CHAPTER

HARDWARE SECURITY PRIMITIVES

12

CONTENTS

12.1 Introduction 312
12.2 Preliminaries 313
12.2.1 Common Hardware Security Primitives 313
12.2.2 Performance of a CMOS Device 313
12.2.3 Performance and Reliability vs. Security 315
12.3 Physical Unclonable Function 316
12.3.1 PUF Preliminaries 316
12.3.2 PUF Classifications 317
12.3.3 PUF Quality Properties 317
12.3.4 Common PUF Architectures 318
12.3.4.1 Arbiter PUF 318
12.3.4.2 Ring Oscillator (RO) PUF 319
12.3.4.3 SRAM-PUF 320
12.3.4.4 Butterfly PUF 320
12.3.4.5 Lightweight PUF 321
12.3.4.6 Bistable Ring PUF 321
12.3.5 PUF Applications 322
12.3.5.1 Secret Key Generation 322
12.3.5.2 Device Authentication 323
12.3.5.3 Software and IP Licensing 324
12.3.5.4 Hardware Metering 324
12.4 True Random Number Generator 324
12.4.1 TRNG Preliminaries 324
12.4.2 TRNG Quality Properties 325
12.4.3 Common TRNG Architectures 325
12.4.3.1 Noise-Based TRNGs 326
12.4.3.2 Oscillator-Based TRNGs 327
12.4.3.3 Memory-Based TRNGs 328
12.4.4 TRNG Applications 328
12.5 Design for Anti-Counterfeit 329
12.5.1 DfAC Preliminaries 329
12.5.2 DfAC Designs 329
12.6 Existing Challenges and Attacks 331
12.6.1 PUFs 331
12.6.1.1 Modeling Attacks Against Strong-PUFs 331
12.6.1.2 Environmental and Aging Impacts on PUF 332
12.6.1.3 Cloning SRAM-PUF 334

Hardware Security. https://doi.org/10.1016/B978-0-12-812477-2.00017-4

12.6.2 TRNGs 335
12.7 Primitive Designs With Emerging Nanodevices 336
12.7.1 Composition of PCM-Based PUFs 336
12.7.1.1 Sources of Process-Induced Variability 336
12.7.1.2 PCM-PUF Designs 337
12.7.2 Composition of Memristor and RRAM-Based PUFs 337
12.7.2.1 Sources of Process-Induced Variability 337
12.7.2.2 Memristor and RRAM-PUF Designs 337
12.7.3 Composition of MRAM and STTRAM-Based PUFs 338
12.7.3.1 Sources of Process-Induced Variability 338
12.7.3.2 MRAM and STT-RAM PUF Designs 338
12.7.4 PUF Composition for Emerging Applications 338
12.8 Hands-on Experiment: Hardware Security Primitives (PUFs and TRNGs) 339
12.8.1 Objective 339
12.8.2 Method 339
12.8.3 Learning Outcome 340
12.8.4 Advanced Options 340
12.9 Exercises 340
12.9.1 True/False Questions 340
12.9.2 Long-Answer Type Questions 340
References 343

12.1 INTRODUCTION

The current-world computer system security issues do not encompass only software and information threats or vulnerabilities of the early 2000s. Rather, hardware-oriented security has become an increasing concern due to growing threat and attack complexity. The advent of novel security threats, such as hardware Trojans, counterfeit electronic products, and various physical attacks have nullified the underlying notion of hardware as the root of trust. Furthermore, low-cost and resource-constrained IoT, mobile, and embedded devices now require secure and reliable hardware platforms more than ever for trustworthy communication, privacy protection, defense against numerous software or hardware threats, and vulnerabilities.

In this regard, hardware security primitives and designs play an important role to ensure trust, integrity, and authenticity of electronic chips and systems. Such primitives can work as the hardware-level building blocks to develop a secure platform. Among common hardware security primitives, physical unclonable functions (PUFs) and true random number generators (TRNGs) are most notable for utilizing device-intrinsic process variations and noise to extract entropy [1–3]. These primitives can be used for generating cryptographic keys and IDs to authenticate devices and systems, to produce session keys, nonce, and many more. These primitives, acting as a hardware alternative to key storage, digital fingerprint, or software-generated bitstreams can provide defense against prevailing adversarial threats, such as spoofing and cloning. In addition, researchers have proposed designs for countermeasures, that can defend against IC counterfeiting, tampering, and reverse engineering, by utilizing a different set of device-intrinsic properties. For example, combating die and IC recycling (CDIR) sensor leverages aging and wear-out mechanisms in common CMOS-devices to offer countermeasures against IC counterfeiting (recycling) [4]. Nevertheless, with the rise of emerging threats and

vulnerabilities, and longstanding attacks becoming more practical, designers constantly seek for novel primitives and countermeasures that utilize the device's inherent properties to enhance security.

12.2 PRELIMINARIES

12.2.1 COMMON HARDWARE SECURITY PRIMITIVES

Hardware security primitives, such as PUFs and TRNGs, and countermeasures, including design for anti-counterfeit (DfAC) offer safeguards to various potential threats and vulnerabilities arising at different phases of the IC lifecycle and device operation. PUFs generate device-specific digital output that is tied to the intrinsic properties of the device. Hence, a PUF-generated signature can be considered as a digital fingerprint. This fingerprint is usually produced by accumulating inherent minute process variations from the manufacturing steps. Since such process variations are random and static in nature, the generated fingerprint is ideally unclonable from device-to-device, and unpredictable to the adversary. This digital fingerprint can be used for cryptographic key generation, device authentication, and preventing cloning.

TRNGs, on the other hand, generate random digital bitstreams that do not have any predictability whatsoever. They exploit the random transient variations in the system, such as power supply fluctuations, and device intrinsic noise. The source of entropy [5] of a TRNG is, therefore, ideally different from that of a PUF. The random bit-stream generated by a TRNG can be used as assets such as session keys and nonce.

DfACs, such as a CDIR module, provide certain signatures by monitoring the lifetime of a chip. In most cases, a recycled IC experience ample amount of prior usage before getting back into the supply chain as a counterfeit product. Electrical components, such as CMOS transistors, tend to deviate from their ideal characteristics, for example, speed and power consumptions, due to aging. A CDIR module can, therefore, monitor such characteristics deviations and perceive prior usage, if any.

12.2.2 PERFORMANCE OF A CMOS DEVICE

Even after several decades, CMOS-based devices still dominate the semiconductor industry due to improved lithographic techniques, ease of fabrication, and high yield with respect to manufacturing cost. The road to advanced nodes has produced multiple architectures: from regular planar bulk-CMOS devices to high-k/metal-gate transistors and tri-gate/FinFET transistors. All these devices are traditionally intended to offer high-performance with smaller size, higher speed, lower leakage, and provide better reliability. However, with the technology nodes getting more advanced with smaller feature sizes, the semiconductor industry faces major manufacturing and reliability issues. In particular, CMOS devices are now experiencing greater process variations and more aggressive performance degradation due to aging and runtime variations [6]. Figure 12.1 lists some key physical parameters and runtime/reliability factors that can have drastic negative effects on IC performance and reliability [7]. Nevertheless, many such properties and phenomena play a key role in devising the aforementioned hardware security primitives (as shown in the right-hand columns of Fig. 12.1; discussed further in Section 12.2.3).

- *Process Variations:* CMOS device manufacturing process has numerous sources of systematic and random variations that play a critical role in yield and performance. CMOS front-end of the line

CMOS Bulk, HK+MG and FinFET Device Properties for Security Applications

Phenomena		Electrical Manifestation (Performance)	Security Applications	
Process Variations	• Geometric Variation (Patterning – W, L) • Random Dopant Fluctuation (RDF) • Line Edge Roughness (LER) • Oxide Thickness (T_{ox}) Fluctuation • Interface defect and traps (ITC) • Polysilicon/Metal Gate Granularity (MGG)	• Threshold Voltage Deviation (ΔV_{TH}) • Carrier Mobility Degradation ($\Delta\mu_n$) • Drain Current Variation (ΔI_{ON}) • Off-state Leakage Current Variation (ΔI_{Leak}) • Drain Induced Barrier Lowering (DIBL)	PUF	• Arbiter • RO • Leakage Current • Bistable Ring • Hybrid Delay/ Cross-coupled PUF
Aging & Wear-out Mechanisms	• Bias Temperature Instability (NBTI/PBTI) • Hot Carrier Injection (HCI) • Time Dependent Dielectric Breakdown (TDDB) • Electromigration (EM)		TRNG	• Thermal/Power Supply Noise • Clock Jitter • Metastability • Oxide soft-breakdown
Runtime Variations	• Power Supply Noise • Temperature Variation		DfAC	• Recycling – Aging (CDIR) • Cloning – Process Variation

FIGURE 12.1

Potential use of inherent device properties for security applications.

(FEOL) variation sources are most notably patterning (proximity) effects, line-edge roughness (LER), nonuniform n/p-type doping, and gate dielectrics, such as oxide thickness variations, defects, and traps. Variations due to random dopant fluctuations and gate material granularity (poly-si or metal gate) are also becoming significant in advanced nodes [6]. These phenomena directly impact the electrostatic integrity of the channel and affect the strength of the device. Back-end of the line (BEOL) sources, such as metal interconnect and dielectric variations, also have a substantial impact [8]. All such variations holistically cause deviation in device characteristics, making every single transistor slightly different from each other with a shift from the nominal performance. Such manufacturing process variations are undesirable for performance-oriented logic and memory applications, yet unavoidable, and often prominent in the advanced nodes.

- *Runtime/environmental variations:* Similar to manufacturing process variations, various runtime/environmental variations, such as temperature fluctuation and power supply noise, also have a direct impact on electrical characteristics of a transistor. For instance, an increase in the operating temperature decreases both carrier mobility (μ) and threshold voltage (V_{th}) of a traditional transistor, and thus impacts the speed (delay) of the device, since they cause an opposing impact on the drain saturation current (I_{DS}) and leakage current (I_{Leak}). However, technology nodes also play a crucial role in the performance impact. For example, technology nodes from 45 nm and below can show an increase in device speed with temperature, whereas it is the opposite in older technologies [9]. In addition, global and local power supply noise has an adverse impact on performance, since such variations also cause a shift in V_{th} and I_{DS} from the nominal value. Hence, for both cases, a system is less robust and prone to produce erroneous results. However, unlike a permanent shift in performance resulting from the manufacturing process variation or aging, a change in the environmental condition usually causes a temporary shift, given that the experienced variations have not

caused a permanent damage or wear out. Therefore, it is compensated once the device returns to its nominal operating state.

- *Aging and wear-out mechanisms:* Aging degradation and wear-out mechanisms, such as bias temperature instability (BTI), hot carrier injection (HCI), electromigration (EM), and time-dependent dielectric breakdown (TDDB), lead to a poor device and circuit performance. The magnitude of such degradation largely depends on the device workload, active bias, inherent random defects, and technology nodes under consideration. BTI and HCI are considered key aging mechanisms that directly impact the speed of CMOS devices. BTI slows down transistors as traps are generated at SiO_2/Si interface, resulting in an increase in threshold voltage magnitude ($|V_{th}|$) over time. Typically negative BTI (NBTI) occurring in PMOS transistors is dominant compared to positive BTI (PBTI) occurring in NMOS transistors beyond 65 nm technology nodes. However, the latter is becoming prominent for high-k metal-gate devices [10]. Additionally, HCI slows down a device by generating charged defects in gate oxide, interfacing to increase V_{th}, and by reducing the mobility of a device. HCI is more prominent in NMOS with smaller feature size. However, HCI-induced degradation can be recovered to a certain limit and is affected by voltage, temperature, and device workload [11]. Both of these mechanisms greatly decrease reliability and, eventually, shorten chip-lifetime with increased failure rate. Although such mechanisms are quite slow and the degradation magnitude is relatively hard to precisely predict, since it is statistical in nature, an accelerated aging (that is, running the chip at a higher voltage and/or temperature than the nominal operating condition) allows us to determine the probable impact on the IC performance and lifetime in most cases. This also helps to deploy any compensating mechanisms.

12.2.3 PERFORMANCE AND RELIABILITY VS. SECURITY

It is evident that the CMOS device performance deviates due to process variation, environmental condition, and aging. Therefore, minimizing process variations and other degradation phenomena is of utmost importance for high-performance circuits. However, one can also see that these variations and degradation mechanisms do not necessarily have adverse impacts on hardware security primitives and applications. In fact, some of these variations and degradation mechanisms can be effectively leveraged for ensuring hardware-based security (see Fig. 12.1). For example, PUFs rely on the manufacturing process variations. An increase in physical variations can potentially improve the PUF outcome quality, although high manufacturing variation is undesirable for high performance and yield. In addition, the detection process of some types of counterfeit electronics can benefit from inherent aging and wear-out mechanisms; the signs of prior use can potentially lead to the detection of recycled chips.

Note that, not every phenomenon is always beneficial to all the security applications either. Let us use the terms – *good*, *bad*, and *ugly* – to qualitatively state the relationship between process variations, reliability degradation, and various hardware-based security mechanisms, as shown in Table 12.1. The first column of Table 12.1 shows conventional and security-based applications and primitives, and respective rows refer to how process variation, temperature, power supply noise, aging, and wear-out mechanisms affect the quality of operation. Here, the good indicates that variation or degradation mechanism is actually desirable and beneficial in relation to a security application or primitive; the bad means that it should be avoided if possible, and the ugly means that it is highly undesirable for a reliable operation. For example, in trivial logic/memory applications, manufacturing process variation is highly undesirable (ugly) to ensure better performance; whereas it is one of the key requirements (that

Table 12.1 Design and technology characteristics vs. security trade-off

Application/primitive	Process variation	Temperature	Power supply noise	Aging (BTI/HCI)	Wear-out (EM)
Logic/memory design	Ugly	Bad	Bad	Bad	Bad
PUF	Good	Bad	Bad	Bad	Bad
TRNG	Good	Bad	Good	Bad	Bad
Recycled IC detection	Ugly	Bad	Bad	Good	Good

is, good) for PUF and TRNG applications. Aging and wear-out mechanisms are bad for both regular logic/memory applications and PUFs. However, they can be leveraged (that is, *good*) for detecting recycled electronics [12].

It is, therefore, vital that one establishes proper correlation status among performance, reliability, and security. To achieve this, rather than building security primitives solely relying on trivial logic/memory application-oriented designs, one should make a balance of performance, reliability, and security to obtain the best trade-off depending on the target application.

12.3 PHYSICAL UNCLONABLE FUNCTION

12.3.1 PUF PRELIMINARIES

As indicated in Section 12.2.1, a PUF generates digital fingerprints using device-intrinsic characteristics. Ideally, it is a cryptographically secure one-way function that generates a digital output (response) for a given input (challenge) without revealing any predictable mapping between the challenge and response [1,2,13]. As the name suggests, a PUF can generate digital output (IDs or keys) by leveraging inherent physical variations from the manufacturing process. Therefore, identical (by design and lithography/mask) integrated circuits manufactured by the same fabrication facility and process can generate different challenge-response pairs (or cryptographic keys), as there always exists small, but nondeterministic, variations in the manufacturing process. Generally, the input or challenges to the PUF excite certain electrical characteristics, such as delay or leakage current, variation to extract maximum entropy.

This is a crucial advancement over the conventional nonvolatile memory-based key-storage mechanisms. In traditional methods, secret keys are stored digitally in a nonvolatile memory (NVM), such as flash memory or electrically erasable programmable read-only memory (EEPROM), which is always vulnerable, based on hardware implementation, key-propagation mechanisms, and physical attacks. The NVM that stores the cryptographic keys can be subjected to tampering, probing, and imaging attacks. Therefore, it must be protected by the physical layer of security, in addition to protocol-level protections. Since a PUF can produce a cryptographic key or digital fingerprint which, in addition, is unique from device to device, it eliminates the requirement to "store" the key in a memory, as it can be generated on demand via input triggers. There is no need to program the secret, and it can generate multiple master keys by changing associated challenge sets. Also, any additional physical attack on the PUF, such as probing, impacts the inherent characteristic and drastically change the PUF's response. Therefore, PUFs offer an attractive volatile and tamper-resistant alternative to conventional cryptographic key-storage techniques [14,15].

12.3.2 PUF CLASSIFICATIONS

Based on the underlying challenge-response pair (CRP)-space, PUFs can be broadly categorized into *weak* and *strong* PUFs. A weak PUF, also known as a *physically obfuscated key* (POK), typically can be interrogated with a very limited number of challenges [16,17]. Therefore, its CRP-space is extremely small, often even only one. Such a PUF can be used for generating cryptographic keys. An SRAM-PUF (to be discussed in Section 12.3.4) is a notable example of this type.

In contrast, a strong PUF can accommodate a very large number of challenges to produce corresponding responses [18]. Ideally, the CRP-space grows exponentially with the length of challenge itself. This allows the PUF to undergo multiple queries and use a new challenge-set every time to avoid any collision or replay attacks. The arbiter-PUF and ring-oscillator PUF (to be discussed in Section 12.3.4) are prominent examples of strong PUFs.

One can also bootstrap a strong PUF to external logic and algorithms for providing secure and controlled access to the PUF via an application programming interface (API). Such a PUF design, known as a controlled-PUF, is very much application-oriented and can offer additional security against spoofing and modeling attacks, where the adversary tries to fool the system using a predictive model-generated response (to be further discussed in Section 12.6.1.1) [14].

12.3.3 PUF QUALITY PROPERTIES

In general, an ideal PUF should only exploit physical properties of a device, such as process variation, to generate the response and not rely on stored data. The unclonability indicates that it cannot be replaced with a software model to replicate the PUF outcome, and must be prohibitively difficult to physically duplicate. To make it unclonable, the design should utilize small and random intrinsic variations, so that the exact response cannot be predicted by the adversary and the challenges undergo a one-way transformation (response). Additionally, the PUF itself should be inexpensive to fabricate for a wide variety of low-cost applications, and be intrinsically attack-resilient, that is, must not generate a deterministic response due to external adversarial control.

Although, most of the PUFs proposed in literature possess trivial properties, the response they produce are not necessarily ideal for targeted applications, for example, key generations and authentication. The most popular quality metrics to evaluate PUF responses include uniqueness, randomness (or uniformity), and reproducibility (or reliability). The qualitative assessment of PUFs is important, as a poor PUF may lead to error in cryptographic applications and authentication protocols, and it may be prone to different modeling and machine learning attacks [15,19].

Uniqueness measures the distinctive challenge-response pair (CRP) generation quality, that is, distinguishability, of a PUF with respect to other instances. This is the very first step of quality assessment for both weak and strong PUFs. A common measurement of uniqueness is the inter-PUF hamming distance (inter-HD), calculated over multiple PUFs, given as [19]

$$HD_{\text{inter}} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{HD(R_i, R_j)}{k} \times 100\%, \tag{12.1}$$

where n stands for the total number of PUFs under assessment, k is the response length, and $HD(R_i, R_j)$ is the hamming distance between the response R_i from PUF_i and the response R_j from

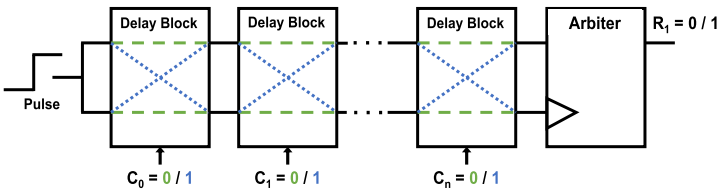

FIGURE 12.2

Standard structure of arbiter PUF.

PUF_j (where $i \neq j$) from the testing pool. An inter-PUF HD of 50% produces the ideal uniqueness, that is, it provides a maximum difference in the response bits between any two PUFs.

Randomness or uniformity stands for the unpredictability of a PUF. Usually, strong PUFs show if there is any measurable trend in the generated response, since an ideal PUF should be free from bias and correlation. In order to assess the randomness of the produced responses from multiple challenges, statistical test suites, such as the NIST test suite [20], DieHARD [21] are commonly used. Additionally, a good diffusive property is expected for strong PUFs, that is, a small change in the input challenge should produce a large variation in the response (also known as the avalanche effect).

Reproducibility or reliability assesses the PUF's quality in terms of its capability to generate same CRPs across different environmental conditions, and over time. This is traditionally measured via intra-PUF hamming distance given as [19]

$$HD_{\text{intra}} = \frac{1}{m}\sum_{y=1}^{m}\frac{HD(R_i, R'_{i,y})}{k} \times 100\%, \tag{12.2}$$

where m stands for the number of samples or run for a PUF, k is the response length from the signature generated by the PUF, and $HD(R_i, R'_{i,y})$ is the hamming distance between the response R_i and the yth sampling $R'_{i,y}$ of the PUF under test. Ideally, a PUF should always maintain the same challenge-response pairs (CRPs) over different operating conditions and/or times resulting into zero-bit error rate (i.e. 0% intra-PUF HD).

12.3.4 COMMON PUF ARCHITECTURES

12.3.4.1 Arbiter PUF

Arbiter-PUF is one of the most notable CMOS logic-based PUF architecture that exploits the randomness of path delay due to uncontrollable process variation [2]. Figure 12.2 shows the generic design of an arbiter-PUF. Each of the building blocks is an individual delay unit with path-switching capability controlled by the challenge bit (denoted by c_i). Given a pulse at the input of a delay stage, it can traverse through two design-wise identical, but different paths (selected by challenge) and reach the final arbiter (or decision-making) component. If the signal in the upper path reaches the arbiter first, it generates "1" (and vice versa). Ideally, in the absence of any manufacturing process variation, the delay through both the paths would be the same and the signal would reach the arbiter at the exact same time. However, there always exists some process variation induced delay difference between these two

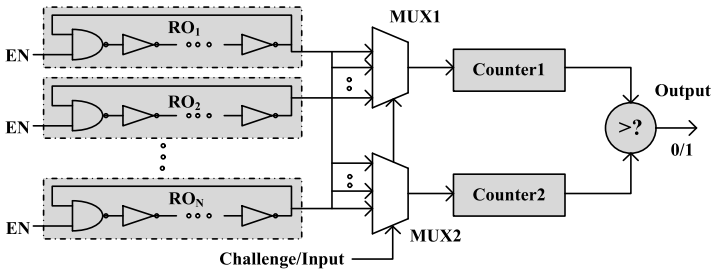

FIGURE 12.3

Conventional RO-PUF.

identical paths, and one of the signals reach the arbiter faster than the other. Since, the paths do not have any systematic or extrinsic delay difference, the shortest/longest path is not deterministic, and only depends on the individual transistor strength and interconnects. As shown in Section 12.2.2, any random deviation in physical or electrical properties would cause this nondeterministic variation. As an increase in the number of delay stages (cascaded one after another in series) exponentially increases possible path-pairs, the arbiter-PUF is capable of generating a large number of CRPs (strong PUF). Another advantage of arbiter PUF is that it takes only one cycle to generate a 1-bit response, although the number of delay stages makes the path (delay) longer.

However, the arbiter PUF has some major drawbacks, one of them being the bias induced at the arbiter itself due to finite delay-difference resolution for the setup, and hold times. Also, this requires symmetric design and routing, which may not be readily available for lightweight and FPGA-applications. Additionally, the arbiter-PUF has been shown to be vulnerable to modeling attack, since it can be represented as a linear delay model.

12.3.4.2 Ring Oscillator (RO) PUF

The schematic of a typical ring-oscillator-based PUF (RO-PUF) is shown in Fig. 12.3. It does not require rigorous design and can be easily implemented in both ASIC and reconfigurable platforms, such as FPGAs [22]. An RO-PUF is generally composed of N identical ring oscillators (ROs), two multiplexers, two counters, and one comparator. When enabled, each of the ROs oscillates at a slightly different frequency from one another due to process variations. A challenge is applied to select one pair of ROs, and the number of oscillations of each of the ROs in the selected pair is counted and compared to generate a "0" or "1', based on which oscillator from the selected RO pair is faster.

Compared to an arbiter PUF, the RO-PUF is larger and slower for generating the same number of response bits. Whereas an arbiter PUF can generate the response bits in one system clock, RO-PUF requires a significant number of counts of the oscillatory signals to obtain a reliable value. The oscillatory switching of the components makes it power-hungry. Since all the components of RO go through significant usage, it suffers from runtime power and temperature variations and aging. This makes the RO-PUF prone to generating erroneous output.

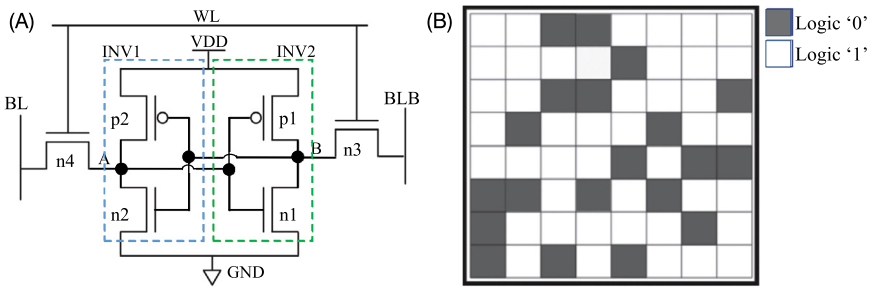

FIGURE 12.4

(A) Typical 6T SRAM cell. (B) Startup fingerprint of an example SRAM array.

12.3.4.3 SRAM-PUF

In contrast to custom-designed arbiter- and RO-PUFs, a static random-access memory (SRAM)-based PUF utilizes the widely available SRAM-matrix used in microprocessors, microcontrollers, field-programmable gate arrays (FPGAs) and in standalone chips for embedded systems. Typically, one SRAM bit is implemented by a symmetrically designed 6-transistor cell, as shown in Fig. 12.4A, where either one of the nodes *A* or *B* is pulled high, and the other is pulled low in the stable state once the cell is programmed (written). Due to the feedback provided by the cross-coupled inverter structure, the stable state holds true until the cell is rewritten or the system power is off.

In contrast, during the startup in absence of any "write"/programming command, both the logic nodes tend to pull up to high voltage. However, only one wins via racing condition to reach high voltage (logic 1), and automatically pulls the other node down to low voltage (logic 0). This initialization typically depends on the minute process-induced strength mismatch among the cell transistors (especially between the two pull-up PMOS-transistors), and is completely unpredictable to an external observer. The start-up outcome, being strongly tied to the physical process variation, is static and tends to produce the same outcome for a given cell over multiple power-ups. Therefore, the startup of the SRAM cell can be utilized as a weak PUF for device-intrinsic fingerprint generation [17,23].

It should be noted that the process variation in the SRAM cell may be small enough to be overcome by environmental noise and, therefore, not all cells can produce a reliable response over time and usages. For example, Fig. 12.4B shows the initial startup values of some cells in the SRAM matrix. Among them, only the cells that show the highest reproducibility should be selected to generate the PUF signature. Additionally, advanced SRAM-inclusive commercial off-the-shelf (CoTs) products may not readily provide suitable PUF application due to various initialization and memory-access processes. For example, in some recent Altera and Xilinx FPGA models, RAM blocks are always initialized to certain logic at startup and, therefore, cannot be used to implement the SRAM-PUF based on a random initialization [24].

12.3.4.4 Butterfly PUF

The design of Butterfly PUF is inspired by the notion of creating a circuit structure with metastable properties in FPGA matrix [25]. Similar to the SRAM-PUF, the floating state of Butterfly PUF can be exploited to obtain a random state at the startup phase of a pair of cross-coupled latches in the

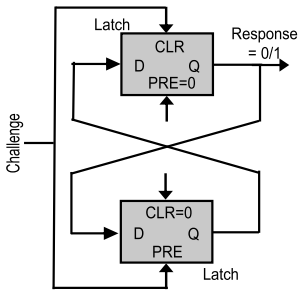

FIGURE 12.5

Typical butterfly PUF schematic.

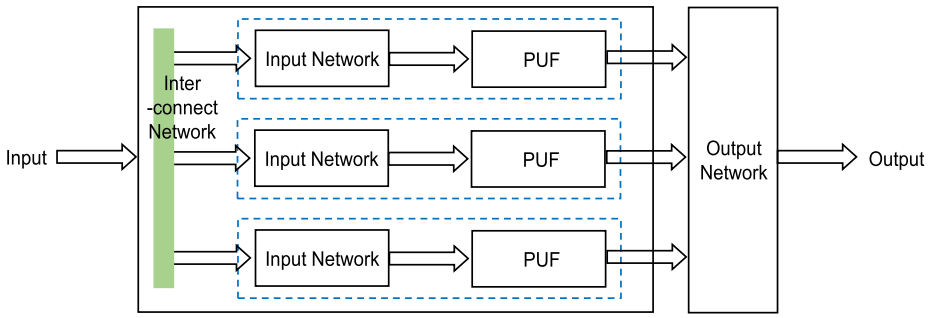

FIGURE 12.6

Generic structure of a lightweight PUF.

FPGA. The latches form a combinational loop that can be excited to an unstable state through a proper signal, that is, challenge as depicted in Fig. 12.5. Although conceived for FPGA implementations, similar implementation can be done utilizing the metastability of any latch or flip-flop-based architecture.

12.3.4.5 Lightweight PUF

Lightweight PUF, as shown in Fig. 12.6 [26], utilizes traditional arbiter-PUF with nontrivial wiring between arbiter stages. Rather than directly feeding the challenges to the PUFs, it creates a networked scheme that breaks down the challenge set into several blocks, and uses them on multiple individual PUFs. The output network then combines all the individual-PUF responses to create a global response, making it more resilient against machine learning attacks [27].

12.3.4.6 Bistable Ring PUF

A bistable ring contains an even number of inverters and can only have two possible stable outcomes. However, due to manufacturing process variation and noise, the bistable ring goes through a set of complex transitions (or metastability) before converging to a stable state.

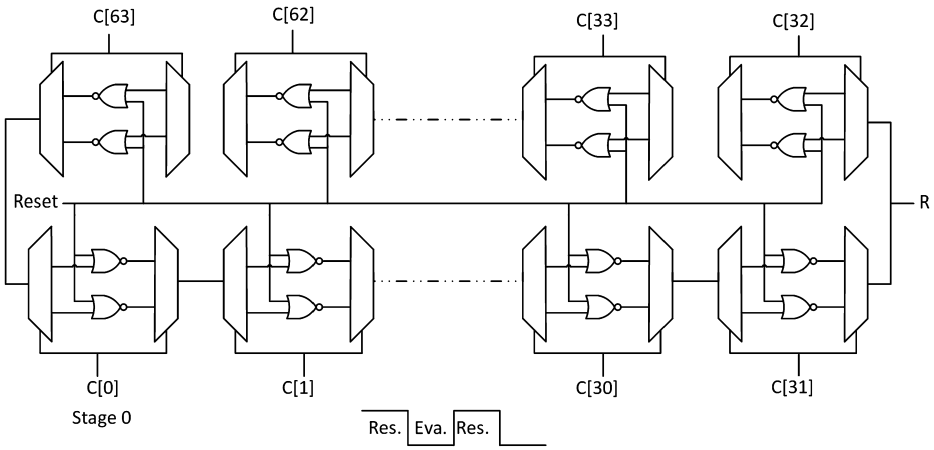

FIGURE 12.7

A 64-stage bistable ring PUF.

As illustrated in Fig. 12.7, a bistable ring PUF can exploit the metastability to generate exponential number of CRPs [28]. Similar to arbiter PUF, it requires a symmetric layout. Nevertheless, this strong PUF offers relatively high resiliency against modeling attacks due to its complex and nonlinear nature, and thereby can be incorporated into emerging PUF applications such as *Virtual Proof of Reality* [29].

12.3.5 PUF APPLICATIONS

12.3.5.1 Secret Key Generation

As mentioned in Section 12.3.1, PUFs can be used to generate secret keys for cryptographic applications. Since weak PUFs have a limited CRP-space, the response from such a PUF can be used as a key, where it does not require to be discarded every time after use. Additionally, a strong PUF can be used to generate a new key every time a cryptographic protocol is under use, for example, to generate a session key in a timely fashion. It should be noted that the "reproducibility" feature of the PUF is extremely crucial for cryptographic applications. Since even a slight mismatch (error) in the generated key due to measurement noise can corrupt the cryptographic application and underlying message, the generated key must be error-free, that is, 0% intra-HD. If not, the PUF (and associated modules) should be bootstrapped with efficient error-correcting code (ECC) memory scheme to provide zero bit flip.

Figure 12.8 shows the cryptographic key generation scheme presented in [22]. The ECC used here gets rid of any unwanted runtime error by initialization and regeneration. Furthermore, the generated key may lack necessary qualities to be treated as an ideal key for certain cryptographic protocols. For example, the key used for RSA needs to satisfy certain mathematical properties, whereas PUF-generated keys are typically arbitrary, due to nondeterministic process variations. Such properties can be achieved by using cryptographic hash algorithms with the PUF responses taken as inputs. Additionally, it helps to prevent any side-channel leakage of the originally generated key. This is especially true for weak PUFs that generates only a single key and, therefore, must be protected against any adversar-

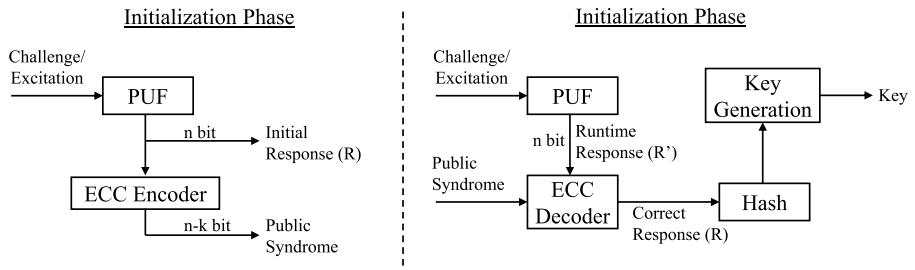

FIGURE 12.8

Cryptographic key generation with PUFs.

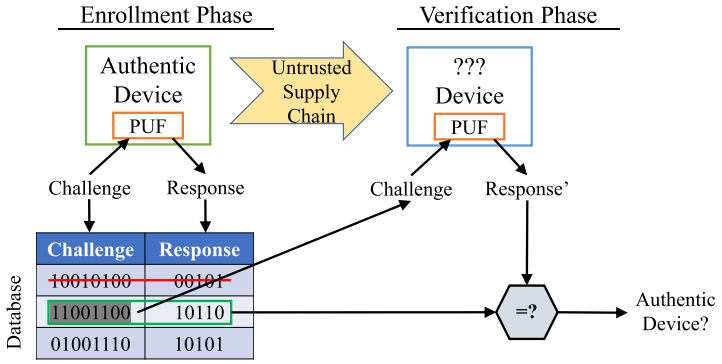

FIGURE 12.9

Strong PUF-based authentication.

ial access. To achieve further mathematical properties for the key (such as in RSA), the hashed output can again be input as a seed to an appropriate key-generation algorithm. Note that none of such steps require storing the key in a nonvolatile memory or leak the key to the outer world, given that a proper secure implementation is undertaken.

12.3.5.2 Device Authentication

Strong PUFs are excellent candidates to provide individual device authentication based on hardware-intrinsic signatures. Figure 12.9 shows a simple PUF-based authentication technique that does not require expensive cryptographic implementations, and can be easily implemented in a resource-constrained platform such as RFID, or in commercial off-the-shelf FPGAs, where cryptographic modules may not be readily available [22].

Since a strong PUF provides a unique and unpredictable output for an individual device, the response can be considered as a device-intrinsic identification signature/fingerprint. This can be used to authenticate (identify) individual devices given that the trusted authenticator already has a recorded copy of the CRPs. A trusted party, therefore, builds the CRP database for an authentic IC by applying random challenges to obtain unpredictable responses. For an in-field authentication operation, the

trusted party selects a recorded (but not previously used in the field) challenge and verifies the obtained response with the stored one. To protect against man-in-the-middle and replay attacks, the used CRP is discarded from the future usable CRP pool.

One should also note that practical authentication protocols usually keep an acceptable error margin in matching the stored and in-field responses to allow noise measurement during authentication. However, the PUFs used for authentication should have high a uniqueness to confidently identify different devices (that is, reduces the probability of ID collisions) even in the presence of small errors.

12.3.5.3 Software and IP Licensing

Another application for PUF is for software/IP licensing and certified executions. There have been various proposed schemes for device-bound certified executions [14]. For example, the PUF response can be used as a seed for a public/private key pair, and the seed is used to generate the private key. A certification authority publishes and certifies the public key. So a particular program can be prepared as copy/execution-protected using the PUF response and will not run on any other chip that does not satisfy the keys.

12.3.5.4 Hardware Metering

Researchers have also proposed PUFs for active and passive IC metering. In passive IC metering, the IP rights owner is able to identify and monitor the devices, whereas in active IP metering, the IP owner can actively control, enable/disable, and authenticate a device. This protects the hardware against foundry piracy (overproduction). Alkabani and Koushanfar [30] proposed a finite-state-machine (FSM)-based approach, where the functional specification of the design is modified and tied to the PUF outcome. This modified FSM architecture hides (obfuscates) a state in the large state-space of the FSM. A PUF-generated key puts the design one of the hidden obfuscated states (locked state) and provides a nonfunctional/erroneous outcome. Only an "unlocking key" provided by the IP owner can put the chip into the original functional state.

12.4 TRUE RANDOM NUMBER GENERATOR

12.4.1 TRNG PRELIMINARIES

A TRNG is a hardware primitive widely used in security and cryptographic applications to generate session keys, one-time pads, random seeds, nonces, challenges to PUFs, and so on, and these applications are growing in number with time [3,31,32]. It typically generates a random digital bitstream with high uncertainty, or entropy, where the sequence of producing 0s and 1s are equal, and completely independent to its previous value or any other external control. To generate an output that is truly random, a TRNG must rely on device intrinsic electrical and/or thermal noise that is inherently non-deterministic and uncontrollable.

A typical TRNG consists of an entropy source, entropy extraction or sampling unit and, in most cases, a postprocessing or cryptographic conditioning unit. The entropy source is the focal point of a TRNG, because the quality of the TRNG system highly depends on the raw entropy coming from this entropy source. For a TRNG, such sources are analog in nature, and include random telegraph noise (RTN) found in scaled transistors, power supply noise, radioactive decay, latch metastability, jitter in ring oscillators, and so on [32]. The throughput (speed) and power consumption of the TRNG

greatly depend on the physical component (analog or digital) used as a source and the resolution of the exploited noise.

The extraction or sampling unit extracts the entropy from the source into a favorable digital form. However, it should not impact the original physical process that produces the noise. The target of the sampling unit is to achieve maximum entropy under design constraints (such as area and power) and, in many cases, does not focus on the quality of the generated random bitstream.

The postprocessing unit focuses on improving the quality of the extracted bitstream from the sampling units to ensure the true randomness of the output. A good post-processor eliminates any hidden bias of the raw output [33]. Some common postprocessing techniques include Von Neumann Extractor [34] and cryptographic hash functions [35]. In many recent designs, the TRNG may contain additional conditioning blocks that can non-deterministically increase noise and/or makes it robust against runtime environmental variations and active fault and side-channel attacks [36–38].

In contrast to the hardware-based TRNG, a software-based "seemingly" random number generator, commonly known as a pseudo-random number generator (PRNG), relies on algorithms and tends to produce high throughput with lightweight implementations, although the output is statistically deterministic. A PRNG is not effectively secure, because its next state can be predicted from the current state if an adversary gains access to the design and knows the seed. In communication and cryptographic applications, a predictable RNG can expose the sensitive data to the adversary, and it is, in such cases, not "truly random" anymore.

12.4.2 TRNG QUALITY PROPERTIES

Unlike PUF quality metrics, the quality of a TRNG relies mostly on *randomness*. In order to assess the *randomness* of the bits produced by a TRNG, statistical test suites, such as the NIST Test Suite [20] and DieHARD [21], are commonly used and are usually the first (and the easiest) step to analyze the randomness of a TRNG.

One problem with TRNG entropy sources is that although they might be "intuitively random", statistical tests run on the output of the TRNG may show a certain level of bias and predictability, especially under conditions such as environmental and process variations. To combat this, cryptographic hash functions, von Neumann corrector, and stream ciphers are employed to manipulate the raw output of the TRNGs to ensure the uniformity and statistical randomness. Also, additional tuners and processing blocks may be employed to control the TRNG quality and throughput [36].

It should be noted that the operating condition of the TRNG is also a key factor for generating "truly" random numbers, as power supply variation, temperature deviation, clock frequency, added noise, or external signal, etc. can impact the intrinsic entropy source and extracted features. Hence, the *reliability* of the TRNG is also crucial in a sense that the TRNG itself needs maintaining the randomness throughout its operational lifetime and, additionally, show resiliency against attacks tailored with operational condition variation.

12.4.3 COMMON TRNG ARCHITECTURES

Typically, CMOS-based random number generators are designed by comparing two symmetric systems (or devices) that possess some process variation or/and ample amount of random inner noise to serve as an inherent entropy source. Based on the sources of randomness and system architecture, TRNGs can be categorized into: device's inherent noise-based TRNGs, jitter and metastability based TRNGs (that

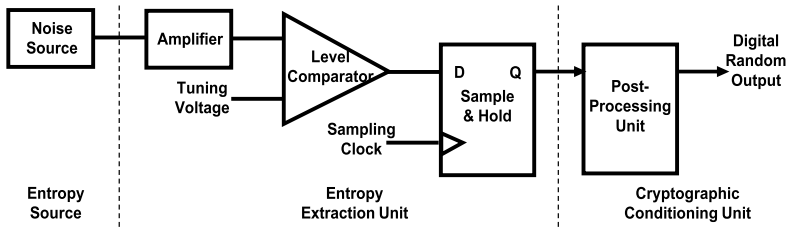

FIGURE 12.10

General schematic of noise-based TRNG.

is, free-running oscillator based TRNGs), chaos TRNGs, and quantum TRNGs [32]. However, not all TRNGs proposed in the literature are entirely CMOS-based, as some may require external optical/laser sources for excitation of the entropy sources. In this section, the discussion is confined to CMOS-based TRNGs.

12.4.3.1 Noise-Based TRNGs

Device-inherent noise is typically random and can be harnessed for generating true random numbers. Common noise sources include TRNG entropy sources (random telegraph noise (RTN)), Zener noise (in semiconductor Zener diodes), Flicker or $1/f$-noise, and Johnson's noise [32]. The basic idea of noise-based TRNG is as follows: the random analog voltage, caused by the noise source, is sampled periodically and compared to a certain pre-defined threshold to produce a "1" or "0" as shown in Fig. 12.10. The threshold can be fine-tuned to produce an ideally equal probability of "1"s and "0"s. However, setting up a proper threshold may be a rigorous process and may need readjustments and fine-tuning, based on run-time conditions.

The earlier version of Intel TRNG was developed leveraging Johnson's noise, where the source of randomness is the random thermal motion of charged carriers [39]. However, a more efficient, faster, and exceptionally simple TRNG was designed in 2011 (see Fig. 12.11) [3]. This TRNG design uses a pair of cross-coupled inverters (or a trimmed RS-type flip-flop), without any analog parts, making it extremely suitable for integrating with the logic chip. Ideally, the design is completely symmetric and both "set" and "reset" inputs are tied together and driven at the same time. Given any random mismatch due to noise, the nodes would be forced to settle to the stable output of either "1" or "0". However, the design is still not entirely free from bias, as any systematic variation would produce a highly biased output. Hence, additional current injecting mechanism and postprocessing techniques, such as raw bit conditioners and PRNGs, are required [3,32].

A major problem with different device-intrinsic noise sources is that not all of them can be appropriately measured, characterized, or controlled during manufacturing phase for proper application. Additionally, with the maturity of the process technology, some noise mechanisms that can be considered as entropy sources get more controlled. Eventually, these are suppressed adequately to not produce a reliable measurement (that is, voltage or current) of the entropy source. Therefore, the extractor unit needs to be quite sophisticated with the capability of strong amplification and sampling for converting the analog noise value to digital bitstream. This introduces further deviations from generated data being "true" random, due to amplifier bandwidth and nonlinear gain limitations. Also, the fast electri-

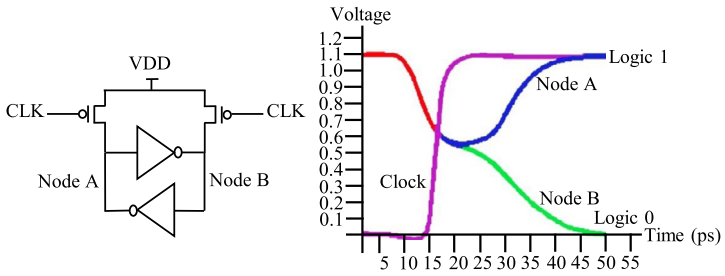

FIGURE 12.11

Intel TRNG. (A) Schematic and, (B) Transient Behavior.

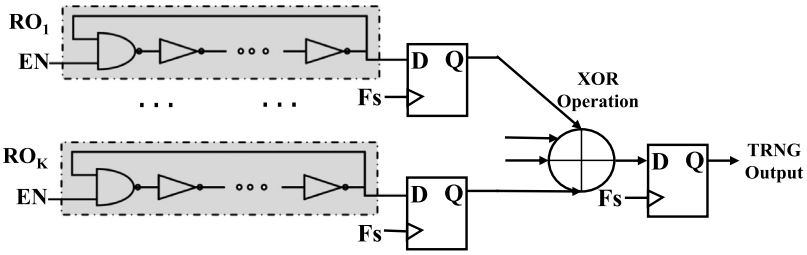

FIGURE 12.12

Ring oscillator-based TRNG.

cal switching of RNG circuitry produce strong electromagnetic interference, creating synchronization among nearby RNGs, causing a drop of overall entropy. Hence, there is a lack of provability of the randomness, since the leveraged noise source cannot be ensured as truly random, because of the effect of extractor unit, and other necessary deterministic postprocessing.

12.4.3.2 Oscillator-Based TRNGs

Another common approach to produce random numbers in the digital domain is to use oscillators and leverage associated jitter and metastability. Odd numbers of back-to-back connected inverters, with a feedback loop, act as a free-running oscillator (FRO), where even without an external input, the oscillator output is capable of driving itself as long as power is on [40]. Figure 12.12 shows a common design of a ring oscillator-TRNG (RO-TRNG). Random electrical noise in the feedback loop causes the frequency and phase of the oscillation to have jitter, that is, the exact time of the signal reaching the extraction point is not deterministic [37]. The entropy is further improved by proper sampling and XORing each FRO output.

One problem with jitter-based TRNG architecture is that the semiconductor industry is constantly working towards the minimization of jitters and noise. Therefore, additional noise-augmenting ring oscillators (NAROs) can be placed near the TRNG in the design to increase power supply noise by exhaustive oscillation [36]. Such NAROs are relatively smaller in length (hence, faster) and can be arbitrarily activated via a linear-feedback shift register (LFSR). Furthermore, in cases when the ran-

domness of output is too weak compared to the load it is driving, the effect of the oscillator sampling may be muffled. For such a scenario, a Schmidt action can be implemented in the input to increase reliability (although the speed is slower). A major concern for this design is that the randomness is not provable. As possible solutions, researchers have proposed different ring oscillator structures, such as Fibonacci ring oscillator and Galois ring oscillator, they can provide variability in the oscillator length (frequency) with different devices (and delay lines) [32].

Additionally, XORing multiple oscillators can produce further entropy in the generated random number, generally with the cost of low throughput. Different postprocessing techniques, such as Von Neumann corrector and cryptographic hash functions can be employed in such cases [41]. Sunar et al. [31] proposed a provable true random number generator with built-in tolerance. Amaki et al. [38] proposed a stochastic behavior modeling-based technique to detect the worst case for deterministic noise, and the oscillator-based TRNG design is noise-tolerant to satisfy the generated random bit-stream quality.

12.4.3.3 Memory-Based TRNGs

One can recall from Section 12.3.4.3 that the power-up states of SRAM blocks are nondeterministic and heavily dependent on the process variation and runtime conditions and noise. This can be used as an entropy source in a TRNG [23]. In contrast to the SRAM-PUF, the SRAM-based TRNG uses the cells with unreliable powerup states, that is, the cells that have statistically equal probabilities to produce "0"/"1" every time the chip powers up. This provides the required entropy that can be extracted by postprocessing to derive the random numbers. However, the throughput and randomness of the SRAM-based TRNG highly depend on the technology and resiliency against runtime variations. Furthermore, it is not always practical to power down and restart the SRAM block every time a new random number is needed.

Metastability-based TRNGs leverage similar technique as they use cross-coupled elements to amplify noise and generate random bits. In a metastability-based TRNG, bits are generated by repeatedly biasing a single cross-coupled element precisely to the point of metastability. The metastability is resolved to a stable logic as determined by the noise.

12.4.4 TRNG APPLICATIONS

Keeping the basic Kerckhoffs' assumption [42] in mind, one can see that the security of a cryptographic system should rely solely on the key, and not on the design of the system (that is, no implementation flaws or side-channel information leakage). This key is often taken from a pool of random numbers. Hence, the quality of numbers generated by a TRNG is extremely important, since it directly determines the security strength of the system. A completely random key can ideally be broken only by brute-force (random guessing) attacks. Any predictability in the key-bits reduces the guessing-space of an adversary, and leads to a weakness in the entire system.

The main application for electronic hardware random number generators is in cryptography, where they are used to generate random cryptographic keys to transmit data securely. They are widely used in Internet encryption protocols, such as secure sockets layer (SSL). For example, Sun Microsystems Crypto Accelerator 6000 [43] contains hardware TRNG to provide the seeds to a FIPS-approved RNG specified in FIPS 186-2 DSARNG, using SHA-1 for generation of cryptographic keys for SSL hardware acceleration (TLS acceleration).

Many other common security protocols require random bits to remain secure and unpredictable by an adversary. True random numbers are widely used in many applications, such as keys and initialization values (IVs) for encryption, Session key generation for conventional encryption, keys for keyed MAC algorithms, private keys for digital signature algorithms, values to be used in entity authentication mechanisms, values to be used in key establishment protocols, PIN and password generation, one time padding, generating nonces to protect against replay attack, and input challenge for strong PUFs.

12.5 DESIGN FOR ANTI-COUNTERFEIT

12.5.1 DfAC PRELIMINARIES

In today's complex electronic component supply chain, it is very challenging to detect and prevent the infiltration of counterfeit chips and FPGAs. As mentioned in Section 12.2.2, a proper exploitation of electrical characteristics can lead to a cheaper, faster, and more successful detection of counterfeit electronics. Since aging and wear-out mechanisms generally make a chip slower over time, one can estimate aging degradation of a circuit under test (CUT) by measuring its speed and comparing it with a reference speed from original unused (golden) chips. However, acquiring such reference (golden) measurements is not always feasible. In addition, manufacturing process variation and defects cause deviation in speed/delay or other electrical measurements, even for golden chips. Hence, the approach requires a large pool of golden data to maintain statistical significance. These issues are more prominent for legacy chips and CoTS [44]. By exploiting such aging issues, researchers have proposed several techniques, such as embedding design for anti-counterfeit (DfAC) structures into the chip, or measuring degradation due to accelerated aging, for recycled IC detection.

12.5.2 DfAC DESIGNS

A combating die and IC recycling (CDIR) scheme takes aging into account to determine whether a chip has gone through a prior use. It is a lightweight and low-cost DfAC sensor with an RO-pair for self-referencing that eliminates the need for golden data [4,45]. The concept behind the RO-CDIR scheme is to put additional circuits or architectures, that is, ring oscillator structures, into new electronic chips in a way that the RO frequency tends to degrade more with aging because the transistors in the RO gets slower with time. The key points here are to employ a very lightweight design that would not practically impact the area, power, and cost requirement of the original chip, and the implemented design should produce readily available data that can reliably predict the aging (if any) or the freshness of the chip; the chip must age rapidly and must not be affected during the testing and validation phase. Also, the impact of process variations and temperature must be minimized, and path delay deviation due to process variations and aging degradation must be satisfactorily separable.

One key point to measure aging-induced delay-degradation is that the RO frequencies are needed to be compared with the golden (fresh) or reference data. The typical RO-CDIR sensor performs a self-referencing scheme to reliably measure delay degradation by comparing the frequencies of two ROs of length, named reference-RO and stressed-RO, respectively (see Fig. 12.13). Reference-RO is designed to age slowly or not age at all, if possible. On the contrary, the stressed-RO is designed to age at a much faster rate. When in operation mode, the stressed-RO's rapid aging reduces its speed

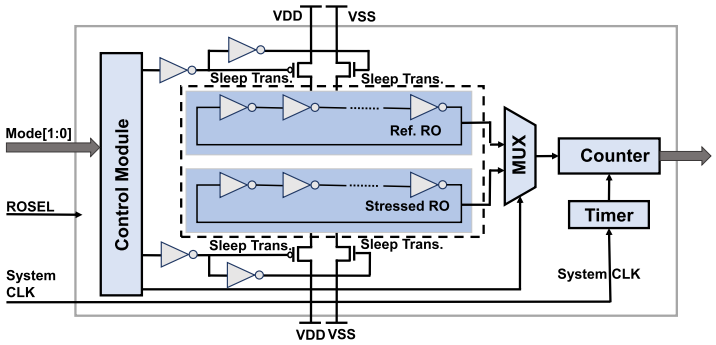

FIGURE 12.13

Combating die recovery (recycling) sensor using stressed and reference RO-pairs.

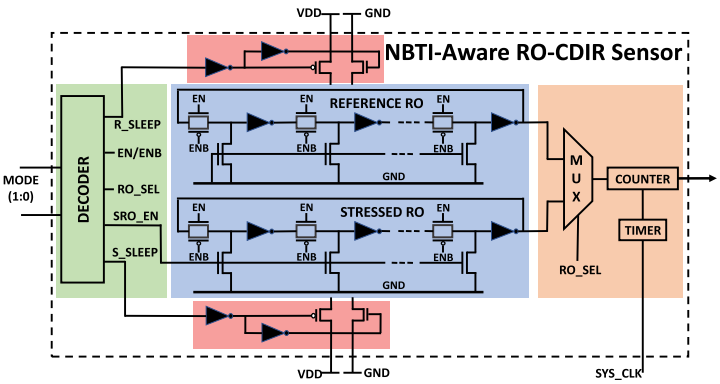

FIGURE 12.14

NBTI-aware RO-CDIR sensor.

(frequency), whereas the reference-RO's speed (frequency) largely remains the same. Thus, a large difference between the RO-frequencies implies that the chip has been used. A close physical placement of the ROs further reduces global and local process variations and environmental variations to give a finer measurement of the usage time. However, a limitation of this approach is that half of the PMOS transistors experience DC NBTI stress, hence experience a limited degradation due to the oscillatory nature of the scheme.

Figure 12.14 shows an NBTI-aware RO-CDIR sensor that exploits NBTI-induced degradation for an improved detection scheme [46]. While in operation mode, it gives maximum NBTI (DC) stress to the stressed-RO by breaking the RO chain and connecting all inverter inputs to the ground, so that they do not get a chance to recover from aging. However, a partial recovery may occur when the chip is completely powered off. The stressed-RO's structure is mimicked by the reference-RO to avoid parametric variations. However, during operation, the reference-RO is kept disconnected from the power

and ground line to minimize aging. Since the two ROs have different aging stress, their frequencies continue to deviate over time, and it increases the probability of a more accurate detection.

12.6 EXISTING CHALLENGES AND ATTACKS

12.6.1 PUFs

12.6.1.1 Modeling Attacks Against Strong-PUFs

Strong PUFs, by definition, are capable of producing an exponential number of challenge-response pairs (CRPs). However, such a large possible CRP-set makes strong PUFs potentially vulnerable to machine-learning assisted modeling attacks [27]. In general, modeling attacks on strong-PUFs starts with an adversary getting hold of a subset of all CRPs of the PUF under attack. Using this CRP-subset, the adversary tries to derive a numerical model to correctly predict the PUF's responses to additional arbitrary challenges. This further enables the attacker to launch a man-in-the-middle and impersonation attack on the existing PUF-based authentication and key generation protocols. Researchers have launched multiple machine-learning- (ML) technique-based modeling attacks on the traditional arbiter-PUF, an ideal example of strong PUFs, and their successes have led to necessary modifications, resulting in several relatively more attack-resilient compositions of the traditional arbiter-PUF [47].

The basic numerical model of a traditional arbiter-PUF is based on the additive linear delay model. The overall delays of the signals that are propagated through multiple paths of the delay-stages can be expressed as the sum of the delays in the stages and associated interconnects, and the response can be determined by the final delay difference, assuming that the arbiter has ideally zero bias [47]. This simple, yet powerful, model thus leads to creating a two-class classification technique based on linear-delay-based hyperplane, trained with the collected/leaked CRP-subset by the adversary.

Researchers have proposed several techniques to strengthen the modeling-attack resiliency of the traditional arbiter-PUF by introducing nonlinearity into the architecture. One such example is the XOR-arbiter PUF [22], presented in Fig. 12.15. This PUF contains multiple same-length arbiter-PUFs, excited with the same challenge. The output of the individual arbiter-PUFs are XORed in order to produce a final response, and hence provides a high level of nonlinearity.

Nonlinearity of arbiter-PUF can also be increased by introducing feed-forward connections into the delay path, as shown in the simple feed-forward arbiter-PUF structure in Fig. 12.16 [47]. This structure utilizes the "unknown" challenges within the PUF architecture, where the intrinsically generated challenges are fed to the forthcoming delay blocks. Hence, the path switching of the delay stages connected to the "feed-forward loop" depends on the behavior of the previous stages. Such dependency creates a non-distinguishable functional model for the arbiter-PUF. As a result, the feed-forward arbiter PUF shows resiliency against machine-learning attacks that utilize linearly separable or differentiable models. Also, the designer can choose the number of feed-forward loops and connection points as necessary, making the attack model more complicated.

However, both such variant compositions of arbiter-PUF are not absolutely resilient from modeling attacks. Hospodar et al. [48] and Ruhrmair et al. [27] presented exhaustive attack results on different compositions of arbiter-PUFs. It is shown that the traditional arbiter- and xor-arbiter-PUF both can be easily modeled to a very high accuracy (for example, $\sim 99\%$), using popular machine learning

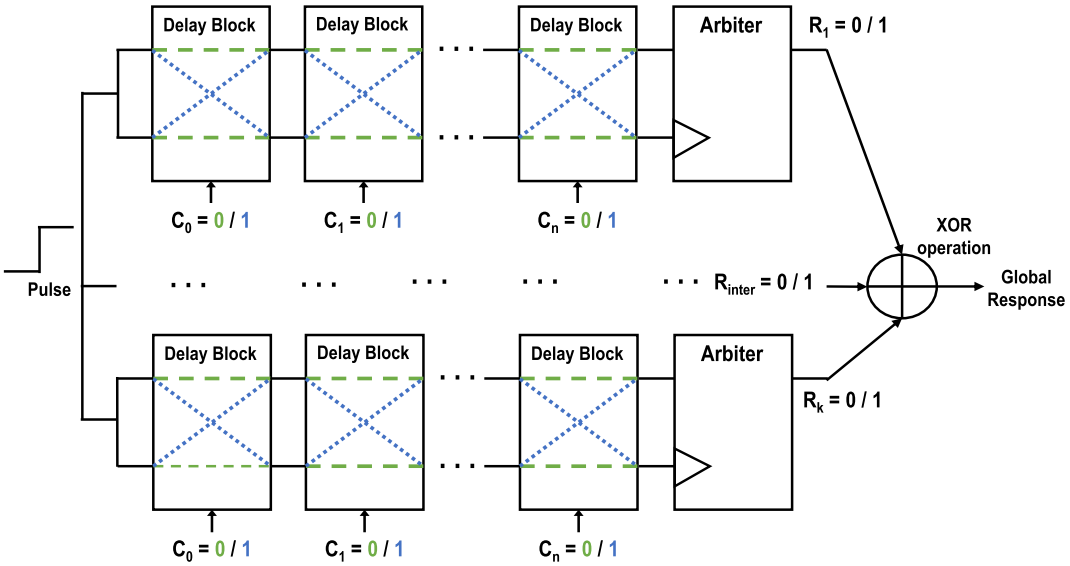

FIGURE 12.15

Schematic of an XOR PUF. It consists of *k*-chains of *n*-bit arbiter PUFs. The output of all arbiters are XORed together to generate the final binary response.

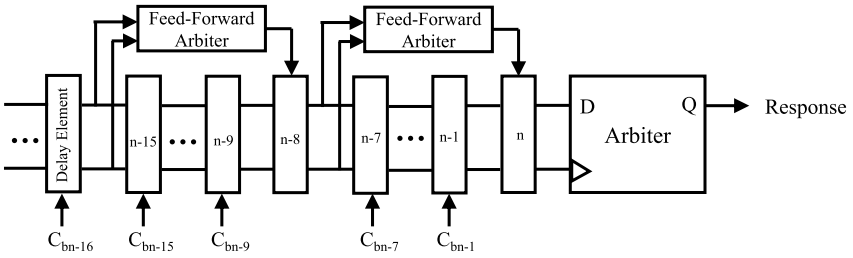

FIGURE 12.16

Structure of feed-forward arbiter PUF, where feed-forward loop size is 8.

techniques, such as logistic regression at the cost of modeling time. Ruhrmair et al. [27] also showed that machine learning techniques that use nonlinear classifications, such as evolution strategies can be used to correctly predict the CRP set of a given feed-forward arbiter PUF to a high accuracy.

12.6.1.2 Environmental and Aging Impacts on PUF

Environmental variations, such as temperature and power supply noise, and aging and wear-out mechanisms, generate unwanted errors in PUF outputs, making it unreliable for cryptographic applications. This is more prominent in RO-PUFs, mostly because of the continuous switching of all those os-

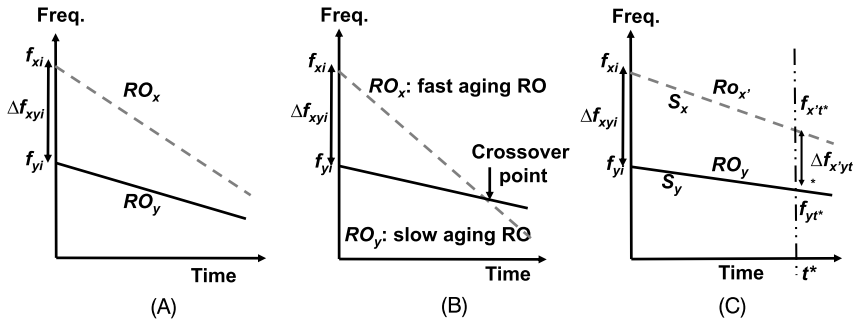

FIGURE 12.17

Bitflip due to frequency degradation in RO-pair. (A) ROs with moderate degradation (stable pair). (B) ROs with high degradation (unstable pair). (C) ROs with negligible degradation (highly stable pair).

cillators. To understand why RO-PUF generates erroneous responses, let us revisit the conventional RO-PUF shown in Fig. 12.3, and consider the frequency profile of a randomly selected RO-pair shown in Fig. 12.17 [49]. For the given RO-pair, if the frequency of RO_x ring oscillator (f_{xi}) is greater than that of RO_y (f_{yi}), then "1" (otherwise "0") is generated as a response (Fig. 12.17A). However, it fails to generate a reliable (that is, the same as before) response if a crossover happens (meaning, $f_{xi} < f_{yi}$ after possible frequency degradation due to environmental variation and aging (Fig. 12.17B). For maintaining maximum reliability, the two frequencies should never cross each other, maintaining a minimum frequency difference (that is, the frequency threshold (Δf_{th})) to compensate counter resolution, if necessary, till the end of operational lifetime t^* (Fig. 12.17C). Other PUF structures also suffer from similar reliability issues.

To produce reliable (error-free) response, bootstrapping efficient error correcting code (ECC) to the PUF can generate reliable output up to a certain margin, despite the presence of noise [50]. However, it relies on helper data that may partially reveal the secret key and potentially compromise the PUF. Most ECC schemes require redundant gates and an additional decode unit. Thus, ECC incurs large area, power, and timing overheads, making it impractical in resource-constrained applications.

NBTI and HCI-aware aging resistant ARO-PUF was proposed by Rahman et al. [49], as shown in Fig. 12.18. It has additional pull-up and pass transistors within the conventional RO-PUF architecture to reduce possible aging degradation. It has two modes of operation. At oscillatory mode (EN = 1), it performs regular PUF operation (Fig. 12.18B). In the nonoscillatory mode (EN = 0) (Fig. 12.18C), it removes DC stress for PMOS transistors, as it ties them to V_{dd} to eliminate NBTI. It also breaks the RO chain and removes AC (oscillatory) stress to eliminate HCI. So, this design successfully minimizes aging degradation, due to NBTI and HCI, by eliminating stress when the PUF is not active.

Yin et al. [51] proposed a temperature-aware cooperative RO-PUF (TAC RO-PUF) scheme to reduce error due to temperature variations. It allows unstable response generation as long as it can be converted into reliable bit by choosing different RO pairs, where temperature variation can cause bit flip. Experimental results indicated an 80% improvement in stable bit-generation capacity. Addition-

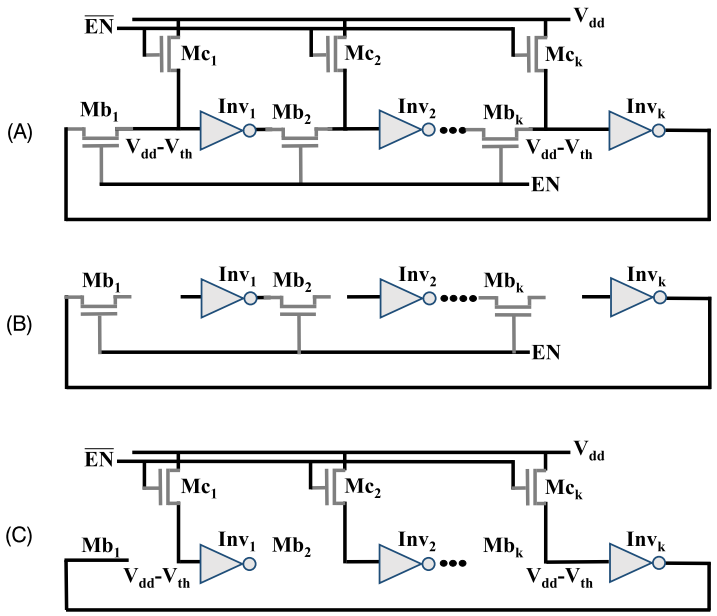

FIGURE 12.18

Operation of the aging-resistant RO (ARO)-PUF: (A) Schematic; (B) Activity in oscillatory mode, and (C) Activity in non-oscillatory mode.

ally, Rahman et al. [52] proposed a reliable pair formation scheme, called RePa, for RO-PUFs to improve robustness against both runtime variation (temperature variation and power supply noise) and aging degradation. It ranks all the ROs based on a predictive aging/voltage-based degradation profile and forms the most-suitable RO-pairs for PUF response generation using a complex algorithm, considering initial frequency differences, speed degradations, and bit-flip probability. This approach has shown to achieve up to 100% reliability, that is, zero error, with $\sim 2.3x$ ROs, eliminating the requirement of a much larger ECC.

12.6.1.3 Cloning SRAM-PUF

Helfmeier et al. [53] have demonstrated the very first physical cloning of the SRAM-PUF. This essentially violates the ideal PUF property of being "unclonable". The physically cloned SRAM-PUF can produce a response that is identical to the original PUF, as shown in Fig. 12.19. It should be noted that this attack is invasive in nature and uses expensive focused ion beam circuit edit (FIB-CE) techniques. This attack first observes the fingerprint of the target SRAM-PUF (see Fig. 12.19A–B) and modifies individual cells of another SRAM-PUF to produce (clone) the exact same signature (see Fig. 12.19C–D). Using the FIB-CE, individual transistors can be modified/removed completely to achieve a deterministic behavior. Also, the individual transistors can be trimmed to alter their dynamic performance and leakage characteristics.

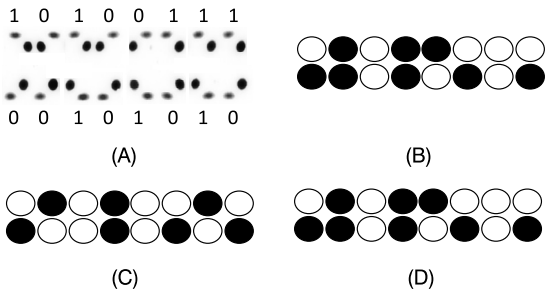

FIGURE 12.19

Physical cloning of the SRAM-PUF: (A) Photonic emission of the target SRAM device; (B) Logic fingerprint of the target device (dark = Logic '0'); (C) Initial fingerprint of the cloned device; (D) Initial fingerprint is modified via FIB CE to match the target device.

12.6.2 TRNGs

A TRNG is affected by limited intrinsic variations, especially, in the older and mature technologies. For such cases, the inherent entropy sources may not be sufficient enough for harvesting true randomness and obtaining maximum throughput. Further, the randomness of TRNGs can become even worse under environmental variations and different aging mechanisms. This opens up a variety of hardware-based attacks on TRNGs. For example, an attacker can vary the device supply voltage (V_{dd}) and temperature beyond the nominal condition, and intentionally bias the output to extract the "predictable" bitstream [36]. A frequency injection attack on RO-based TRNGs can lead to the clock jitter impacting the entropy, and, thereby, can facilitate the guessing of the key, for example from a smart card, with minimal effort [54]. Additionally, electromagnetic attacks can leak this information without physically destroying the chip [55].

Temperature variation poses a more potential threat to SRAM-based TRNGs. Randomness in ICs is, at least partially, coming from physically random thermal noise. For example, the magnitude of thermal noise exploited during powerup state depends on the temperature. A lower temperature reduces the exploitable noise, making the produced bitstream less random, whereas a higher temperatures make the power-up state more random [23].

To achieve more uniformity and statistical randomness in spite of low inherent entropy, researchers have proposed cryptographic hash functions, von Neumann corrector, and stream ciphers to be employed at the TRNG outputs. However, these additional modifications reduce the throughput and increase area and power overhead. Further, vendor agnostic TRNG design for FPGA was proposed by Schellekens et al. [56]. Rahman et al. [36] proposed a technology-independent (TI) TRNG to combat the security issues arising from such various run-time-/environmental-based degradations. It uses a "tunable"-RO architecture to leverage power supply noise along with clock jitter as the entropy source, and can overcome environmental variation and aging-induced bias by controlling jitter, and adjust RO delay by monitoring the run-time condition. A power supply noise enhancement and tuning block, and a self-calibration scheme on bias detection further improve the performance and serve as countermeasures against the hardware-based attack. The TRNG model presented by Robson et al. [57] utilized multiple threshold crossing methods to increase timing jitters.

Device	Manufacturing Variability Sources	Exploitable Features	Reliability Factors
PCM	• Geometric Variation (GST layer thickness and Bottom Electrode Contact Diameter) • Cell Resistance (R_{Cell}) • Write/Read Strength (required min. reset current) • Write current magnitude (I_{Write})	• Cell Variability - Stochastic Resistance Variation per cell • Programming sensitivity - variation in resistance level due to the nature of applied pulses	• Power supply Noise and Temperature Variation • Resistance Drift • Poor Endurance and Retention
Memristor & RRAM	• Variability in dope and undoped region length • Device thickness and Cross-sectional area • Stochasticity in doped and undoped resistance.	• Stochastic switching mechanism and intrinsic variability of devices. • Resistance variability due to applied voltage pulse duration	• Power supply Noise and Temperature Variation • Moderate Endurance and Retention • Read Disturbance
MRAM & STTRAM	• Geometric variation in free layer volume • Spin-torque Switching • Threshold voltage • Thermal Stability • Critical Switching Current	• Read Current variation • Meta-stability of free layer magnetization. • Variation in thermal energy • Back-hopping	• Temperature • External EM field • High Endurance and Retention

FIGURE 12.20

Emerging nanodevice properties for PUF applications.

12.7 PRIMITIVE DESIGNS WITH EMERGING NANODEVICES

Emerging memory devices, such as phase change memory (PCM), memristor, resistive random access memory (RRAM), and spintronic memory devices, such as spin-torque-transfer random access memory (STT-RAM), and magnetic random access memory (MRAM) are gaining popularity for high-capacity and low-power non-volatile storage applications [58]. Apart from being used as a traditional storage space, applications of emerging nano-device as security primitives have also gained much attention, as it can offer authentication and secure operation, reducing area and computational overhead. Researchers have proposed several hardware security primitives, with a majority being PUFs, over the last decade. As shown in Fig. 12.20, these emerging devices contain intrinsic features suitable for PUF applications in a similar fashion to traditional CMOS-based devices [59].

12.7.1 COMPOSITION OF PCM-BASED PUFs

12.7.1.1 Sources of Process-Induced Variability

PCM, one of the emerging NVMs, operates through a reversible transition between the low resistance (crystalline phase) and high-resistance (amorphous phase), which can be controlled through 'set/reset'

current pulses with the pre-defined magnitude and pulse duration [60]. The resistance of the PCM cells varies randomly during this programming process, based on the set/reset current pulse. Also, the manufacturing process variation affects the physical dimensions and the strength of PCM cells, consequently inducing stochastic nature in the electrical properties. Cell geometry features (such as GST layer thickness, heater thickness, and bottom electrode contact diameter) and the electrical and thermal conductivities of the GST, and heater material are prominent factors responsible for the process variations [61]. Due to the inherent randomness originating from the cell geometry and other structural and characteristic variations, and the different types of dynamics exhibited by PCM cells, they are well-suited for cryptographic measures, especially for PUF applications.

12.7.1.2 PCM-PUF Designs

The process variation and programming sensitivity of PCM can be exploited to implement PCM-based PUFs. A novel design for generating cryptographic keys via reconfigurable PUF, based on PCM cells, is proposed by Zhang et al. [62]. It uses the variations between PCM cells in an array to generate keys by random cell selection. The PCM-PUF itself is composed of the traditional crossbar architecture, where, for the PUF application, individual PCM cells or a set of cells in the crossbar array can be programmed with a pre-defined pulse, and later read out to produce a unique response. Although all the cells can be programmed with the same pulse (which is analog in nature before any comparative sensing), the generated read-out value will be unique from cell to cell within the same circuit, and also across different circuits, due to the stochastic nature of the PCM cell. Due to the nonvolatile nature of the cell, the same unique response can be regenerated, considering an ideal noise-free scenario. It should be noted that the composition of the PCM-PUF, being a traditional cross-bar architecture, would result in a "weak" PUF, that is, it will have a small number of CRPs, unless the memory access mechanism is designed to perform a random and nonlinear access technique. The produced signature may also require further postprocessing for providing output suitable for cryptographic operations.

Generation of unique signature from PCM crossbar architecture can be further extended by varying the cell-resistance distribution using programming pulse modification [63]. Additionally, the multilevel cell operation of PCM cell has been exploited to generate unique signatures in [64].

12.7.2 COMPOSITION OF MEMRISTOR AND RRAM-BASED PUFS

12.7.2.1 Sources of Process-Induced Variability

Similar to PCM, it is possible to exploit the write mechanism of memristors and RRAMs, and extract random cell behavior influenced by process variations for potential PUF applications. Since these devices have a larger resistance window, one can exploit the uncertainty in the logic state from the undefined region between high and low states. For memristors, such analog resistance variability and associated memory write/read time depends on the cell structure and dimensions, such as device thickness, which can be leveraged as entropy sources due to manufacturing process variation. Additionally, the source of entropy in RRAMs originates from intrinsic features, such as oxide thickness and defect density, and these can very well be exploited for PUF functionalities.

12.7.2.2 Memristor and RRAM-PUF Designs

The effect of process variability in memristor cell operation is exploited in [65] to develop a write-time-based memristive PUF-cell. Here, the time for the write operation is set to the minimum value required

to switch the cell from high resistance state to low resistance state. Hence, generating a probability of the output logic being "1" is raised to 50%, and vice-versa. Researchers have also proposed RRAM-based PUFs that utilize the traditional cross-bar memory architecture [66]. This composition improves the accuracy of split references, using dummy cells that eventually reduces the offset by decreasing the transistor size of the split sense amplifier (S/A).

12.7.3 COMPOSITION OF MRAM AND STTRAM-BASED PUFS

12.7.3.1 Sources of Process-Induced Variability

Spintronic devices, such as STT-RAM and MRAM, also offer novel opportunities for hardware security applications as they exhibit device-intrinsic phenomena, such as chaotic magnetization, statistical read/write failures, stochastic retention failure, and back-hopping as depicted in Fig. 12.20 [67]. The chaotic and random dynamics of the free layer of the magnetic tunnel junction (MJT) is exploited by researchers to develop hardware security primitives, for example, PUFs [68,69]. Additionally, the statistical and stochastic nature of read-write failures, back-hopping, and retention times can also be exploited to develop novel spintronic circuit-based PUFs and TRNGs [70]. The compatibility of such spintronic devices with silicon substrate makes them a potential candidate for complementing existing CMOS-based security primitives.

12.7.3.2 MRAM and STT-RAM PUF Designs

Physical variation-dependent MPAM-PUF is proposed by Das et al. [69]. This technique exploits the random tilt generated in the energy barrier of MTJs, due to manufacturing process variations and extracted the randomness to generate PUF responses. As the distribution of tilt angle is Gaussian in nature, the free-layer orientation of the MTJs is inclined to a certain initial value similar to the behavior of the SRAM-PUF. Additional improvements, such as gradual decrement of aspect ratio at a constant volume, and increment of volume at constant ratio, enhance the variation in tilt, and obtain more stable PUF output.

The random initialization of the free layer in STT-RAM is also utilized in designing PUFs [68]. In this technique, the responses are generated during the registration phase by comparing the STT-RAM bits of complimentary rows. Noise and sense amplifier offsets are utilized to produce response bits in the case of comparison bits, which are initialized with similar values. The repeatability of bit generation is ensured by writing the values at MTJs. The write-back is employed to preserve the responses for multiple accesses, and to prevent bit flips due to voltage and temperature variations. However, a downside is that this desing requires post-processing architectures, such as a fuzzy extractor, to maintain and enhance the quality of the PUF output.

12.7.4 PUF COMPOSITION FOR EMERGING APPLICATIONS

In addition to traditional key-based cryptographic applications, researchers have proposed several emerging security applications, such as virtual proofs of reality [29], that utilize the inherent characteristics of the PUF. The idea of virtual proof (VP) is based on the underlying fact that there are possibilities to verify the correctness and authenticity of the digital data obtained from tangible physical system properties or processes between two communicating parties, that is, prover and verifier, located at a distance without using any secret key-based classical mechanisms. As classical keys are

deemed vulnerable to physical- and software/malware-based attack techniques, the avoidance of the keys might lead to safer, cost effective, and compact designs of modern cryptographic hardware. The example of VPs, based on witness objects (WOs), for instance temperature variant ICs, disordered optical scattering media, and quantum systems, can be novel protocols that successfully verify the temperature, relative position of objects, or destruction of specific physical objects.

As for proof-of-concept experimentation, the VPs of temperature, and optical systems for VPs of relative distance, co-locality and destruction are demonstrated in [29]. Several novel variants of PUFs can be exploited in this regard. For example, the temperature proof can be obtained by CMOS-based bistable ring PUF, where the high-temperature sensitivity is advantageously exploited. To verify the VP of distance and destruction claims, one can employ variants of optical PUF.

These virtual proof mechanisms do not necessarily utilize the traditional PUF key-generation concepts, where the CRPs required to maintain some significant properties (for example, CRPs need to be "robust" across all temperature/voltage corners). For instance, the virtual proof of temperature, in fact, utilizes the high error-rate occurring in the CRPs at different temperature corners to obtain temperature-dependent CRPs (or signatures), which allow the verifier or the authenticator to obtain temperature information. For such a specific application, the underlying PUF composition would rather be tailored to be highly impacted by the temperature, as opposed to traditional robust PUFs. Exploiting the active components, such as transistor features, and variations in logic cells, should lead to such PUF designs targeting particular emerging applications.

12.8 HANDS-ON EXPERIMENT: HARDWARE SECURITY PRIMITIVES (PUFs AND TRNGs)

12.8.1 OBJECTIVE

This experiment is designed to give students an exposure to hardware security primitives such as PUFs and TRNGs.

12.8.2 METHOD

The experiment consists of two major parts – first part dealing with the design/analysis of PUFs and the second part with design/analysis of TRNGs. Each part consists of multiple components. All parts of the experiment are designed on the HaHa platform. The first part of the experiment illustrates the design of PUFs. Students will utilize the differences in power-up states in the SRAM to create random binary signatures, which can be used for authentication or cryptographic key generation. They will also map Ring Oscillators (ROs) in the FPGA to create an RO-PUF structure and use it to generate 128-bit keys. Next, the students will vary the operating voltage using a built-in potentiometer in HaHa board to obtain PUF responses at different operating voltages. The second part of the experiment focuses on the TRNG generation, where ROs mapped in the FPGA in the first part are made physically symmetric in order to create high-quality TRNGs.

12.8.3 LEARNING OUTCOME

By performing the specific steps of the experiments, the students will learn ways to exploit the intrinsic manufacturing as well as temporal variations in silicon devices for creating strong security primitives. They will also learn the metrics/processes to analyze various properties of these primitives, such as the level of uniqueness, randomness and robustness as well as hardware overhead of the PUFs and TRNGs.

12.8.4 ADVANCED OPTIONS

Additional exploration on this topic can be done through modification in PUF and TRNG structures to improve their various security properties.

More details about the experiment are available in the supplementary document. Please visit: http://hwsecuritybook.org/.

12.9 EXERCISES

12.9.1 TRUE/FALSE QUESTIONS

1. Weak PUF is ideal for detecting counterfeit ICs via CRP-based authentication schemes.
2. Error Correction Code (ECC) schemes can be used to improve the robustness for both PUF responses and TRNG outputs.
3. A ring oscillator (RO)-PUF is basically a delay-based PUF.
4. A TRNG must rely on random intrinsic or runtime noise.
5. Depending on the startup behavior, the SRAM-array can be used both as a PUF and a TRNG.
6. Runtime variation (such as power supply noise and Vdd fluctuations) is good for PUFs, but unacceptable for TRNGs.
7. DfAC structures, such as CDIR sensors, utilize aging phenomena for detecting prior usages of the IC.
8. Ideally, PUFs should have 50% intra-Hamming distance.
9. Ideally, TRNGs should have 50% intra-Hamming distance.
10. Frequency injection attack on an oscillator-based TRNG reduces the throughput, but entropy remains the same.

12.9.2 LONG-ANSWER TYPE QUESTIONS

1. Briefly discuss the major characteristic differences between PUFs, TRNGs, and DfACs.
2. Briefly discuss the quality metrics used to evaluate PUFs and TRNGs.
3. Explain why RO-PUFs tend to produce more erroneous responses due to runtime noise and aging compared to traditional arbiter-PUFs.
4. Briefly describe what security primitives (strong PUFs, weak PUFs, or TRNGs) are ideal for following applications:
   (a) Chip ID generation.
   (b) Authentication.
   (c) Licensing.

Table 12.2 RO frequency degradation over time

RO Name	Initial Frequency (F_0)	Predicted Frequency After 5 Years ($F_{5\ \text{years}}$)
RO1	5.31 MHz	5.27 MHz
RO2	5.30 MHz	5.24 MHz
RO3	5.27 MHz	5.23 MHz
RO4	5.41 MHz	5.35 MHz
RO5	5.35 MHz	5.30 MHz
RO6	5.22 MHz	5.19 MHz
RO7	5.26 MHz	5.22 MHz
RO8	5.39 MHz	5.36 MHz

(d) Cryptographic nonce.
(e) Key-generation for in-situ AES encryption/decryption system.

5. Briefly explain the differences between a TRNG and a PRNG? Which one of them would you consider, if you were only interested in:
(a) High entropy.
(b) High speed.
(c) Low runtime noise.

6. Explain how you can increase the security of the response of a weak PUF (used for key generation) against possible guessing or side-channel attacks. [Hint: One common technique is to use the hashing mechanism.]

7. Briefly explain the major challenges that PUFs and TRNGs suffer.

8. Consider the conventional RO-PUF shown in Fig. 12.3. Assume that it contains eight independent ROs with the initial and predicted free-running frequencies given in Table 12.2. The RO pairs are formed at the very begining of the operation (that is, at time $(t) = 0$) either by choosing randomly or by doing "intelligent" pairing. The given PUF response behavior is
"$True(RO_x \geq RO_y) \Rightarrow Response = 0$".
(a) What is the bit error rate (BER) after 5 years of operation if the RO pairs are formed randomly?
(b) One can perform an "intelligent pairing" by forming the RO pairs with the predicted frequency degradation (due to aging) taken into account. What is the minimum BER after 5 years of operation if the RO pairs are initially formed considering the predicted frequency degradation?

9. Consider the SSL/TLS hardware accelerator (see Section 12.4.4) that is required to generate a 128-bit random key at a speed of 1 Gbps (system clock speed). However, the hardware TRNG inside it produces raw true random bits at a 1000x slower speed. Provide a scheme that can produce random output at a required speed. For simplicity, you can assume the key-bits are generated in parallel.

10. Identify the following security primitives as either PUF or TRNG. Briefly justify your choice.
(a) Figure 12.21A shows that the intra-chip hamming distance = 50% of the output of a security primitive X. That is, for the same primitive/device, the output bits are in average 50% different for the same environmental condition and same input pattern.
(b) Figure 12.21B shows that the inter-chip hamming distance = 50% of the output of multiple security primitives of the same type Y. That is, for a pool of same primitive/device, the output bits are in average 50% different for different instances.

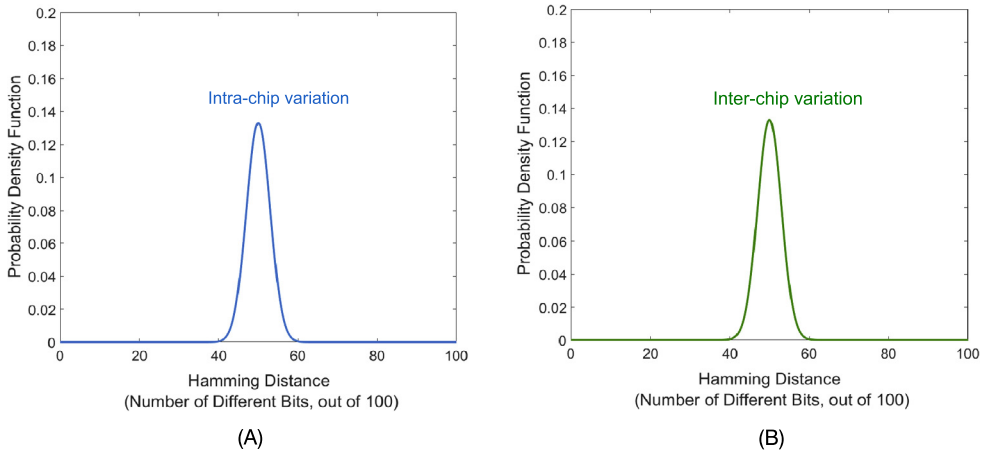

FIGURE 12.21

Hamming Distance for primitive instance X (A) and primitive type Y (B).

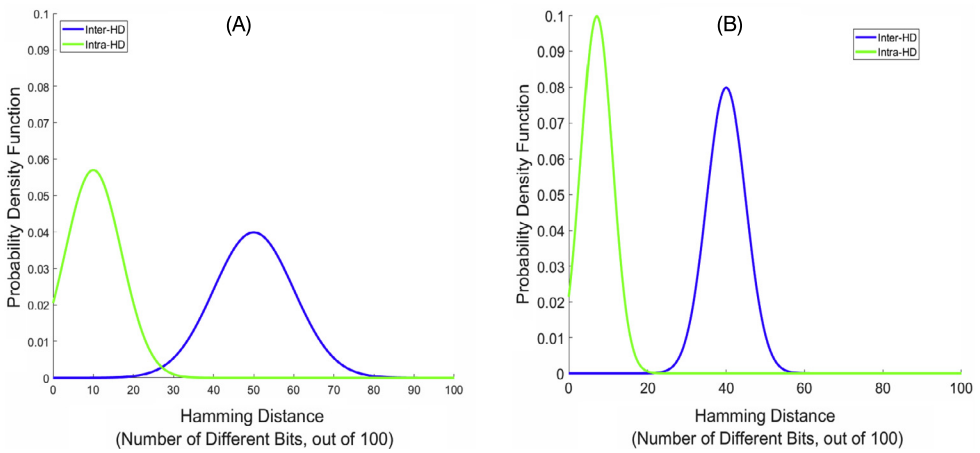

FIGURE 12.22

Inter- and intra-hamming distance for PUF M (A) and PUF N (B).

11. Consider two types of PUFs, namely "M" and "N". Their intra- and inter-hamming distances are presented in Fig. 12.22. Explain which type of the two PUFs serves better for the following applications:
(a) authentication,
(b) key generation.

REFERENCES

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

[2] B. Gassend, D. Clarke, M. Van Dijk, S. Devadas, Silicon physical random functions, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, ACM, pp. 148–160.

[3] G. Taylor, G. Cox, Behind Intel's new random-number generator, IEEE Spectrum 24 (2011).

[4] X. Zhang, M. Tehranipoor, Design of on-chip lightweight sensors for effective detection of recycled ICs, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22 (2014) 1016–1029.

[5] C.E. Shannon, A mathematical theory of communication, Bell System Technical Journal 27 (1948) 379–423.

[6] K.J. Kuhn, M.D. Giles, D. Becher, P. Kolar, A. Kornfeld, R. Kotlyar, S.T. Ma, A. Maheshwari, S. Mudanai, Process technology variation, IEEE Transactions on Electron Devices 58 (2011) 2197–2208.

[7] F. Rahman, A.P.D. Nath, D. Forte, S. Bhunia, M. Tehranipoor, Nano CMOS logic-based security primitive design, in: Security Opportunities in Nano Devices and Emerging Technologies, CRC Press, 2017, pp. 41–60.

[8] R. Kumar, Interconnect and noise immunity design for the Pentium 4 processor, in: Proceedings of the 40th Annual Design Automation Conference, ACM, pp. 938–943.

[9] R. Kumar, V. Kursun, Reversed temperature-dependent propagation delay characteristics in nanometer CMOS circuits, IEEE Transactions on Circuits and Systems II: Express Briefs 53 (2006) 1078–1082.

[10] S. Zafar, Y. Kim, V. Narayanan, C. Cabral, V. Paruchuri, B. Doris, J. Stathis, A. Callegari, M. Chudzik, A comparative study of NBTI and PBTI (charge trapping) in SiO2/HfO2 stacks with FUSI, TiN, Re Gates, in: VLSI Technology, 2006. Digest of Technical Papers. 2006 Symposium on, IEEE, pp. 23–25.

[11] D. Saha, D. Varghese, S. Mahapatra, On the generation and recovery of hot carrier induced interface traps: a critical examination of the 2D RD model, IEEE Electron Device Letters 27 (2006) 188–190.

[12] F. Rahman, D. Forte, M.M. Tehranipoor, Reliability vs. security: challenges and opportunities for developing reliable and secure integrated circuits, in: Reliability Physics Symposium (IRPS), 2016 IEEE International, IEEE, pp. 4C–6.

[13] C. Herder, M.-D. Yu, F. Koushanfar, S. Devadas, Physical unclonable functions and applications: a tutorial, Proceedings of the IEEE 102 (2014) 1126–1141.

[14] U. Rührmair, S. Devadas, F. Koushanfar, Security based on physical unclonability and disorder, in: Introduction to Hardware Security and Trust, Springer, 2012, pp. 65–102.

[15] C. Böhm, M. Hofer, Physical Unclonable Functions in Theory and Practice, Springer Science & Business Media, 2012.

[16] B.L.P. Gassend, Physical random functions, Ph.D. thesis, Massachusetts Institute of Technology, 2003.

[17] J. Guajardo, S.S. Kumar, G.-J. Schrijen, P. Tuyls, FPGA intrinsic PUFs and their use for IP protection, in: International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 63–80.

[18] U. Rührmair, H. Busch, S. Katzenbeisser, Strong PUFs: models, constructions, and security proofs, in: Towards Hardware-Intrinsic Security, Springer, 2010, pp. 79–96.

[19] A. Maiti, V. Gunreddy, P. Schaumont, A systematic method to evaluate and compare the performance of physical unclonable functions, in: Embedded Systems Design with FPGAs, Springer, 2013, pp. 245–267.

[20] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, A statistical test suite for random and pseudorandom number generators for cryptographic applications, Technical Report, DTIC Document, 2001.

[21] G. Marsaglia, Diehard: a battery of tests of randomness, See http://stat.fsu.edu/geo/diehard.html, 1996.

[22] G.E. Suh, S. Devadas, Physical unclonable functions for device authentication and secret key generation, in: Proceedings of the 44th Annual Design Automation Conference, ACM, pp. 9–14.

[23] D.E. Holcomb, W.P. Burleson, K. Fu, Power-Up SRAM state as an identifying fingerprint and source of true random numbers, IEEE Transactions on Computers 58 (2009) 1198–1210.

[24] A. Wild, T. Güneysu, Enabling SRAM-PUFs on Xilinx FPGAs, in: Field Programmable Logic and Applications (FPL), 2014 24th International Conference on, IEEE, pp. 1–4.

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

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

[27] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, S. Devadas, PUF modeling attacks on simulated and silicon data, IEEE Transactions on Information Forensics and Security 8 (2013) 1876–1891.

[28] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, U. Rührmair, The bistable ring PUF: a new architecture for strong physical unclonable functions, in: Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on, IEEE, pp. 134–141.
[29] U. Rührmair, J. Martinez-Hurtado, X. Xu, C. Kraeh, C. Hilgers, D. Kononchuk, J.J. Finley, W.P. Burleson, Virtual proofs of reality and their physical implementation, in: Security and Privacy (SP), 2015 IEEE Symposium on, IEEE, pp. 70–85.
[30] Y. Alkabani, F. Koushanfar, Active hardware metering for intellectual property protection and security, in: USENIX Security, Boston MA, USA, pp. 291–306.
[31] B. Sunar, W.J. Martin, D.R. Stinson, A provably secure true random number generator with built-in tolerance to active attacks, IEEE Transactions on Computers 56 (2007).
[32] M. Stipčević, Ç.K. Koç, True random number generators, in: Open Problems in Mathematics and Computational Science, Springer, 2014, pp. 275–315.
[33] B. Sunar, True random number generators for cryptography, in: Cryptographic Engineering, Springer, 2009, pp. 55–73.
[34] J. Von Neumann, 13. Various techniques used in connection with random digits, Applied Mathematics Series 12 (1951) 3.
[35] B. Preneel, Analysis and design of cryptographic hash functions, Ph.D. thesis, Citeseer, 1993.
[36] M.T. Rahman, K. Xiao, D. Forte, X. Zhang, J. Shi, M. Tehranipoor, TI-TRNG: technology independent true random number generator, in: Proceedings of the 51st Annual Design Automation Conference, ACM, pp. 1–6.
[37] T. Amaki, M. Hashimoto, T. Onoye, An oscillator-based true random number generator with jitter amplifier, in: Circuits and Systems (ISCAS), 2011 IEEE International Symposium on, IEEE, pp. 725–728.
[38] T. Amaki, M. Hashimoto, Y. Mitsuyama, T. Onoye, A worst-case-aware design methodology for noise-tolerant oscillator-based true random number generator with stochastic behavior modeling, IEEE Transactions on Information Forensics and Security 8 (2013) 1331–1342.
[39] B. Jun, P. Kocher, The Intel random number generator, Cryptography Research Inc., 1999, white paper.
[40] N. Stefanou, S.R. Sonkusale, High speed array of oscillator-based truly binary random number generators, in: Circuits and Systems, 2004. ISCAS'04, in: Proceedings of the 2004 International Symposium on, vol. 1, IEEE, pp. I–505.
[41] S.-H. Kwok, Y.-L. Ee, G. Chew, K. Zheng, K. Khoo, C.-H. Tan, A comparison of post-processing techniques for biased random number generators, in: IFIP International Workshop on Information Security Theory and Practices, Springer, pp. 175–190.
[42] L.R. Knudsen, Block Ciphers, in: Encyclopedia of Cryptography and Security, Springer, 2014, pp. 153–157.
[43] Sun crypto accelerator 6000: FIPS 140-2 non-proprietary security policy – Sun Microsystems, http://www.oracle.com/technetwork/topics/security/140sp1050-160928.pdf. (Accessed August 2018).
[44] M.M. Tehranipoor, U. Guin, D. Forte, Counterfeit integrated circuits, in: Counterfeit Integrated Circuits, Springer, 2015, pp. 15–36.
[45] X. Zhang, N. Tuzzio, M. Tehranipoor, Identification of recovered ICs using fingerprints from a light-weight on-chip sensor, in: Proceedings of the 49th Annual Design Automation Conference, ACM, pp. 703–708.
[46] U. Guin, X. Zhang, D. Forte, M. Tehranipoor, Low-cost on-chip structures for combating die and IC recycling, in: Proceedings of the 51st Annual Design Automation Conference, ACM, pp. 1–6.
[47] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. Van Dijk, S. Devadas, Extracting secret keys from integrated circuits, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13 (2005) 1200–1205.
[48] G. Hospodar, R. Maes, I. Verbauwhede, Machine learning attacks on 65nm Arbiter PUFs: accurate modeling poses strict bounds on usability, in: Information Forensics and Security (WIFS), 2012 IEEE International Workshop on, IEEE, pp. 37–42.
[49] M.T. Rahman, F. Rahman, D. Forte, M. Tehranipoor, An aging-resistant RO-PUF for reliable key generation, IEEE Transactions on Emerging Topics in Computing 4 (2016) 335–348.
[50] M.-D.M. Yu, S. Devadas, Secure and robust error correction for physical unclonable functions, IEEE Design & Test of Computers 27 (2010) 48–65.
[51] C.-E. Yin, G. Qu, Temperature-aware cooperative ring oscillator PUF, in: Hardware-Oriented Security and Trust, 2009. HOST'09, IEEE International Workshop on, IEEE, pp. 36–42.
[52] M.T. Rahman, D. Forte, F. Rahman, M. Tehranipoor, A pair selection algorithm for robust RO-PUF against environmental variations and aging, in: Computer Design (ICCD), 2015 33rd IEEE International Conference on, IEEE, pp. 415–418.
[53] C. Helfmeier, C. Boit, D. Nedospasov, J.-P. Seifert, Cloning physically unclonable functions, in: Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on, IEEE, pp. 1–6.
[54] A.T. Markettos, S.W. Moore, The frequency injection attack on ring-oscillator-based true random number generators, in: Cryptographic Hardware and Embedded Systems-CHES 2009, Springer, 2009, pp. 317–331.

[55] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Poucheret, B. Robisson, P. Maurine, Contactless electromagnetic active attack on ring oscillator based true random number generator, in: International Workshop on Constructive Side-Channel Analysis and Secure Design, Springer, pp. 151–166.
[56] D. Schellekens, B. Preneel, I. Verbauwhede, FPGA vendor agnostic true random number generator, in: Field Programmable Logic and Applications, 2006. FPL'06. International Conference on, IEEE, pp. 1–6.
[57] S. Robson, B. Leung, G. Gong, Truly random number generator based on a ring oscillator utilizing last passage time, IEEE Transactions on Circuits and Systems II: Express Briefs 61 (2014) 937–941.
[58] A. Chen, Emerging nonvolatile memory (NVM) technologies, in: Solid State Device Research Conference (ESSDERC), 2015 45th European, IEEE, pp. 109–113.
[59] F. Rahman, A.P.D. Nath, S. Bhunia, D. Forte, M. Tehranipoor, Composition of physical unclonable functions: from device to architecture, in: Security Opportunities in Nano Devices and Emerging Technologies, CRC Press, 2017, pp. 177–196.
[60] H.-S.P. Wong, S. Raoux, S. Kim, J. Liang, J.P. Reifenberg, B. Rajendran, M. Asheghi, K.E. Goodson, Phase change memory, Proceedings of the IEEE 98 (2010) 2201–2227.
[61] W. Zhang, T. Li, Characterizing and mitigating the impact of process variations on phase change based memory systems, in: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, ACM, pp. 2–13.
[62] L. Zhang, Z.H. Kong, C.-H. Chang, PCKGen: a phase change memory based cryptographic key generator, in: Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, IEEE, pp. 1444–1447.
[63] L. Zhang, Z.H. Kong, C.-H. Chang, A. Cabrini, G. Torelli, Exploiting process variations and programming sensitivity of phase change memory for reconfigurable physical unclonable functions, IEEE Transactions on Information Forensics and Security 9 (2014) 921–932.
[64] K. Kursawe, A.-R. Sadeghi, D. Schellekens, B. Skoric, P. Tuyls, Reconfigurable physical unclonable functions-enabling technology for tamper-resistant storage, in: Hardware-Oriented Security and Trust, 2009. HOST'09, IEEE International Workshop on, IEEE, pp. 22–29.
[65] G.S. Rose, N. McDonald, L.-K. Yan, B. Wysocki, A write-time based memristive PUF for hardware security applications, in: Proceedings of the International Conference on Computer-Aided Design, IEEE Press, pp. 830–833.
[66] R. Liu, H. Wu, Y. Pang, H. Qian, S. Yu, A highly reliable and tamper-resistant RRAM PUF: design and experimental validation, in: Hardware Oriented Security and Trust (HOST), 2016 IEEE International Symposium on, IEEE, pp. 13–18.
[67] J.S. Meena, S.M. Sze, U. Chand, T.-Y. Tseng, Overview of emerging nonvolatile memory technologies, Nanoscale Research Letters 9 (2014) 526.
[68] L. Zhang, X. Fong, C.-H. Chang, Z.H. Kong, K. Roy, Highly reliable memory-based Physical Unclonable Function using Spin-Transfer Torque MRAM, in: Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, IEEE, pp. 2169–2172.
[69] J. Das, K. Scott, S. Rajaram, D. Burgett, S. Bhanja, MRAM PUF: a novel geometry based magnetic PUF with integrated CMOS, IEEE Transactions on Nanotechnology 14 (2015) 436–443.
[70] S. Ghosh, Spintronics and security: prospects, vulnerabilities, attack models, and preventions, Proceedings of the IEEE 104 (2016) 1864–1893.