INVITED PAPER

Physical Unclonable Functions and Applications: A Tutorial

This paper is a tutorial on ongoing work in physical-disorder-based security, security analysis, and implementation choices.

By Charles Herder, Meng-Day (Mandel) Yu, Farinaz Koushanfar, and Srinivas Devadas, *Fellow IEEE*

ABSTRACT | This paper describes the use of physical unclonable functions (PUFs) in low-cost authentication and key generation applications. First, it motivates the use of PUFs versus conventional secure nonvolatile memories and defines the two primary PUF types: "strong PUFs" and "weak PUFs." It describes strong PUF implementations and their use for low-cost authentication. After this description, the paper covers both attacks and protocols to address errors. Next, the paper covers weak PUF implementations and their use in key generation applications. It covers error-correction schemes such as pattern matching and index-based coding. Finally, this paper reviews several emerging concepts in PUF technologies such as public model PUFs and new PUF implementation technologies.

KEYWORDS | Arbiter; index-based coding; pattern matching; physical unclonable function (PUF); public model PUFs; ring oscillator; SRAM; unclonable

I. INTRODUCTION

Mobile and embedded devices are becoming ubiquitous, interconnected platforms for everyday tasks. Many such tasks require the mobile device to securely authenticate and be authenticated by another party and/or securely handle private information. Indeed, smartphones have become a unified platform capable of conducting financial transactions, storing a user's secure information, acting as an authentication token for the user, and performing many other secure applications. The development of powerful mobile computing hardware has provided the software flexibility to enable convenient mobile data processing. However, comparable mobile hardware security has been slower to develop. Due to the inherent mobility of such devices, the threat model must include use cases where the device operates in an untrusted environment and the adversary has a degree of physical access to the system.

The current best practice for providing such a secure memory or authentication source in such a mobile system is to place a secret key in a nonvolatile electrically erasable programmable read-only memory (EEPROM) or battery-backed static random-access memory (SRAM) and use hardware cryptographic operations such as digital signatures or encryption. This approach is expensive both in terms of design area and power consumption. In addition, such nonvolatile memory is often vulnerable to invasive attack mechanisms. Protection against such attacks requires the use of active tamper detection/prevention circuitry which must be continually powered.

Physical unclonable functions (PUFs) are a promising innovative primitive that are used for authentication and secret key storage without the requirement of secure EEPROMs and other expensive hardware described above [7], [34]. This is possible, because instead of storing secrets in digital memory, PUFs derive a secret from the physical characteristics of the integrated circuit (IC). For example, this paper will discuss a PUF that uses the innate manufacturing variability of gate delay as a physical characteristic from which one can derive a secret. This approach is

Manuscript received September 3, 2013; accepted April 8, 2014. Date of publication May 30, 2014; date of current version July 18, 2014.
C. Herder and **S. Devadas** are with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Department of Electrical Engineering and Computer Science (EECS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: cherder@mit.edu).
M.-D. Yu is with Verayo, Inc., San Jose, CA 95129 USA and also with the Computer Security and Industrial Cryptography (COSIC) research group, KU Leuven, Leuven-Heverlee B-3001, Belgium.
F. Koushanfar is with the Adaptive Computing and Embedded Systems Lab (ACES), Department of Electrical and Computer Engineering (ECE), Rice University, Houston, TX 77005 USA.
Digital Object Identifier: 10.1109/JPROC.2014.2320516

advantageous over standard secure digital storage for several reasons.

- PUF hardware uses simple digital circuits that are easy to fabricate and consume less power and area than EEPROM/RAM solutions with antitamper circuitry. In addition, simple PUF applications do not require expensive cryptographic hardware such as the secure hash algorithm (SHA) or a public/private key encryption algorithm.
- Since the "secret" is derived from physical characteristics of the IC, the chip must be powered on for the secret to reside in digital memory. Any physical attack attempting to extract digital information from the chip, therefore, must do so while the chip is powered on.
- Invasive attacks are more difficult to execute without modifying the physical characteristics from which the secret is derived. Therefore, continually powered active antitamper mechanisms are not required to secure the PUF [4].
- Nonvolatile memory is more expensive to manufacture. EEPROMs require additional mask layers, and battery-backed RAMs require an external always-on power source.

A PUF is based on the idea that even though the mask and manufacturing process is the same among different ICs, each IC is actually slightly different due to normal manufacturing variability. PUFs leverage this variability to derive "secret" information that is unique to the chip (a silicon "biometric"). In addition, due to the manufacturing variability that defines the secret, one cannot manufacture two identical chips, even with full knowledge of the chip's design. PUF architectures exploit manufacturing variability in multiple ways. In addition to gate delay, architectures also use the power-on state of SRAM, threshold voltages, and many other physical characteristics to derive the secret.

This paper discusses the most popular PUF architectures. After defining a conceptual model for a PUF, this paper defines protocols to address two primary applications: strong authentication and cryptographic key generation. It then provides a case study on the most popular approach to each of these respective applications. It focuses on the experimental results from extensively tested realizations of actual PUF architectures that currently implement these protocols in real-world applications. Finally, this paper provides a perspective on future research relating to PUFs by briefly discussing current open problems as well as the latest proposed solutions.

A. Previous Work: Unique Objects

Although many of the architectures that integrate PUFs into existing IC technology are new, it should be noted that the concepts of unclonability and uniqueness have been used extensively in the past for other applications [13]. For example, "unique objects" are well defined as objects with a unique set of properties (a "fingerprint") based on the unique disorder of the object [34]. This fingerprint should be stable over time and robust to other environmental conditions and to readout. Further, it must be "unclonable" in the sense that the cost to engineer and manufacture another object with the same fingerprint must be prohibitively expensive or impractical using known manufacturing techniques (including by the original manufacturer).

One example of early usage of unique objects for security was proposed for the identification of nuclear weapons during the Cold War [9]. One would spray a thin coating of randomly distributed light-reflecting particles onto the surface of the nuclear weapon. Since these particles are randomly distributed, the resulting interference pattern after being illuminated from various angles is unique and difficult to reproduce.

Therefore, immediately after being applied, each interference pattern would be measured as a "signature" and stored in a secure database. A weapon could then be identified at any later time by re-illuminating the surface and comparing the interference pattern against the measured interference pattern. At the time, it was presumed to be infeasible to reproduce such an interference pattern even if an adversary knew the illumination angle(s) and the resulting pattern.

II. TYPES OF PUFs

The two primary applications of PUFs are for: 1) low-cost authentication; and 2) secure key generation. These two applications have resulted from the fact that PUFs designed during the past decade have mostly fallen into two broad categories. These categories are described as "strong PUFs" and "weak PUFs." Strong PUFs are typically used for authentication, while weak PUFs are used for key storage.

Each PUF can be modeled as a black-box challenge–response system. In other words, a PUF is passed an input challenge c, and returns a response $r = f(c)$, where $f(\cdot)$ describes the input/output relations of the PUF. The black-box model is appropriate here, because the internal parameters of $f(\cdot)$ are hidden from the user since they represent the internal manufacturing variability that the PUF uses to generate a unique challenge–response set. Such parameters would include the variability of a circuit's internal gate delay as described in the Introduction. PUF security relies on the difficulty of measurement or estimation of these parameters as well as the difficulty of manufacturing two chips with the same set of parameters.

The fundamental difference between weak and strong PUFs is the domain of $f(\cdot)$, or informally, the number of unique challenges c that the PUF can process. A weak PUF can only support a small number of challenges (in some cases only a single challenge). A strong PUF can support a large enough number of challenges such that complete determination/measurement of all challenge–response pairs (CRPs) within a limited timeframe is not feasible.

A. Weak PUF Model

The first class of PUFs leveraging manufacturing variability are weak PUFs [also known as physically obfuscated keys (POKs)]. These PUFs can be thought of as PUFs that directly digitize some "fingerprint" of the circuit. This direct measurement results in a digital signature that can be used for cryptographic purposes. Because the fingerprint signature remains largely invariant, this means that the PUF can only be interrogated by one or a small number of challenges. In the above black-box description, this corresponds to $f(\cdot)$ having a domain of one or only a small number of inputs. Correspondingly, $f(\cdot)$ will also have a very small range, as a given challenge should always result in the same response (ignoring noise, which is considered later). One can clearly use several instances of the above black box to support more CRPs or response bits. However, this is still considered a weak PUF, because the number of responses is linearly related to the number of components subject to manufacturing variation. Explicitly stated, weak PUFs have the following properties:

- a small number of CRPs (linearly related to the number of components whose behavior depends on manufacturing variation);
- response is stable and robust to environmental conditions and multiple readings so that a challenge always yields the same response;
- responses are unpredictable and depend strongly on the innate manufacturing variability of the device;
- it is impractical to manufacture two devices with the same physical fingerprint.

An example weak PUF is the power-on state of an SRAM. Although a SRAM cell is symmetric, manufacturing variability will give each cell a tendency toward a logical "1" or "0" at power-on. This variability is random across the entire SRAM, giving it a unique fingerprint on power-on that can be identified. In this case, if the "response" consists of the entire SRAM state at power-on, the notion of a "challenge" is not useful, as there is only one possible "challenge": powering on the SRAM. The output signature is always the same (ignoring noise). One can allow for more output bits by increasing the size of the SRAM, but the response space is still linearly related to the number of components subject to manufacturing variation (each SRAM cell). The SRAM is an extreme example of a weak PUF in the sense that it only has one "CRP."

Note that since weak PUFs in general have only a small number of CRPs, these pairs must be kept secret. If a weak PUF only has one CRP, and it is revealed, then any device can emulate the PUF. For this reason, weak PUFs are well suited for use in key derivation processes. The PUF provides the randomness and secure storage, and the secret key (derived from the PUF's response bits) is never revealed during operation.

Once the key is recovered by the PUF (this typically requires error correction), any cryptographic process may follow. For example, the weak PUF output may be used as the key in a keyed-hash message authentication code (HMAC) challenge–response sequence. In addition, the output may be used as a secret key to encrypt/decrypt data on the device.

B. Strong PUF Model

Strong PUFs differ from weak PUFs in that a strong PUF can support a large number of CRPs. As a result, a strong PUF can be authenticated directly without using any cryptographic hardware. The requirements for a strong PUF are:

- large enough challenge–response space such that an adversary cannot enumerate all CRPs within a certain fixed time (ideally, exponential in the number of challenge bits);
- responses stable to environment, multiple readings;
- an adversary given a polynomial-sized sample of adaptively chosen CRPs cannot predict the response to a new, randomly chosen challenge;
- not feasible to manufacture two PUFs with the same responses;
- the readout only reveals the response $r = f(c)$ and no other data about the internal functionality of the PUF.

It should be noted that a weak PUF can provide authentication capabilities if the weak PUF is paired with crypto hardware supporting HMAC or similar authentication processes (note that HMAC and others support exponentially sized challenge–response spaces but their use requires 100% response stability and, therefore, error-correction logic). It should also be noted that the security models for weak and strong PUFs differ. The output of a weak PUF must be kept private, while a strong PUF's responses do not have the same restriction.

The strong PUF has the additional requirement of readout access restriction [only $r = f(c)$ is revealed] due to this difference in security models. In addition, to prevent total enumeration of the strong PUF, one must also consider the readout time of the PUF in conjunction with the number of CRPs. A faster PUF response allows for faster enumeration of all PUF CRPs. Since a weak PUF provides a secret key, the surrounding digital cryptographic hardware is responsible for limiting access to the weak PUF output. However, the strong PUF does not require the use of additional crypto hardware to provide authentication services, and therefore must itself prevent unauthorized access into its own internal structure.

C. Error Correction

Both weak and strong PUFs rely on analog physical properties of the fabricated circuit to derive secret information. Naturally, these analog properties have noise and variability associated with them. Consider the example introduced in Section I that uses gate delay. This delay depends on temperature, supply voltage, and other environmental parameters. As these parameters vary, so does

the "digital fingerprint" measured by the PUF. If the parameters vary too much, the digital key (for the weak PUF) or response (for the strong PUF) will change, and the crypto operation will fail.

The first mechanism to mitigate such effects is to use differential design techniques to cancel out first-order environmental dependencies. Using the gate delay example, typical PUFs using this effect will not measure a single gate's delay, but rather the difference between two identically designed, but distinct gates on a die. In this way, any environmental factor should affect each gate equally.

Although differential design methodologies do improve reliability, noise is still a factor in PUF design. Even in optimal environmental conditions, noise will result in one or several of the output bits of the PUF being incorrect for any given challenge. Therefore, modern PUF designs employ multiple error-correction techniques to correct these bits, improving reliability. However, many of these error-correcting techniques have been shown to leak bits of the secret key, since they require the computation and public storage of syndrome bits. As such, an excess number of PUF bits are generated and then downmixed to produce a full entropy key.

In addition to standard error-correction techniques, PUFs also use soft-decision coding. This coding technique takes advantage of the reliability information of a given response bit to improve error-correction performance. This reliability information can be obtained from repeated PUF response readings in the case of SRAM PUFs, or the magnitude of frequency difference values in the case of ring-oscillator PUFs. Both of these error-correcting techniques will be discussed further in the context of the weak and strong PUF examples to be presented in Sections IV-E, VI-B, VI-C1, and VI-D1.

III. EXAMPLE STRONG PUF ARCHITECTURES

A. Optical PUF

One of the first implementations of a strong PUF was constructed by Pappu *et al.* in 2001 [30]. The paper terms the device a "physical one-way function," but the functionality is identical to that of a strong PUF. Pappu *et al.* describe a device with three primary components: 1) a laser directed along the *Z*-axis that can be moved in the *XY*-plane and whose polarization can be modified; 2) a stationary scattering medium that sits along the path of the laser beam; and 3) an imaging device that records the output "speckle" pattern of laser light exiting the scattering medium.

In this device, the input challenge is a laser *XY* location and polarization, and the response is the associated speckle pattern. The speckle pattern is strongly dependent on the input location/polarization because multiple scattering events occur inside the scattering medium. In the implementation by Pappu *et al.*, the scattering medium consisted of a large number of randomly positioned 100-μm silica spheres suspended in a hardened epoxy. Each sphere acts as a small lens, refracting individual rays of light as they move through the scattering block. The overall size of the scattering block was on the order of 1-mm thickness. Therefore, even a relatively simple optical path must encounter $\sim$10 spheres as it travels through the scattering block.

All of these paths then are focused into an image on the detector. It is intuitively true that each of these paths will be very sensitive to input coordinates. Studies on speckle patterns produced by reflection/transmission by rough surfaces have found this to be true both experimentally and mathematically [2]. In addition, the speckle pattern is also sensitive to the internal structure of the scattering block. Therefore, it is difficult to fabricate two blocks with identical speckle patterns. Finally, due to the complex nature of the physical interactions, it is difficult to model the internal dynamics of the scattering medium. It is also difficult to use the output speckle to determine properties of the scattering block (such as the locations of the silica spheres).

These assumptions, while not strictly based on known computationally difficult problems, can be trusted to be difficult due to the fact that ray-tracing electromagnetic simulation is a well-studied field with established theoretical models and best practices. One can make the statement that if an adversary were able to break the above optical PUF by efficiently reproducing the physical device, modeling the entire scattering block, or discovering the sphere locations via observation of the speckle pattern, this would represent a major advancement in the field of ray-based models of electromagnetic simulation. It is for this reason that Pappu *et al.* described this optical PUF as a "physical one-way function."

B. Arbiter PUF

Although the capabilities of the above optical PUF are significant, and they represented a significant step forward in the understanding and construction of PUFs, the practical applications are limited due to the macroscopic optical nature. This limitation stemmed from two properties.

First, the actual unclonable object (the scattering block) was separate from the measurement apparatus (the imaging device). As a result, the trust gained from authenticating an optical PUF is more limited. In a practical use case, the objective of authenticating the PUF is typically to authenticate the associated processor to which it is connected. However, since the optical PUF is separated from the digital measurement circuitry, an optical PUF as described by Pappu *et al.* designed to authenticate processor A can easily be detached from processor A and connected to processor B. Processor B could then authenticate itself as processor A. It is more desirable for the digital measurement apparatus to be integrated in with the

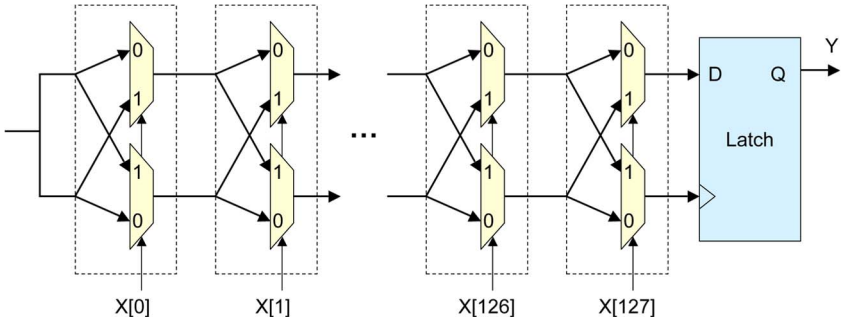

Fig. 1. *Arbiter PUF circuit. The circuit creates two delay paths with the same layout length for each input X, and produces an output Y based on which path is faster.*

PUF such that the PUF is not separable from the device it is used to authenticate.

Second, since both key generation and authentication applications use integrated electronics, a more practical PUF would have the same properties as the optical PUF and simultaneously be integrated directly with a conventional complementary metal–oxide–semiconductor (CMOS) process. This integration would be such that the IC could not be separated from the PUF.

Silicon implementations of strong PUFs were described in the paper by Gassend *et al.* beginning in 2002 using manufacturing variability in gate delay as the source of unclonable randomness [7]. In one implementation, a race condition is established in a symmetric circuit. This is shown in Fig. 1. An input edge is split to two multiplexors (muxes). Depending on the input challenge bits ($X[0]$–$X[127]$), this path will vary. Although the layout is identical (propagation time should be the same for each edge no matter what challenge bits are chosen), manufacturing variability in the gate delay of each mux will result in one edge arriving at the latch first, and the latch acts as the "arbiter." The output will, therefore, depend on the challenge bits.

In Fig. 1, there are 128 challenge bits and one response bit. Of course, one typically operates multiple identical circuits in parallel to achieve 128 response bits. In this way, the arbiter PUF can be scaled to an almost arbitrary number of CRPs.

The security of the arbiter PUF, like the optical PUF before it, is based on assumptions regarding manufacturing capabilities and ultimately metrology of the individual gate delays. Because the design is symmetric, the design does not contain any "secret" information. An adversarial manufacturer that has the PUF design cannot manufacture a duplicate PUF, because the behavior of the PUF is defined by the inherent variability in the manufacturing process. Even the original manufacturer of the PUF could not produce two identical PUFs, since this would require a significant improvement in manufacturing control.

The second security assumption is that the individual gate delays are difficult to measure directly. It assumes that an invasive attacker would have difficulty in extracting the individual delays even with physical access. This assumption is based on the hypothesis that an invasive attacker would destroy the gate delay properties using his/her measurement techniques.

The last security assumption is that given a set of CRPs from an arbiter PUF, an adversary could not calculate the internal delays of the gates. For the architecture described above, this is actually not the case. Each delay is independent from all other delays, and the delays add linearly. As a result, one can use standard linear system analysis to intelligently gather data about the gate delays from the response bits. In fact, it can be shown that this system breaks after only a small number of challenges [17]. This problem can be resolved by several approaches proposed by Gassend *et al.* and described in Section IV-D.

Finally, in both optical and arbiter PUF architectures, it should be noted that environmental factors play a significant role. For the optical PUF, calibration of the input location is a concern. In the case of the arbiter PUF, one can easily recognize that environmental variations such as temperature, supply voltage, aging, and even random noise will affect the delay of each edge through the arbiter PUF. In addition, if the delays are close enough, the latch's setup time will be violated, potentially resulting in an unpredictable output. As a result, the response bits may not be stable. In this case, error-correcting techniques are used to increase the stability of the PUF while maintaining its security. Techniques for accomplishing this will be covered in Section IV-E. Although key generation has zero error tolerance, PUF authentication usually incorporates an allowable error threshold, thereby decreasing the stability requirement, and often obviating the need for error correction.

IV. LOW-COST AUTHENTICATION: STRONG PUFS

The strong PUF architectures described above are typically associated with the application of low-cost authentication.

In this case, a strong PUF will replace the secure memory and crypto hardware on an embedded device and is used to securely identify the device to a server. Because the PUF does not require secure nonvolatile memory, antitamper circuitry, or additional supporting crypto acceleration hardware, a PUF-based solution requires less area, power, and mask layers than a traditional approach to secure authentication.

A. Authentication Protocol

As described previously, the strong PUF receives a challenge and generates a response. However, the requirements of a strong PUF state that an adversary provided with polynomial CRPs should not be able to predict the response to a new challenge.

Although this is a desirable property, it also presents a usage problem. Since the PUF acts as a "black box," even the authentication server only has access to previously observed CRPs and, therefore, also cannot predict the response to a new challenge.

Therefore, the protocol for using PUFs is significantly different than most public/private key cryptographic systems. Consider a server authenticating a client.

1) PUF is manufactured.
2) Server obtains access to PUF and generates a table of CRPs. These pairs are stored in internal secret storage.
3) PUF is given to the client.
4) The client submits a request to the server to authenticate.
5) Server picks a known CRP and submits the challenge to the client.
6) The client runs the challenge on the PUF, returns the response to the server.
7) Server checks to see that the response is correct and marks the CRP as used.

Because the server cannot predict the PUF behavior, it must internally store CRPs to be used later. Each CRP must be used only once. Therefore, the server must either store enough CRPs so that it will not run out, or it must periodically "recharge" the table by establishing secure communication with an authenticated client and requesting responses to new challenges. To address the CRP table scalability problem, newer protocols based on storage of a compact model for PUF have emerged. A brief discussion of these protocols is included in Section VII-A.

Note that each client PUF will have unique CRPs, and therefore can be individually authenticated. In addition, the server must store tables of CRPs for each of the clients to be authenticated.

B. Arbiter PUF Topologies

The initial implementation of silicon PUFs had known security issues due to the fact that the delays were linearly added to produce the resultant response bit [8]. As a result, they could be learned with relative ease. This issue naturally led to the introduction of other "nonlinear" effects to make such modeling attacks more difficult. These efforts included XOR arbiter PUFs, lightweight secure PUFs, and feedforward arbiter PUFs [8], [16], [17], [22], [39].

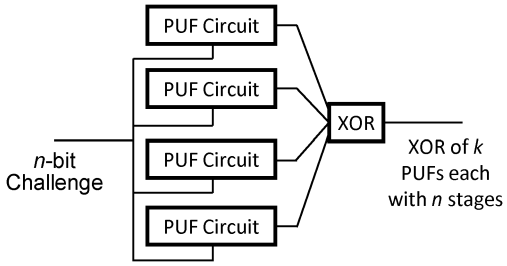

Fig. 2. *Four individual arbiter PUF circuits with nonlinearities introduced via XOR'ing their outputs.*

In a XOR arbiter, multiple arbiter PUF outputs are XOR'ed to form a single response bit. This is shown in Fig. 2. These structures have shown greater resilience against machine learning attacks [24], [35]. Recent studies have demonstrated the vulnerability of the XOR arbiters to a combination of machine learning and side-channel attacks [19], [36]. Developing methods to suppress the side channels could help in alleviating this vulnerability.

C. Arbiter PUF Implementation

The arbiter PUF was implemented and studied by Devadas *et al.* as a part of a radio-frequency identification (RFID) IC fabricated in 0.18-μm technology [4]. In this implementation, a single arbiter PUF is implemented on-chip. This primitive has an input challenge of 64 bits and a single output bit. To construct a k-bit response, a linear feedback shift register (LFSR) is used to generate a pseudorandom sequence based on the input challenge. The PUF is then evaluated k times using k different bit vectors from this larger pseudorandom sequence. Finally, to prevent learning attacks on the PUF output bits, an additional scrambling routine is performed.

In this implementation, area and power consumption represented a major design constraint. Therefore, the above PUF implementation with only a single arbiter is used. As a result, the majority of the silicon area is consumed by standard RFID components (RFID front–end, one-time programmable memory, digital logic). The PUF and associated LFSR have been implemented in less than 0.02 mm^2 using 0.18-μm fabrication technology. In addition, the PUF only consumes dynamic power during evaluation, and the power consumption was shown to be small with respect to the power stored on the RFID chip.

In order to understand the PUF's utility as an identification and authentication source, intra-PUF and inter-PUF variation are defined as follows [7].

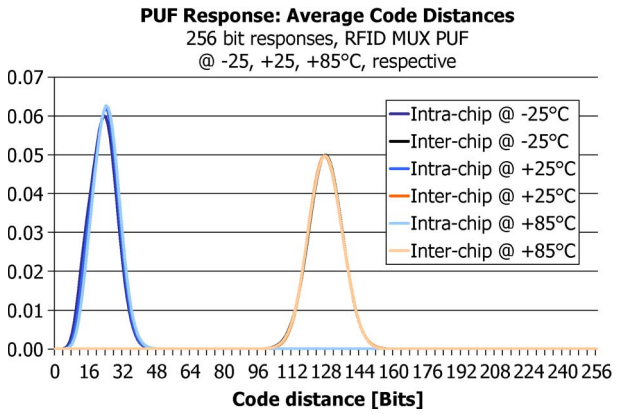

Fig. 3. ***Code distance distribution for 256-bit PUF responses.***

- Intra-PUF variation: Defined as the number of bits in a PUF response that vary when an identical challenge is repeatedly queried on a given PUF device in a changing environment. This variation is due to this environmental change as well as statistical noise. As a result, it is commonly represented in the form of a statistical distribution. Intra-PUF variation is a measure of the reproducibility of responses from an individual PUF circuit.
- Inter-PUF variation: Defined as the number of bits in a PUF response that vary between different devices for a set of shared challenges. This is due to differences between the physical ICs and is also commonly represented in the form of a statistical distribution. The inter-PUF variation is a measure of the uniqueness of an individual PUF circuit.

For the application of secure authentication, intra-PUF variation should be low (ideally 0%) so that the PUF can be verified. On the other hand, inter-PUF variation should be high (ideally 50% on average) so that two separate PUFs have a maximally decorrelated responses. This behavior has been observed for arbiter PUFs, as shown in Fig. 3. Note that in this figure, the two interchip variations are roughly 50% (128 bits out of 256), and the intrachip varia, tions are much smaller (∼10%). Clearly, the implemented PUF has the desired properties. (However, it is also clear that error correction must be used to compensate for the ∼10% intrachip variation for key generation applications.)

D. Attacks on Arbiter PUFs

The security of a strong PUF depends on several factors (if any one of these factors is compromised, the security of the PUF itself is also compromised):

- difficulty of measurement of PUF internal parameters (only CRPs can be measured);
- difficulty of manufacturing "clones";
- difficulty of predicting PUF behavior based on past CRPs.

For arbiter PUFs, an area of continued active research is in ensuring the difficulty of predicting PUF behavior based on past CRPs. In the RFID IC example, this issue is addressed by "scrambling" the output bits of the PUF. In other words, the output bits of the PUF pass through some digital circuit that obfuscates the linear behavior of the PUF before being returned as a response. The simple arbiter PUF implementation without output postprocessing is linear and, therefore, significantly easier to predict. (We note that this scrambling will increase the noise in the output bits and, therefore, has to be done carefully.)

Studies have been performed by Majzoobi *et al.* [23] and Rührmair *et al.* [35] using machine learning to predict the behavior of PUFs after a certain number of CRPs have been observed. In these studies, learning attacks were perpetrated on simple arbiter PUFs, feedforward arbiter PUFs, arbiter PUFs with output XOR'ing, "lightweight secure" arbiter PUFs (these use a more complicated output postprocessing circuit, but are based on principles similar to output XOR'ing).

The linear behavior of simple arbiter PUFs was clearly demonstrated, as a learning algorithm predicted the behavior of a 64-bit arbiter PUF with 95% accuracy after observing 640 CRPs (the model training time on a standard PC was 0.01 s). To predict with 99.9% accuracy, 18 050 CRPs are needed to be observed (model training time of 0.6 s). This demonstrates that the behavior of a simple arbiter PUF can be learned efficiently [35].

In addition, PUFs with 64-bit challenges and 128-bit challenges were tested. It was found that the number of CRPs and model learning complexity scaled as expected with the input challenge size. This proved to be true not just for simple arbiter PUFs, but also for the nonlinear implementations discussed below.

The arbiter PUF with output XOR'ing, on the other hand, is able to make the problem intractable to such learning attacks. Rührmair *et al.* identified an exponential dependence on the number of output XORs and the required complexity of the learning attack. This is due in part to the actual computational complexity of learning the model. It is also due to the fact that this model learning process is looking for a global optimum on a nonconvex parameter space. As a result, the learning algorithm has some probability of failure that scales inversely with the size of the training set (the number of observed CRPs). If the algorithm fails, it must be restarted again with different parameters.

It was identified that an arbiter PUF with a 512-bit challenge and eight output XORs would defeat the machine learning approaches used by Rührmair *et al.* in 2010. The postprocessing scheme used in "lightweight secure" PUFs proved to have the same exponential dependence with a similar complexity requirement.

A newer set of attacks leveraging both machine learning and side-channel information has recently emerged [19], [36]. It has been shown that by coordinated

application of timing or power side-channel analysis and adapted machine learning techniques, very efficient attacks can be performed, i.e., attacks that use linearly many CRPs and low degree polynomial computation times. The practical viability of the combined attacks has been demonstrated by machine learning experiments on numerically simulated CRPs. This work has shown that XOR arbiter PUF and lightweight PUFs have to be implemented in such a manner that power side channels are protected, else PUFs can be easily cloned.

One key consideration in studying the complexity of PUFs is stability (described as "intra-PUF variation" in the RFID IC example). This will be discussed further in Section IV-E. However, it should be noted that although output XOR'ing has an exponential effect on modeling complexity, it also has an exponential effect on decreasing stability. In doing so, it decreases the effectiveness of a PUF in an actual authentication environment and simultaneously decreases the accuracy requirement of an attack model, as the greater intrinsic PUF error must be tolerated by the authentication protocol.

The task of identifying an approach to exponentially increase model complexity while only having a polynomial effect decreasing PUF stability is an area of active research.

E. Error Correction Versus Tolerance

In the RFID IC example, random noise contributes to the PUF stability being roughly 90%, i.e., the intra-PUF variation is ~10%. In addition, this stability worsens when the temperature changes.

In perhaps the earliest reference to error correction in silicon PUFs, Gassend mentioned the use of 2-D Hamming codes [6]. Suh suggested the use of Bose–Chaudhury–Hochquenghen (BCH) codes—more specifically the BCH (255,63, $t = 30$) code [38]. In this case, the PUF generates 255 bits, but the code exposes 192 syndrome bits publicly, so the actual security of the system is at most 63 bits. This error corrects at most 30 errors out of 255 bits. This corresponds to a PUF with ~88% stability.

In low-cost authentication applications, the host instead gives a certain error tolerance or multiple authentication opportunities to a PUF before rejecting the PUF as invalid. Error tolerance is typically the preferred methodology. Using the arbiter PUF described with code distances shown in Fig. 3, false positive/negative identification probabilities were measured for specific allowed error tolerances. These data are summarized in Fig. 4.

Majzoobi *et al.* showed a strong PUF remote authentication protocol which does not require traditional error correction while also being secure against machine learning attacks [25], [31]. The protocol is inspired by the pattern matching techniques described in Section VI-B but is used for authentication rather than secret key generation. A large class of machine learning attacks are shown to fail when applied to this protocol.

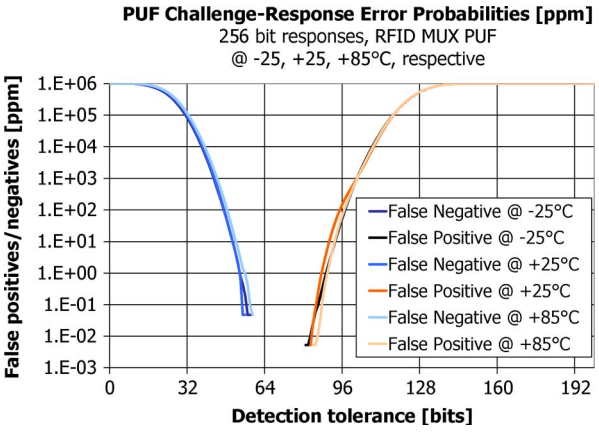

Fig. 4. *False positives and negatives for strong PUF operation with a given error tolerance.*

V. EXAMPLE WEAK PUF ARCHITECTURES

A. Ring-Oscillator PUF

In addition to arbiter PUFs, the manufacturing variability intrinsic to circuit gate delay can also be used to instantiate a "ring-oscillator PUF" [39]. This PUF architecture contains N identically designed ring oscillators synthesized onto a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC).

Due to the variation in delay of the inverters in the ring oscillator, each will have a slightly different frequency. The frequencies of two oscillators are measured and compared to reveal one of the PUF output bits. If there are N oscillators, there are $N(N-1)/2$ possible pairings. However, the number of output bits is limited due to correlations (if ring oscillator A is faster than B, and B is faster than C, then clearly A is faster than C). For N oscillators, there is a specific ordering of fastest to slowest. If the oscillators are truly identical and manufacturing variation dominates, then each of these $N!$ orderings is equally likely. Therefore, there are a maximum of $\log(N!)$ bits that can be extracted from the PUF.

Note that the ring-oscillator PUF is a weak PUF, since there are a limited number of "challenge bits" that can configure the PUF's operation. Once fabricated, the ring oscillators' frequency is set, so the output bits of the PUF will always remain constant.

Because the ring-oscillator PUF measures differences in gate delay like the arbiter PUF, the ring-oscillator PUF is susceptible to the same set of environmental variations and noise sources. Therefore, error correction will be equally important in this application.

One approach that can be taken immediately to mitigate potential errors is to recognize that oscillators that are "close" in frequency have much greater likelihood of causing an output error than oscillators that are "far apart"

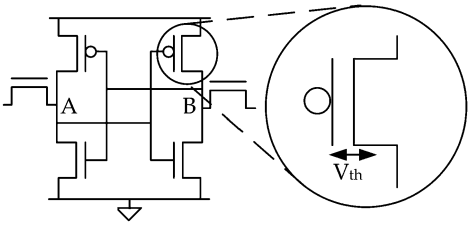

Fig. 5. *SRAM cell [12]. V_{th} differences in the transistors result in the SRAM powering up in either a logic "0" (A = 0, B = 1) or logic "1" (A = 1, B = 0).*

in frequency. This is because the small fluctuations in oscillator frequency due to noise or environmental variations are less likely to cause a bit flip when the two oscillator frequencies are far apart [39].

Therefore, at the time of provisioning, one can select only pairs of oscillators whose frequencies are sufficiently far apart to define the PUF output bits. This increases the PUFs robustness toward noise and environmental variations.

B. SRAM PUF

Both the arbiter PUF and the ring-oscillator PUF ultimately depend on variations in the propagation delay of gates. However, this is not the only physical property on which a PUF can be built. A popular weak PUF structure exploits the positive feedback loop in a SRAM or SRAM-like structure shown in Fig. 5. A SRAM cell has two stable states (used to store a 1 or a 0), and positive feedback to force the cell into one of these two states and, once it is there, prevent the cell from transitioning out of this state accidentally.

A write operation forces the SRAM cell to transition toward one of the two states. However, if the device powers up and no write operation has occurred, the SRAM cell exists in a metastable state where theoretically, the feedback pushing the cell toward the "1" state equals the feedback pushing the cell toward the "0" state, thereby keeping the cell in this metastable state indefinitely. In actual implementations, however, one feedback loop is always slightly stronger than the other due to small transistor threshold mismatches resulting from process variation. Natural thermal and shot noise trigger the positive feedback loop, and the cell relaxes into either the "1" or "0" state depending on this process variation.

Note that since the final state depends on the difference between two feedback loops, the measurement is differential. Therefore, common mode noise such as die temperature, power supply fluctuations, and common mode process variations should not strongly impact the transition.

Although this idea was patented in 2002, the first experimental implementation was performed in 2007, where a custom SRAM array based on 0.13-μm technology was shown to generate random values based on threshold mismatches [15], [37]. This work demonstrated a roughly normal distribution of bits and more than 90% bit stability. Additional work showed that SRAM initialization can produce a unique physical fingerprint for each chip [11], [12].

Like other strong and weak PUF implementations, the SRAM PUF is also sensitive to noise. If the two feedback loops of the SRAM cell are sufficiently close, then random noise or other small environmental fluctuations can result in an output bit flip. Therefore, error correction of this output will be necessary.

Like the ring-oscillator PUF, the architecture of the SRAM PUF can be used to make intelligent decisions regarding error coding. The key recognition is that the relative strengths of the two feedback loops in a SRAM cell are relatively static. A cell strongly biased toward "1" or "0" will remain strongly biased toward "1" or "0," respectively. Therefore, by using repeated measurements, one can assess the stability of a SRAM PUF output bit and selectively use the most stable bits as the PUF output. This process is used in conjunction with traditional coding techniques to mitigate the noise inherent to SRAM PUFs.

VI. CRYPTOGRAPHIC KEY GENERATION: WEAK PUFS

Due to their limited challenge–response space, weak PUF architectures are typically used for cryptographic key generation. In this case, a weak PUF will replace a secure nonvolatile memory that would have stored the cryptographic key. Once the key is derived from the weak PUF, it is stored in secure volatile memory during the device's operation. This key can then be used for authentication, encryption, and other cryptographic protocols. Due to the fact that one or very few keys can be generated by the PUF, the security of this key during operation is of paramount importance. If the secure key is revealed, any device can emulate the weak PUF.

A. Key Generation Protocol

Because weak PUFs like the ones discussed above have effectively fixed "challenge bits," the key generation protocol is fairly simple. In the case of the SRAM PUF, one simply powers on the SRAM and observes the memory state. Similarly for the ring-oscillator PUF, one simply pairwise compares each of the oscillators in order to measure the correct ordering of oscillation frequency.

In both of these cases, the complexity lies in the limitations of physical implementations that result in both statistical and systematic noise that must be corrected/mitigated. The actual approach used to address these issues differs for SRAM and ring-oscillator PUFs because the underlying physical implementation is different.

Ultimately, a stable set of unique bits is extracted from the weak PUF. These bits can then be used in any of a number of cryptographic protocols. Note that weak PUFs

can be used for authentication (similar to strong PUFs) even though they do not have a large number of CRPs. By supplementing the weak PUF with a hardware HMAC/AES implementation, one can achieve authentication capability at the cost of the additional power and area required by the cryptographic hardware primitives that embody the HMAC/AES protocol.

B. Arbiter PUF With Pattern Matching Error Correction

Although weak PUFs are typically used for secure key generation due to their limited challenge–response space size, protocols allowing for strong PUFs to be used in this capacity have also been developed. A key challenge in adapting strong PUFs to key generation is in correcting errors in PUF response bits. To this end, Paral and Devadas have proposed the use of a "pattern matching" technique to correct for errors in strong PUFs for use as a key generation mechanism [29].

A full description of this work is beyond the scope of this paper. In a nutshell, this approach reverses the traditional challenge–response format of a PUF. In this case, a secret offset I is chosen (the key is derived from I). A W bit portion of the response at offset I is published publicly. To recover the key, a strong PUF iterates through a deterministic set of challenges (which may depend on previous secret data that have been measured). The PUF response to this challenge contains the W pattern bits in the output. The PUF uses an error-tolerant comparison circuit (up to T bits of error) to identify the offset I of the W-bit block in the PUF response bits. The secret key is then derived from I. This process can be repeated several times to obtain larger sets of secret bits.

The study identified that, with the correctly chosen parameters (PUF output size: 1024, W: 256, T: 80), the error occurrence could be decreased such that the PUF always succeeded in regenerating the key on the first try across environmental conditions.

C. SRAM PUF Implementation

As previously mentioned, the SRAM PUF leverages the threshold voltage mismatch of transistors in a SRAM cell due to manufacturing mismatch. This mismatch results in a repeatable tendency to settle into a "1" or "0" state when the SRAM cell is powered on with no writes occurring. Several studies have constructed SRAM PUFs and analyzed their properties.

One of the first implementations of such a chip identification system was tested by Su *et al.* with RFID applications [37]. In this study, a custom SRAM cell was constructed to minimize potential systematic mismatch between the two transistors. Such a skew would result in a given SRAM cell being more likely to favor a "1" than a "0" or *vice versa*, even with random process variation. To prevent such systematic skew, they used analog layout techniques to construct a "symmetric" and "common centroid" layout of the SRAM cell.

The study demonstrated that the SRAM PUF behaved as desired. After fabrication, an equal number of SRAM cells tended toward "1" and "0" to within experimental error for both layouts. The study identified that cell positioning within the SRAM, SRAM positioning on the wafer, and subsequent wafers were all decorrelated with the SRAM cell's tendency toward "1" or "0."

A challenge arose with the recognition that roughly 4% of the SRAM cells did not have enough mismatch to strongly favor "1" or "0." These cells probabilistically settled into "1" or "0" at random due to the contributions of thermal and shot noise. The number of these unstable bits increased at temperature/voltage corners and as the chip aged.

The study by Holcomb *et al.* tested the functionality of SRAM PUFs on off-the-shelf RAM and processor products such as the MSP430 and Intel's WISP RFID device [11]. In this application, the SRAM cell was a part of another on-chip SRAM that was actively used for program/data and not custom fabricated in any way to enhance stability or skew performance.

In this way, an end user can use an existing off-the-shelf component with no silicon modification and, using software alone, implement a weak PUF for cryptographic or identification purposes.

Although off-the-shelf SRAM cells are not optimized for usage as a PUF, Holcomb *et al.* did observe a bit stability of 5% across temperatures from 0 °C to 50 °C. This stability is roughly the same as the stability measured by Su *et al.*, indicating that the custom fabrication did not help significantly in this regard.

However, the off-the-shelf SRAM cells were observed to have a significant bias toward the "1" state. This changes the entropy and unique identification analysis. In this study, 512 B of SRAM were used as a fingerprint. Due to the systematic skew of the SRAM cells, the min-entropy of this block was roughly 200 bits (plus/minus 10 bits depending on temperature). This 512-B block was then passed through a universal hash to extract 128 bits of output data.

Finally, because the SRAM is being used as a memory element for the processor, it is always powered, even if the PUF section of the SRAM is never written during normal operation. This continual powering of the SRAM in a "1" or "0" state results in negative bias temperature instability (NBTI). This is a type of "burn-in" for deep submicrometer metal–oxide–semiconductor field-effect transistor (MOSFET) technology, where the threshold voltage of a transistor increases over time due to the applied stress conditions of high temperature and a constant vertical electric field across the gate terminal while the transistor is "on."

Therefore, if a SRAM cell is powered on and set to the "0" state for a long time (~10 days), then on subsequent

power-on sequences, the cell is more likely to skew toward the "1" state. This predictable behavior stands in contrast to the behavior of temperature variations, which can either skew the cell toward "0" or "1" as the temperature fluctuates.

1) SRAM PUF Error Correction: Because these papers targeted the application of die identification, rather than key generation, this problem could be sidestepped by a statistical analysis of the probability that two independent dies would have IDs that were close enough to be misidentified as a result of this noise. Unfortunately, the nature of cryptographic operations is such that not even a single bit can be incorrect. This will require a different approach to error correction.

Maes *et al.* describe a low-overhead approach to implementing a soft-decision helper algorithm [18]. They describe a method wherein one collects confidence data in each bit by taking between 10 and 100 measurements of the SRAM PUF with N output bits prior to provisioning. This yields an output estimate vector $X \in \{0,1\}^N$, and a vector of error probabilities P_e, where the ith element of this vector corresponds to the probability that a measurement of X_i will be erroneous. Going forward, X serves as the "fuzzy secret," and P_e is public information. It has been proven by Maes *et al.* that revealing P_e does not leak any min-entropy of the response X. This work goes on to describe an implementation where the above soft-decision helper algorithm is combined with Reed–Muller codes and a universal hash function to distill the PUF output bits to a full-entropy reproducible secret key. This implementation takes 1536 SRAM PUF response bits (78% min-entropy with an average bit-error probability of 15%), and distills these data down to a 128-bit full-entropy key with a failure rate of $\leq 10^{-6}$. This approach demonstrates the feasibility of using SRAM PUFs as cryptographic key sources in spite of the errors inherent in SRAM PUF output bits.

2) Attacks on SRAM PUFs: Because the SRAM PUF provides a secure key (as opposed to providing challenge–response functionality like the strong PUF), it relies on other conventional security primitives to keep that key protected while the chip is powered. As a result, any side channel or other vulnerabilities associated with the cryptographic hardware pose a threat to the secret key outputted by the SRAM PUF. In addition, since this key is kept secret, the modeling attacks used against strong PUFs cannot be used, since no input/output relations of the PUF should ever be revealed.

However, there are other ways identified in the literature to attack a SRAM PUF more directly. Many of these depend on the level of access that one has to the SRAM. If one can insert a "write" command, then one could leverage the NBTI to deliberately force individual bits toward "1." If one could modify the temperature, one could potentially cause the PUF to fail by running the PUF outside its design area. Finally, the ability for a SRAM cell to maintain its state depends on the supply voltage. If, during the turn-on process, the supply voltage is held for some time at a low ($\sim$100 mV) voltage, the thermal noise will induce a transition into the cell's favored state, resulting in higher stability. However, if the voltage turn-on is fast, then cells become less stable. An attacker with access to the power channel could potentially control the stability of some of the SRAM PUF bits through this mechanism [12].

Recently, it has also been identified by Helfmeier *et al.* that the SRAM power-on state can be observed via near-infrared imaging of the SRAM during the turn-on transient. Once the SRAM "fingerprint" has been measured (the PUF response bits have been stolen), one can use focused ion beam (FIB) techniques to modify a second IC to have a matching fingerprint as the first by cutting traces and/or demolishing transistors in the SRAM cell [10].

Finally, one notes that SRAM data are not erased immediately on power down. The data remain "stored" in the SRAM cell for a certain short time after the cell is powered down due to an effect called "data remanence." Oren *et al.* have demonstrated that this effect can be used to inject faults into the SRAM PUF. In doing so, one can noninvasively learn the SRAM PUF output bits indirectly [28].

D. Ring-Oscillator PUF Implementation

Yu and Devadas designed a delay-based weak PUF based on the ring-oscillator architecture, and proposed the first PUF key generation architecture that does not require traditional error correction [40]–[42]. The proposed index-based syndrome coding method is a departure from prior error-correction schemes based on code-offset syndrome [5], where the syndrome format enables soft-decision functionality without the complexities associated with an explicit traditional soft-decision error-correction decoder, which in general has a higher complexity than an equivalent hard-decision error-correction decoder.

In this architecture, several oscillator PUF banks are instantiated, with each oscillator bank comprising $2k$ ring oscillators. A k-bit challenge is applied to each bank, to determine which oscillators correspond to the top delays, and which oscillators correspond to the bottom delays. The top and bottom rows are summed to produce x and y, respectively. These values are used to produce a single bit PUF output and associated "soft-decision" information corresponding to a PUF challenge. Specifically, the output bit is the sign of x–y. The "confidence" (discussed more in Section VI-D1) is derived from the magnitude of x–y.

Fig. 6 shows a simplified diagram for illustrative purposes. More complex "recombination" functions using XORs or amplitude modulation based on additional challenge bits were used in actual implementation.

Each of the oscillators is configured with "challenge bits." For the purpose of cryptographic key generation, these bits are fixed (see the "fixed challenge" in Fig. 6) in order to reproduce the same key each time.

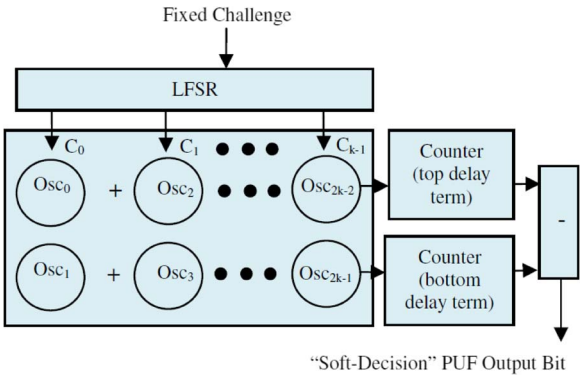

Fig. 6. ***A k-sum ring-oscillator PUF. Ring oscillators that are closer in frequency do not affect the output bit due to the summation process.***

Yu *et al.* designed multiple circuit topologies to implement this technique using a 0.13-μm CMOS process [42]. They found that, as expected, these devices produced output bits that passed all National Institute of Standards and Technology (NIST) standard randomness tests, demonstrated close to ideal decorrelation between different PUF devices, had a worst case bit bias less than 0.5%, and a raw intradevice variation of $\sim$10%.

1) Ring-Oscillator PUF Error Correction: Like the SRAM PUF, the ring-oscillator PUF also must correct for noise and environmental factors. The primary noise sources for ring-oscillator PUF architectures are similar to those for analog electronics. First, the die temperature and source voltage affect the gate delay in a known way. If the two frequencies are too close together, this noise will potentially change the output measurement bit.

The ring-oscillator PUF is inherently a differential measurement (measuring the difference in two identical sets of oscillators' paths). However, it is still susceptible to noise. As mentioned for arbiter PUFs, one can use block error codes, but this leaks some of the bits of the secret key. This is very undesirable for a weak PUF. In the case of the SRAM PUF mentioned above, Maes *et al.* were able to leverage the "confidence" in a given bit to improve coding efficiency. Yu and Devadas proposed a new error-correction scheme, "index-based coding" (IBS), that leverages this same principle for ring-oscillator PUFs [40] but does not require repeated PUF response measurements.

As mentioned above, if two sets of oscillators have frequency sums that are close together, then the readout circuit will not always report the same output bit due to slight frequency variations caused by noise and other environmental factors. Simply put, the output bit will take a value of either 0 or 1 with some probability. In such a situation, the "confidence" of a 0 or 1 measurement should be relatively low.

The IBS coding scheme recognizes the relative confidence in each bit measurement and adaptively chooses more confident bits to be part of the PUF output. In doing so, this increases the response reproducibility and decreases error.

In addition to being more efficient than block codes, Yu and Devadas also showed that the scheme leaked no information about the PUF itself under the assumption that the PUF output bits are independent and identically distributed (i.i.d.). Note that i.i.d. is the same assumption used by Maes *et al.* in proving security of the use of the probability of bit error P_e in their scheme. This is a common assumption made about PUFs, but it is actually difficult to validate in practice. Nevertheless, this is to be contrasted with the previous use of block codes (Maes' method layers block coding on top of P_e, with P_e portion proved to be information theoretically secure), which do leak some information even under a PUF i.i.d. assumption (consider a PUF with a bias of 1% and use of repetition coding). By contrast, IBS, even when used with a second stage traditional (hard-decision) error correction, remains information theoretically secure under a PUF i.i.d. assumption, even for a heavily biased PUF. IBS decouples the PUF bias statistics from the syndrome leakage security in an i.i.d. PUF setting.

A full description of the IBS coding scheme is outside the scope of this paper. Yu and Devadas implemented an IBS code on a delay-based PUF with a challenge block size of 63 bits and an average of 25 errors to correct (35.9% error) in each block. Using the IBS approach, this was reduced to a 6-bit error (9.4%). It was identified that the probability of seven or more errors was less than 0.5 ppm, so a BCH (63, 30, $t = 6$) code was used. This system successfully corrected errors across environmental conditions without failure.

In the study by Yu *et al.* in 2012, similar results were obtained [42]. In this study, and IBS coding scheme was used followed by the same BCH (63, 30, $t = 6$) code. It was observed across the four extreme voltage–temperature corners that not only did the error-corrected PUF never output an erroneous bit, but also the maximum number of errors output from the IBS scheme (the number of errors that the BCH code had to correct) was three. The BCH code can correct up to six errors, so the design was shown to have a "stability safety margin" of 50%.

2) Attacks on Ring-Oscillator PUFs: Like the SRAM PUF, the ring-oscillator PUF relies on downstream cryptographic hardware/software to protect the security of the key that is generated. However, there are ways of potentially modifying the ring-oscillator PUF's behavior. Such an attack does not reveal the output key, but may be able to influence the device to either fail to regenerate a key (denial of service), or even manipulate secret key bits if such an attack were to occur during provisioning.

For example, it was shown in 2009 that driving a sinusoidal signal on the ground plane of a ring oscillator can cause it to "lock" to that signal [26]. This study demonstrated such an attack compromising a true random number generator (TRNG). Although an attack on a PUF

would have to take into account its differential nature, the same principle can be used. By locking the frequency, an attacker can drive the frequency of a given PUF to a desired value without invasive measures.

In addition, it was shown in 2011 that the electromagnetic radiation from the ring-oscillator PUF could also be used to steal the output bits [27]. This attack can be defeated by running several oscillators in parallel, which has been done in many studies on the ring-oscillator PUF, some of which predate the identification of the attack [40]–[42].

VII. EMERGING PUF CONCEPTS

Although existing PUF technology has been successful in addressing applications in low-cost authentication and secure key generation, PUF technology still has significant untapped potential. New PUF architectures and applications are continually being developed. A full review of each of these paths is outside the scope of this paper, but a few of the popular emerging trends will be covered. For a more comprehensive coverage, we refer interested readers to a recent article on the topic [32].

A. Model-based PUFs

When considering the application of low-cost authentication, one of the primary drawbacks of strong PUF architectures is the establishment of the secret challenge–response table. Not only does this require the server to securely communicate with the PUF prior to any authentication rounds in a "secure bootstrapping" phase, but also once a CRP is used, it must be discarded and never used again. Therefore, the server must collect a large number of CRPs at manufacture and store them secretly. For large applications with thousands of PUF clients, this corresponds to a large amount of required secret storage.

To mitigate these challenges, researchers have recognized that if a PUF could have an associated "secret model" that emulates the PUF challenge–response behavior, then the secure storage requirements could be alleviated [3], [21], [24]. The secret model in the case of an arbiter PUF would be the delays of the individual stages. Newer delay-based PUF constructions have even used this compact model to link software-based attestation to intrinsic device characteristics [14]. This linking enables secure timed (and even) remote attestation.

Such a "secret model" PUF still requires both the "secure bootstrapping" phase as well as the secure storage, as the PUF and authenticating server must "agree" on a secret PUF model that describes the PUF behavior. This model must be kept secret, as it exactly describes the PUF behavior and can be used to spoof an authentication sequence. However, a server may now choose any random challenge and independently compute the correct PUF response. Further, an encrypted model can be stored on the PUF device and a reader that knows the encryption key can authenticate the device in an offline fashion.

B. Timed Authentication and Public Models

Although the secure model PUF architecture described above mitigates the secure storage requirement for PUF usage in authentication applications, it still requires secure bootstrapping and secure storage of the secret model.

Both of these requirements are alleviated by a new type of PUF described as timed authentication PUFs, public model PUFs (PPUFs), or SIMulation Possible but Laborious (SIMPL) systems [1], [20], [24], [33]. This paper will refer to this concept as a PPUF. An FPGA implementation of the PPUF was proposed alongside the concepts of an FPGA erasable PUF [20], [21]. A full characterization and compaction of the physical delays of the FPGA components is performed.

A PPUF has a model that emulates the challenge–response behavior of the PPUF hardware. This model is public—known to everyone. The key difference between the PPUF model and the PPUF hardware is that the PPUF hardware computes the response in a measurably faster time. Therefore, the authentication scheme works as follows (where a server is authenticating a Client):

1) server obtains the desired PPUF model from a trusted third party storage;
2) server generates a challenge and computes the response using the PPUF model;
3) server sends challenge to the client and begins timer;
4) the client uses its PPUF hardware to compute a response and sends it back to the server;
5) server measures the client response time T;
6) server accepts if $T < T_0$ and client's response is correct.

In the above scheme, first note that the PPUF model is stored publicly. Although it may be publicly read, the PPUF model storage must be resistant against tampering or rewriting, as the server must be able to trust that a given PPUF model is associated with a certain PPUF hardware owner. This can be done using the traditional public key infrastructure (PKI) or other similar roots of trust.

In addition, the server must be able to establish some value T_0 as described in the above scheme. This time is 1) long enough to allow the PPUF hardware to compute the response and allow for roundtrip network latency; and simultaneously, 2) short enough that no model could emulate the PPUF hardware and correctly produce a response in that time. This establishment of T_0 is the fundamental challenge of designing a PPUF capable of enacting the above authentication scheme.

It is clear that such a PPUF system would have widespread application. The key recognition demonstrating the power of a PPUF is that the PPUF hardware contains no secrets. Counterintuitively, the device is still capable of securely authenticating itself to any server. The server also contains no secret information. Simply put, there is no secret information anywhere in the protocol. The authentication capability derives solely from the computational

difference between the hardware and the model, and the unclonability of the hardware.

With this in mind, applications in embedded security are immediately obvious. A modern embedded electronic device being authenticated by a server currently uses secure nonvolatile memory (NVM) to store secret bits. Even strong and weak PUFs can be considered as "storage" devices using manufacturing variation in variables such as dopant concentration to store secret bits. In all modern electronics, this secure storage acts as the "root of trust" on which all authentication and cryptographic mechanisms are based. If the secret bits can be stolen (in the case of secure NVM) or approximated (in the case of PUF modeling), then the security is broken.

In the case of a PPUF, there are no bits to steal. The security is instead based on the difficulty of reproducing an exact copy of the PPUF hardware. This represents a fundamental shift in security paradigms. Using this mechanism, a secure embedded system can be deployed in a highly untrusted environment with a strong threat model (an adversary already has access to both the PPUF design and PPUF model) and still act as a trusted authentication source.

C. New PUF Architectures

The current primary open problem to PPUF design is the identification of a system with a provable T_0 parameter as described in Section VII-B. Intuitively, one immediately would hope for a provable asymptotic separation between the PPUF computational hardware and the computational hardware used to execute the model. Such a separation has been observed between computers leveraging quantum effects and classical computers, but such quantum computational devices are still not close to the scale required for such cryptographic applications.

Therefore, classical systems must be considered for potential practical PPUF implementations. It is recognized that such an asymptotic speedup between the PPUF hardware and the computer running the model is not possible. Classical dynamical systems at a fundamental level are governed by local differential equations (e.g., Maxwell's equations, Lagrangian mechanics, and Newtonian gravitation). As a result, one can see that a discretized universe can be simulated with only constant factor slowdown.

Qualitatively speaking, one can imagine a computer with a processor dedicated to simulating each point in space and communicates with its neighbors. Each time step can then be simulated in constant time by this set of processors running in parallel. Therefore, with enough parallelism, any classical system can be simulated with only constant factor slowdown.

Therefore, if one accepts that the PPUF model will only be slower than the PPUF hardware by a constant factor, the next step is to design a system with bounds on this constant factor. If a model is provably $10^6\times$ slower than the PPUF hardware, then this constant factor is large enough to derive an acceptable T_0 for PPUF operation.

Many PPUF architectures have been proposed [1], [20], [21], [33]. However, to date, the authors are unaware of a proposed PPUF architecture where such constant factor bounds are provable, or even those with bounds that can be strongly argued.

To establish such bounds, one first recognizes that any computational model will use CMOS technology, since CMOS is simultaneously the fastest and least expensive computational platform currently available. In recognizing this fact, one can then identify the minimum timescale of active CMOS devices as a comparison benchmark to the timescale of the differential equations describing the PPUF hardware system.

One potential avenue of approach that has been identified is in the use of optoelectronics. Optoelectronic systems simultaneously have fast enough internal dynamics to allow for significant constant factor slowdown, and are also integrable into existing CMOS processes.

In conclusion, this paper has introduced two primary applications of PUF technology: low-cost authentication and secret key generation. It has covered several of the most popular approaches to each of these applications, including arbiter PUFs, SRAM PUFs, and ring-oscillator PUFs. It has discussed potential mathematical and physical attacks on each PUF technology as well as popular error-correcting techniques for each. Finally, this paper discusses new PUF technologies such as PPUFs that demonstrate that PUF technology still has tremendous untapped potential. PUFs provide a new, secure technology for authentication and secure key storage with many advantages over existing approaches. New PUF error-correction approaches and technologies such as PPUFs represent an exciting new frontier for both PUF research as well as cryptography as a whole. ■

REFERENCES

[1] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," *Information Hiding*, vol. 5806. Berlin, Germany: Springer-Verlag, 2009, pp. 206–220, ser. Lecture Notes in Computer Science.

[2] C. Dainty, *Laser Speckle and Related Phenomena*. New York, NY, USA: Springer-Verlag, 1984.

[3] S. Devadas, "Non-networked RFID PUF authentication," U.S. Patent 8 683 210, U.S. Patent Appl. 12/623 045, 2008.

[4] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal, "Design and implementation of PUF-Based 'unclonable' RFID ICs for anti-counterfeiting and security applications," in *Proc. IEEE Int. Conf. RFID*, May 2008, pp. 58–64.

[5] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *Advances in Cryptology—Eurocrypt 2004*, vol. 3027. Germany: Springer-Verlag, 2004, pp. 523–540, ser. Lecture Notes in Computer Science.

[6] B. Gassend, "Physical random functions," M.S. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Jan. 2003.

[7] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. 9th ACM Conf. Comput. Commun. Security (CCS)*, 2002, pp. 148–160.

[8] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency Comput., Practice Exp.*, vol. 16, no. 11, pp. 1077–1098, 2004.

[9] S. Graybeal and P. McFate, "Getting out of the STARTing block," *Sci. Amer.*, vol. 261, no. 6, 1989.

[10] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2013, DOI: 10.1109/HST.2013.6581556.

[11] D. Holcomb, W. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," presented at *Conf. RFID Security*, Malaga, Spain, Jul. 11–13, 2007.

[12] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[13] D. Kirovski, "Anti-counterfeiting: Mixing the physical and the digital world," *Towards Hardware-Intrinsic Security*, A.-R. Sadeghi and D. Naccache, Eds. New York, NY, USA: Springer-Verlag, 2010, pp. 223–233.

[14] J. Kong, F. Koushanfar, P. K. Pendyala, A.-R. Sadeghi, and C. Wachsmann, "PUFatt: Embedded platform attestation based on novel processor-based PUFs," presented at the ACM/IEEE Design Autom. Conf., San Francisco, CA, USA, Jun. 1–4, 2014.

[15] P. Layman, S. Chaudhry, J. Norman, and J. Thomson, "Electronic fingerprinting of semiconductor integrated circuits," U.S. Patent 6 738 294, Sep. 2002.

[16] J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits with identification and authentication applications," in *Proc. IEEE VLSI Circuits Symp.*, 2004, pp. 176–179.

[17] D. Lim, "Extracting secret keys from integrated circuits," M.S. thesis, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, May 2004.

[18] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs," *Cryptographic Hardware and Embedded Systems—CHES 2009*, vol. 5747. Berlin, Germany: Springer-Verlag, 2009, pp. 332–347, ser. Lecture Notes in Computer Science.

[19] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong PUFs," Rep. 2013/632, 2013.

[20] M. Majzoobi, A. Elnably, and F. Koushanfar, "FPGA time-bounded unclonable authentication," *Information Hiding*, vol. 6387. Berlin, Germany: Springer-Verlag, 2010, pp. 1–16, ser. Lecture Notes in Computer Science.

[21] M. Majzoobi and F. Koushanfar, "Time-bounded authentication of FPGAs," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pt. 2, pp. 1123–1135, Sep. 2011.

[22] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design*, 2008, pp. 670–673.

[23] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *Proc. IEEE Int. Test Conf.*, 2008, DOI: 10.1109/TEST.2008.4700636.

[24] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, 2009, DOI: 10.1145/1502781.1502786.

[25] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, secure authentication by substring matching," in *Proc. IEEE Symp. Security Privacy Workshops*, 2012, pp. 33–44.

[26] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2009, pp. 317–331.

[27] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive em attack on FPGA RO PUFs and countermeasures," in *Proc. Workshop Embedded Syst. Security*, 2011, pp. 2:1–2:9.

[28] Y. Oren, A.-R. Sadeghi, and C. Wachsmann, "On the effectiveness of the remanence decay side-channel to clone memory-based PUFs," in *Proc. Int. Workshop Cryptogr. Hardware Embedded Syst.*, 2013, pp. 107–125.

[29] R. S. Pappu, Z. Paral, and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2011, pp. 128–133.

[30] R. S. Pappu, P. S. Ravikanth, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, pp. 2026–2030, 2002.

[31] M. Rostami, M. Majzoobi, F. Koushanfar, D. Wallach, and S. Devadas, "Robust and reverse-engineering resilient PUF authentication and key-exchange by substring matching," *IEEE Trans. Emerging Topics Comput.*, 2014, DOI: 10.1109/TETC.2014.2300635.

[32] M. Rostami, J. B. Wendt, M. Potkonjak, and F. Koushanfar, "Quo vadis, PUF?: Trends and challenges of emerging physical-disorder based security," in *Proc. Conf. Design Autom. Test Eur.*, 2014, article 352.

[33] U. Rührmair, "SIMPL systems: On a public key variant of physical unclonable functions," International Association for Cryptologic Research, Tech. Rep., 2009.

[34] U. Rührmair, S. Devadas, and F. Koushanfar, "Security based on physical unclonability and disorder," *Introduction to Hardware Security and Trust*, M. Tehranipoor and C. Wang, Eds. New York, NY, USA: Springer-Verlag, 2012, pp. 65–102.

[35] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 237–249.

[36] U. Rührmair, X. Xu, J. Slter, A. Mahmoud, F. Koushanfar, and W. Burleson, "Power and timing side channels for PUFs and their efficient exploitation," Rep. 2013/851, 2013.

[37] Y. Su, J. Holleman, and B. Otis, "A 1.6 pJ/bit 96 (percent) stable chip ID generating circuit using process variations," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2007, pp. 200–201.

[38] G. E. Suh, "AEGIS: A single-chip secure processor," Ph.D. dissertation, Dept. Electr. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Aug. 2005.

[39] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. ACM/IEEE Design Autom. Conf.*, 2007, pp. 9–14.

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

[41] M.-D. M. Yu, D. M'Raihi, R. Sowell, and S. Devadas, "Lightweight and secure PUF key storage using limits of machine learning," in *Cryptographic Hardware and Embedded Systems—CHES 2011*, vol. 6917. Berlin, Germany: Springer-Verlag, 2011, pp. 358–373, ser. Lecture Notes in Computer Science.

[42] M.-D. M. Yu, R. Sowell, A. Singh, D. M'Raihi, and S. Devadas, "Performance metrics and empirical results of a PUF cryptographic key generation ASIC," in *Proc. IEEE Int. Symp. Hardware-Oriented Security Trust*, 2012, pp. 108–115.

ABOUT THE AUTHORS

Charles Herder received the B.S. degree in electrical engineering and computer science, the B.S. degree in physics, and the M.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA.

He has published in many fields including physical chemistry, single-photon detection, embedded cryptosystems, and power management. His research interests are in the physics of computation, cryptography, computer architecture, and computer security. His prior experience at Texas Instruments includes developing embedded systems authentication technology and serving as lead technical point-of-contact for the development of proprietary power management and authentication systems for several major developers including Dell, RIM, Cisco, and Phillips.

Meng-Day (Mandel) Yu received the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA.

He is a Technical Director at Verayo, San Jose, CA, USA. Before that, he was the Manager of R&D Engineering at TSI, a secure wireless radio startup. Before that, he was an ASIC Engineer and later a Systems Engineer at TeraLogic focused on signal processing and conditional access systems. His research interests include coding theory, signal processing, and computer security.

Farinaz Koushanfar received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1998, the M.S. degree from the University of California Los Angeles (UCLA), Los Angeles, CA, USA, and the M.A. degree in statistics and the Ph.D. degree in electrical engineering from the University of California Berkeley, Berkeley, CA, USA in 2005.

She is an Associate Professor of Electrical and Computer Engineering (ECE) at Rice University, Houston, TX, USA. She is the Director of the Adaptive Computing and Embedded Systems (ACES) Laboratory.

Srinivas Devadas (Fellow, IEEE) received the M.S. and Ph.D. degrees from the University of California Berkeley, Berkeley, CA, USA, in 1986 and 1988, respectively.

He is the Webster Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He joined MIT in 1988 and served as the Associate Head of the Department of Electrical Engineering and Computer Science, with responsibility for Computer Science, from 2005 to 2011. His research interests include Computer-Aided Design, computer architecture, and computer security.