010, pp. 382–387.

[18] T. Xu and M. Potkonjak, "Robust and flexible FPGA-based digital PUF," in *Proc. Int. Conf. on Field Programmable Logic and Applications (FPL'14)*, Munich, Germany, Sep. 2014, pp. 1–6.

[19] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. Design Autom. Conf (DAC'07)*, San Diego, USA, Jun. 2007, pp. 9–14.

[20] Q. Chen *et al.*, "The bistable ring puf: A new architecture for strong physical unclonable functions," in *Proc. IEEE Int. Sympos. on Hardw. Orient. Secur. and Trust (HOST'11)*, San Diego, USA, Jun. 2011, pp. 134–141.

[21] J. H. Anderson, "A PUF design for secure fpga-based embedded systems," in *Proc. Asia and South Pacific Design Automat. Conf. 2010 (ASP-DAC'10)*, Taipei, Taiwan, Jan. 2010, pp. 1–6.

[22] P. Grabher, D. Page, and M. Wojcik. (2013) On the (re)design of an FPGA-based PUF. [Online]. Available: https://eprint.iacr.org/2013/195.pdf

[23] V. Rozic *et al.*, "The monte carlo puf," in *Proc. Int. Conf. on Field Progr. Log. and Appl. (FPL'17)*, Ghent, Belgium, Sep. 2017, pp. 1–6.

[24] G. Prophet. (2016, Oct.) Xilinx to add PUF security to ZYNQ devices. [Online]. Available: http://www.eenewseurope.com/news/xilinx-add-puf-security-zynq-devices-0

[25] I. ID. (2015, Jun.) Altera reveals stratix 10 with intrinsic-ID's PUF technology. [Online]. Available: https://www.intrinsic-id.com/altera-reveals-stratix-10-with-intrinsic-ids-puf-technology/

[26] Microsemi. (2017, Jun.) Microsemi and intrinsic-ID collaboration delivers SRAM-PUF in PolarFire FPGAs providing advanced security. [Online]. Available: https://investor.microsemi.com/2017-06-01-Microsemi-and-Intrinsic-ID-Collaboration-Delivers-SRAM-PUF-in-PolarFire-FPGAs-Providing-Advanced-Security

[27] P. Dillien. (2017, Mar.) And the winner of best fpga of 2016 is. [Online]. Available: https://www.eetimes.com/author.asp?doc_id=1331443

[28] Xilinx. (2017) Xilinx ip cores. [Online]. Available: https://www.xilinx.com/products/intellectual-property.html

[29] Intel. (2017) Altera megacore. [Online]. Available: https://www.altera.com/support/support-resources/download/megacore-ip.html

[30] Microsemi. (2017) Microsemi directcore and companioncore. [Online]. Available: https://www.microsemi.com/products/fpga-soc/design-resources/ip-cores

[31] Gartner. (2015, Nov.) Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[32] Statista. (2016, Nov.) Internet of things (iot): number of connected devices worldwide from 2012 to 2020 (in billions). [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[33] A. Luigi, A. Lera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[34] X. Teng, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities," in *Proc. IEEE/ACM Int. Conf. on Comp.-Aided Design (ICCAD'14)*, San Jose, USA, Nov. 2014, pp. 417–423.

[35] H. Abie and I. Balasingham, "Risk-based adaptive security for smart iot in ehealth," in *Proc. Int. Conf. on Body Area Networks*, Oslo, Norway, Sep. 2012, pp. 269–275.

[36] A. Rupani and G. Sujediya, "A review of FPGA implementation of internet of things," *Int. J. of Innov. Resear. in Comp. and Comm. Eng.*, vol. 4, no. 9, pp. 16 203–16 207, Sep. 2016.

[37] Z. Zalewski, G. Manager, and H. Division. (2018) FPGAs accelerating IoT gateway and infrastructure tiers. [Online]. Available: https://www.aldec.com/en/company/blog/125–fpgas-accelerating-iot-gateway-and-infrastructure-tiers

[38] S2C. (2018) FPGA prototyping speeds design realization for the internet of things. [Online]. Available: http://www.s2cinc.com/technology-applications/internet-of-things-iot

[39] M. Alba. (2017, Aug.) FPGAs for the internet of things. [Online]. Available: https://www.engineering.com/IOT/ArticleID/15484/FPGAs-for-the-Internet-of-Things.aspx

[40] A. Rupani, P. Whig, G. Sujediya, and P. Vyas, "A robust technique for image processing based on interfacing of raspberry-pi and FPGA using IoT," in *Proc. Int. Conf. on Comp., Comm. and Electr. (Comptelix'17)*, Jaipur, India, Jul. 2017, pp. 350–353.

[41] S. Dhote *et al.*, "Using FPGA-SoC interface for low cost iot based image processing," in *Proc. Int. Conf. on Adv. in Comp., Comm. and Inform. (ICACCI'16)*, Jaipur, India, Sep. 2016, pp. 1963–1968.

[42] H. Q, O. Ayorinde, and B. H. Calhoun, "An ultra-low-power FPGA for IoT applications," in *Proc. IEEE SOI-3D-Subthr. Microel. Techn. Uni. Conf. (S3S'2017)*, Burlingame, USA, Oct. 2017, pp. 1–3.

[43] H.-Y. Kim and P. J. KIM, "Embedded systems of internet-of-things incorporating a cloud computing service of FPGA reconfiguration," USA Patent US20 160 321 081A1, 2016. [Online]. Available: https://patents.google.com/patent/US20160321081A1/en

[44] K. D. Krishna *et al.*, "Computer aided abnormality detection for kidney on FPGA based IoT enabled portable ultrasound imaging system," *Innov. and Res.in BioMed. eng. (IRBM)*, vol. 37, no. 4, pp. 189–197, Aug. 2016.

[45] T. Gomes *et al.*, "Towards an FPGA-based edge device for the internet of things," in *Proc. IEEE Conf. on Emerg. Techn. & Fact. Autom. (ETFA'15)*, Luxembourg, Luxembourg, Sep. 2015, pp. 1–4.

[46] M. Rao, T. Newe, and I. Grout, "Secure hash algorithm-3 (SHA-3) implementation on xilinx FPGAs, suitable for IoT applications," in *Proc. Int. Conf. on Sens. Technol. (ICST'14)*, Liverpool, UK, Sep. 2014, pp. 352–357.

[47] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-enabled secure architecture for FPGA-based IoT applications," *IEEE Trans. on Mult.-Scal, Comp. Syst.*, vol. 1, no. 2, pp. 110–122, Jun. 2015.

[48] J. Dorsch. (2017, Sep.) A chip for all seasons. [Online]. Available: https://semiengineering.com/a-chip-for-all-seasons/

[49] Intel. (2017, Feb.) A new intel FPGA for industrial and automotive markets. [Online]. Available: https://iot.do/new-intel-fpga-industrial-automotive-markets-2017-02

[50] P. Tanner. (2017, Mar.) How intel is leveraging FPGA technology to grow in IoT. [Online]. Available: https://marketrealist.com/2017/03/intel-leveraging-fpga-technology-grow-iot-market

[51] Z. C. Jouini, J.-L. Danger, and L. Bossuet. (2013) Evaluation of delay PUFs on CMOS 65 nm technology: ASIC vs FPGA. European Cooperation in Science and Technology. [Online]. Available: http://trudevice.com/Workshop/program/20TRUDEVICE_2013.pdf

[52] S. Walters, "Computer-aided prototyping for ASIC-based systems," *IEEE Design and Test of Comp.*, vol. 8, no. 2, pp. 4–10, 1991.

[53] S. Katzenbeisser *et al.*, "PUFs: Myth, fact or busted? a security evaluation of physically unclonable functions (PUFs). cast in silicon," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'12)*, Leuven, Belgium, Sep. 2012, pp. 283–301.

[54] RTC. (2012) Accelerating time-to-market using an FPGA and customizable SoC methodology. [Online]. Available: http://rtcmagazine.com/articles/view/102721

[55] J. Zhang, J. Cai, Y. Meng, and T. Meng, "Fault self-repair strategy based on evolvable hardware and reparation balance technology," *Chinese Journal of Aeronautics*, pp. 1211–1222, 2014.

[56] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *Proc. Design, Automat. and Test in Eur. Conf. (DATE'15)*, Grenoble, France, Mar. 653–658, pp. 653–658.

[57] Y. Cao *et al.*, "CMOS image sensor based physical unclonable function for coherent sensor-level authentication," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 62, no. 11, pp. 2629–2640, Oct. 2015.

[58] Z. Wang *et al.*, "Current mirror array: A novel circuit topology for combining physical unclonable function and machine learning," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 11, no. 99, pp. 1–13, Sep. 2017.

[59] Y. Cao, L. Zhang, C. H. Chang, and S. Chen, "A low-power hybrid RO PUF with improved thermal stability for lightweight applications," *IEEE Trans. on*

Comp.-Aided Design of Integr. Circ. and Syst., vol. 34, no. 7, pp. 1143–1147, Apr. 2015.

[60] J. Lee *et al.*, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Int. Symp. VLSI Circuits (VLSI'04)*, Honolulu, USA, Jun. 2004, pp. 176—-179.

[61] AnySilicon. (2016) FPGA vs ASIC, what to choose? [Online]. Available: http://anysilicon.com/fpga-vs-asic-choose/

[62] Xilinx. (2018, Apr.) Xilinx reports record annual and quarterly revenues. [Online]. Available: https://www.xilinx.com/news/press/2018/xilinx-reports-record-annual-and-quarterly-revenues.html

[63] A. Shilov. (2017, Feb.) Intel announces cyclone 10 FPGAs for iot devices. [Online]. Available: https://www.anandtech.com/show/11134/intel-announces-cyclone-10-fpgas-for-iot-devices

[64] EFY. (2015, Nov.) FPGA for internet of things. [Online]. Available: https://electronicsofthings.com/iot-projects/fundamentals-basics-start-101/fpga-internet-things/

[65] S. Morozov, A. Maiti, and P. Schaumont, "An analysis of delay based PUF implementations on FPGA," in *Proc. Int. Symp. Applied Reconfigurable Computing (ARC'10)*, Bangkok, Thailand, Mar. 2010, pp. 382–387.

[66] U. Ruhrmair *et al.*, "PUF modeling attacks on simulated and silicon data," *IEEE Trans. on Inf. Forens. and Secur.*, vol. 8, no. 11, pp. 1876–1891, Aug. 2013.

[67] A. Vijayakumar, V. C. Patil, C. B. Prado, and S. Kundu, "Machine learning resistant strong PUF: Possible or a pipe dream?" in *Proc. Hardw. Orient. Secur. and Trust (HOST'16)*, McLean, USA, May 2016, pp. 19–24.

[68] G. T. Becker, "On the pitfalls of using arbiter-PUFs as building blocks," *IEEE Trans. on Comp.-Aided Design of Integr. Circ. and Syst.*, vol. 34, no. 8, pp. 1295–1307, Apr. 2015.

[69] J. Delvaux. (2017, Dec.) Machine learning attacks on Poly-PUF, OB-PUF, RPUF, and PUF–FSM. [Online]. Available: https://eprint.iacr.org/2017/1134.pdf

[70] F. Koushanfar, "Integrated circuits metering for piracy protection and digital rights management: an overview," in *Proc. Great Lakes Symp. on VLSI (GLSVLSI '11)*, Lausanne, Switzerlan, May 2011, pp. 449–454.

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

[72] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. ACM Conf. Comput. and Comm. Secur. (CCS'02)*, Washington DC, USA, Nov. 2002, pp. 148–160.

[73] P. Tuyls *et al.*, "Read-proof hardware from protective coatings," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'06)*, Yokohama, Japan, Oct. 2006, pp. 369–383.

[74] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proc. Conf. RFID Secur. (RFID'07)*, Malaga, Spain, Jul. 2007, pp. 1–2.

[75] L. Zhang *et al.*, "Highly reliable memory-based physical unclonable function using spin-transfer torque MRAM," in *IEEE Int. Symp. on Circ. and Syst. (ISCAS'14)*, Melbourne, Australia, Jun. 2014, pp. 2169–2172.

[76] L. Zhang, Z. H. Kong *et al.*, "Exploiting process variations and programming sensitivity of phase change memory for reconfigurable physical unclonable functions," *IEEE Trans. on Inf. Foren. and Secur.*, vol. 6, no. 9, pp. 921–932, 2014.

[77] U. Ruhrmair *et al.*, "Modeling attacks on physical unclonable functions," in *Proc. ACM Conf. Comp. and Comms. Security (CCS'10)*, Chicago, USA, Oct. 2010, pp. 237–249.

[78] F. Koushanfar, G. Qu, and M. Potkonjak, "Intellectual property metering," in *Information Hiding*, ser. Lecture Notes in Computer Science, I. Moskowitz, Ed. Springer, 2001, vol. 2137, pp. 81–95.

[79] F. Koushanfar and G. Qu, "Hardware metering," in *Proc. IEEE Design Automation Conference (DAC'01)*, Scottsdale, USA, Jun. 2001, pp. 490–493.

[80] G. Qu and M. Potkonjak, "Fingerprinting intellectual property using constraint-addition," in *Proc. ACM Design Automat. Conf. (DAC'00)*, Los Angeles, USA, Jun. 2000, pp. 587–592.

[81] C. H. Chang and L. Zhang, "A blind dynamic fingerprinting technique for sequential circuit intellectual property protection," *IEEE Trans. Comput.-Aided Design of Integrated Circuits and Syst.*, vol. 33, no. 1, pp. 76–89, 2014.

[82] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proc. USENIX Secur. Symp. (USENIX'07)*, Boston, USA, Aug. 2007, pp. 291–306.

[83] F. Koushanfar, "Provably secure active IC metering techniques for piracy avoidance and digital rights management," *IEEE Trans. Inf. Forens. and Secur.*, vol. 7, no. 1, pp. 51–63, 2012.

[84] Y. Alkabani, F. Koushanfar, and M. Potkonjak, "Remote activation of ICs for piracy prevention and digital right management," in *Proc. IEEE Int. Conf. on Comp.-Aided Design (ICCAD'07)*, San Jose, USA, Nov. 2007, pp. 674–677.

[85] J. Roy, F. Koushanfar, and I. Markov, "Ending piracy of integrated circuits," *IEEE Computer*, vol. 43, no. 10, pp. 30–38, Oct. 2010.

[86] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs," in *Proc. Int. Conf. Reconfigurable Computing and FPGAs (ReConFig'10)*, Quantana Roo, Mexico, Dec. 2010, pp. 298–303.

[87] C. E. Shannon and W. Weaver, *A Mathematical Theory of Communication*. Chicago, USA: Univ of Illinois Press, 1963.

[88] A. Rukhin *et al.*, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards & Technology, Gaithersburg, USA, Tech. Rep., Apr. 2010.

[89] G. Marsaglia, "Diehard: A battery of tests of randomness," http://stat.fsu.edu/_geo, 1995.

[90] P. L'Ecuyer and R. Simard, "Testu01. a software library in ANSI C for empirical testing of random number generators," http://www.iro.umontreal.ca/ simardr/testu01/guideshorttestu01.pdf, 2013.

[91] R. G. Brown, "Dieharder: A random number test suite," http://webhome.phy.duke.edu/ rgb/General/dieharder.php, 2004.

[92] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms. A Classification Perspective*. Cambridge, UK: Cambridge University Press, 2011.

[93] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. on Computers*, vol. 11, no. 57, pp. 1198–1210, 2008.

[94] R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic PUFs from flip-flops on reconfigurable devices," in *Proc. Benelux Workshop on Information and System Security (WISSec'08)*, Eindhoven, The Netherlands, Nov. 2008, pp. 1–17.

[95] S. Zalivaka *et al.*, "Design and implementation of high-quality physical unclonable functions for hardware-oriented cryptography," in *Secure System Design and Trustable Computing*, C. Chang and M. Potkonjak, Eds. New York, USA: Springer, 2016, ch. 2, pp. 39–81.

[96] D. Ganta and L. Nazhandali, "Study of IC aging on ring oscillator physical unclonable functions," in *Proc. Int. Symp. on Qual. Electr. Design (ISQED'14)*, Santa Clara, USA, Mar. 2014, pp. 461–466.

[97] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," in *IEEE Int. Symp. on Inf. Theory (ISIT'09)*, Seoul, South Korea, Jul. 2009, pp. 2101–2105.

[98] R. Maes, A. V. Herrewege, and I. Verbauwhede, "PUFKY: A fully functional PUF-based cryptographic key generator," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'12)*, Leuven, Belgium, Sep. 2012, pp. 302–319.

[99] J.-L. Zhang *et al.*, "Techniques for design and implementation of an FPGA-specific physical unclonable function," *J. of comp. sci. and technology*, vol. 1, no. 31, pp. 124–136, Jan. 2015.

[100] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA PUF using programmable delay lines," in *Proc. IEEE Int. Workshop Info. Forens. Secur. (WIFS'10)*, Seattle, USA, Dec. 2010, pp. 1–6.

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

[102] M.-D. Yu, M. Hiller, and S. Devadas, "Maximum-likelihood decoding of device-specific multi-bit symbols for reliable key generation," in *Proc. Hardw. Orient. Secur. and Trust (HOST'15)*, Washington, USA, May 2015, pp. 38–43.

[103] V. P. Klybik and A. A. Ivaniuk, "Use of arbiter physical unclonable function to solve identification problem of digital devices," *Autom. Control Comput. Sci.*, vol. 49, no. 3, pp. 139–147, 2015.

[104] R. R. Sokal and C. D. Michener, "A statistical method for evaluating systematic relationships," *The Univ. of Kansas Sci. Bullutin*, vol. 38, no. 22, pp. 1409–1438, Mar. 1958.

[105] T. Kacprzak, "Analysis of oscillatory metastable operation of an RS flip-flop," *IEEE J. Solid State Circ.*, vol. 23, no. 1, pp. 260–266, 1988.

[106] U. Chatterjee, R. S. Chakraborty, H. Kapoor, and D. Mukhopadhyay, "Theory and application of delay constraints in arbiter PUF," *ACM Trans. on Embed. Comp. Syst.*, vol. 15, no. 1, pp. 1–20, Jan. 2016.

[107] E. Ozturk, G. Hammouri, and B. Sunar, "Physical unclonable function with tristate buffers," in *Proc. IEEE Int. Symp. on Circ. and Syst. (ISCAS'08)*, Seattle, USA, May 2008, pp. 3194–3197.

[108] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proc. IEEE Int. Test Conf. (ITC'08)*, Santa Clara, USA, Oct. 2008, pp. 1–10.

[109] Xilinx, "Vivado design suite," http://www.xilinx.com/products/design-tools/vivado.html, Jun. 2016.

[110] S. S. Zalivaka *et al.*, "Multi-valued arbiters for quality enhancement of PUF responses on FPGA implementation," in *Proc. Asia and South Pacific Design Automat. Conf. (ASP-DAC'16)*, Macau, China, Jan. 2016, pp. 533–538.

[111] Xilinx, "Xilinx zynq-7000 all programmable SoC ZC706 evaluation kit," https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html, Jan. 2015.

[112] Digilent, "Nexys 4 DDR artix-7 FPGA: Trainer board recommended for ECE curriculum," http://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/, Dec. 2014.

[113] S. S. Zalivaka, A. A. Ivaniuk, and C. H. Chang, "FPGA implementation of modeling attack resistant arbiter PUF with enhanced reliability," in *Proc. IEEE Int. Symp. on Quality Electronics Design (ISQED'17)*, Santa Clara, USA, Mar. 2017, pp. 313–318.

[114] M. Bhargava and K. Mai, "An efficient reliable PUF-based cryptographic key generator in 65nm CMOS," in *Proc. Design, Autom. and Test in Europe Conf. (DATE'14)*, Dresden, Germany, Mar. 2014, pp. 1–6.

[115] Thermotron, "8800 controller," http://thermotron.com/8800-controller.html, Aug. 2007.

[116] A. Shrivastava *et al.*, "Design of a reliable RRAM-based PUF for compact hardware security primitives," in *IEEE Int. Symp. on Circ. and Syst. (ISCAS'16)*, Montreal, Canada, May 2016, pp. 2326–2329.

[117] A. Kholosha, "Clock-controlled shift registers and generalized geffe key-stream generator," in *Int. Conf. on Crypt. in India (INDOCRYPT'01)*, Chennai, India, Dec. 2001, pp. 287–296.

[118] E. Dubrova, M. Teslenko, and H. Tenhunen, "On analysis and synthesis of (n,k)-non-linear feedback shift registers," in *Proc. Design, Automat. and Test in Europe (DATE '08)*, Munich, Germany, Mar. 2008, pp. 1286–1291.

[119] R. Wicik, R. Gliwa, and P. Komorowski, "Cryptanalysis of alternating step generators," in *Proc. Int. Conf. on Milit. Communic. and Inf. Syst. (ICMCIS'15)*, Gdansk, Poland, May 2015, pp. 1–6.

[120] T. Machida, D. Yamamoto, M. Iwamoto, and K. Sakiyama, "Implementation of double arbiter PUF and its performance evaluation on FPGA," in *Proc. Asia and South Pacific Design Automat. Conf. (ASP-DAC'15)*, Chiba/Tokyo, Japan, Jan. 2015, pp. 6–7.

[121] Y. Lao and K. K. Parhi, "Statistical analysis of MUX-based physical unclonable functions," *IEEE Trans. on Comp.-Aided Design of Int. Circ. and Syst.*, vol. 33, no. 5, pp. 649–662, 2014.

[122] A. Vijayakumar and S. Kundu, "A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics," in *Proc. Des., Autom. and Test in Europe Conf. (DATE'15)*, Grenoble, France, Mar. 2015, pp. 653–658.

[123] R. Kumar and W. Burleson, "On design of a highly secure PUF based on non-linear current mirrors," in *Proc. Hardw. Orient. Secur. and Trust (HOST'14)*, Arlington, USA, May 2014, pp. 38–43.

[124] M. Kalyanaraman and M. Orshansky, "Novel strong PUF based on nonlinearity of MOSFET subthreshold operation," in *Proc. Hardw. Orient. Secur. and Trust (HOST'13)*, Austin, USA, Jun. 2013, pp. 13–18.

[125] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. ACM Annual Comp. Secur. Appl. Conf. (ACSAC'02)*, Washington DC, USA, Dec. 2002, pp. 149–161.

[126] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and reliable xor arbiter puf design: An experimental study based on 1 trillion challenge response pair measurements," in *Proc. Design Autom. Conf. (DAC'17)*, Austin, USA, Jun. 2017, pp. 101–106.

[127] Y. Gao *et al.*, "Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices," in *Proc. IEEE Int. Worksh. on Secur., Priv. and Trust for IoT*, Sydney, Australia, Mar. 2016, pp. 1–6.

[128] M. Rostami *et al.*, "Robust and reverse-engineering resilient puf authentication and key-exchange by substring matching," *IEEE Trans. on Emerg. Topics in Comput.*, vol. 2, no. 1, pp. 37–49, Jan. 2014.

[129] V. N. Yarmolik, S. Hellebrand, and H.-J. Wunderlich, "Self-adjusting output data compression: An efficient BIST technique for RAMs," in *Proc. Design, Automat. and Test in Eur. Conf. (DATE'98)*, Paris, France, Feb. 1998, pp. 173–179.

[130] V. Agrawal, C. Kime, and K. Saluja, "A tutorial on built-in self-test. 2. applications," *IEEE Trans. on Electr. Design and Automat.*, vol. 10, no. 2, pp. 69–77, Jun. 1993.

[131] S. S. Zalivaka, A. A. Ivaniuk, and C. H. Chang, "Low-cost fortification of arbiter PUF against modeling attack," in *Proc. IEEE Int. Symp. on Circ. and Syst. (ISCAS'17)*, Baltimore, USA, May 2017, pp. 1600–1603.

[132] H. C. A. van Tilborg and S. Jajodia, *Encyclopedia of Cryptography and Security*. Springer, 2011. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4419-5906-5_487

[133] A. Mahmoud, U. Ruhrmair, M. Majzoobi, and F. Koushanfar. (2013) Combined modeling and side channel attacks on strong PUFs. [Online]. Available: https://eprint.iacr.org/2013/632.pdf

[134] S. Burman, D. Mukhopadhyay, and K. Veezhinathan, "LFSR based stream ciphers are vulnerable to power attacks," in *Int. Conf. on Cryptol. in India (Indocrypt'07)*, Chennai, India, Dec. 2007, pp. 384–392.

[135] S. T. C. Konigsmark, D. Chen, and M. D. F. Wong, "PolyPUF: Physically secure self-divergence," *IEEE Trans. on Comp.-Aided Design of Integr. Circ. and Syst.*, vol. 35, no. 7, pp. 1053–1066, Oct. 2015.

[136] Y. Gao *et al.*, "PUF-FSM: A controlled strong PUF," *IEEE Trans. on Comp.-Aided Design of Integr. Circ. and Syst.*, vol. 1, no. 99, pp. 1–5, 2017.

[137] J. Ye, Y. Hu, and X. Li, "RPUF: Physical unclonable function with randomized challenge to resist modeling attack," in *Proc. IEEE Asian Hardw.-Orient. Secur. and Trust (AsianHOST'16)*, Yilan, Taiwan, Dec. 2016, pp. 1–6.

[138] J. Zhang and L. Wan. (2018, Jun.) CMOS: Dynamic multi-key obfuscation structure for strong PUFs. Cornell University Library. Ithaca, USA. [Online]. Available: https://arxiv.org/pdf/1806.02011.pdf

[139] C. Herder *et al.*, "Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions," *IEEE Trans. on Depend. and Secur. Comput.*, vol. 14, no. 1, pp. 65–82, Mar. 2016.

[140] Z. Shi, C. Ma, J. Cote, and B. Wang, *Introduction to Hardware Security and Trust*. Springer, 2012, ch. Hardware Implementation of Hash Functions, pp. 27–50.

[141] P. Chodowiec and K. Gaj, "Very compact FPGA implementation of the AES algorithm," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'03)*, Cologne, Germany, Sep. 2003, pp. 319–333.

[142] T. Good and M. Benaissa, "AES on FPGA from the fastest to the smallest," in *Proc. Int. Worksh. Crypt. Hardw. and Emb. Syst. (CHES'05)*, Edinburg, UK, Sep. 2005, pp. 427–440.