Secure Remote Sensing and Communication using Digital PUFs

Teng Xu, James B. Wendt, and Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{xuteng, jwendt, miodrag}@cs.ucla.edu

ABSTRACT

Small form, mobile, and remote sensor network systems require secure and ultralow power data collection and communication solutions due to their energy constraints. The physical unclonable function (PUF) has emerged as a popular modern low power security primitive. However, current designs are analog in nature and susceptible to instability and difficult to integrate into existing circuitry. In this paper, we present the digital PUF which is stable in the same sense that digital logic is stable, has a very small footprint and very small timing overhead, and can be easily integrated into existing designs. We demonstrate the use of the digital PUF on two applications that are crucial for sensor networks: trusted remote sensing and logic obfuscation. We present our security analysis using standard randomness tests and confusion and diffusion analysis, and apply our new obfuscation approach on a set of standard design benchmarks.

Categories and Subject Descriptors

B.7 [**Hardware**]: Integrated Circuits; C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*

General Terms

Security.

Keywords

Sensor networks, digital PUFs, trusted sensing, hardware logic obfuscation.

1. INTRODUCTION

Sensor networks have been widely researched over the last decade but have received new interest and a fresh perspective with emerging trends and developments in the Internet of Things (IoT). Sensor networks are comprised of small form, mobile, and remote devices. They are also often placed in unattended locations, and sometimes even in hostile environments. Due to their remote and unattended nature, they are susceptible to physical and side-channel attacks. Thus, it is crucial that these devices be developed with security in mind. Specifically, both the device itself as well as the data it collects must be secured.

The physical unclonable function (PUF) is a cryptographic primitive that has been suggested for sensor network security due to its low power requirements. PUFs are physical devices that have a random but deterministic mapping of inputs to outputs. Their unclonability—and functionality—are often inextricably tied to the physical characteristics of the device components (e.g. gate delay, leakage energy). While PUFs receive and generate digital inputs and outputs, they are analog in nature due to their reliance and design based on their inherent physical characteristics. Thus, current PUFs have many limitations. The most limiting of which includes stability and susceptibility to environmental and operational conditions. Many PUFs, including the standard delay-based PUF require arbiters to operate. These memory components limits the PUF in terms of placement and coordination in circuitry since their outputs cannot be used directly in the current cycle like a combinational module, but require an additional clock cycle to be used.

These main limitations can be removed by creating a purely digital PUF. The digital PUF must be stable in the same sense that digital logic is stable against environmental and operational conditions and must produce deterministic outputs for all input vectors. The digital PUF must integrate with existing combinational logic without requiring additional clock cycles to use its outputs. And lastly, the digital PUF must be flexible in the sense that its structure can be altered for different tradeoffs between security, energy, and delay as required by the pertinent task.

In this paper, we present a digital PUF design with such characteristics. Its underlying architecture consists of a series of lookup tables (LUTs) which are initialized using standard delay-based PUFs. The standard PUFs enable both unclonability and configurability in our design. Despite the inherent instabilities known to exist in them, we ensure stability through two means: (a) through a slight modification in the standard delay-based PUF design that enables stable output validation and input selection, and (b) through a reduction in use to only circuit initialization, thus tremendously reducing the impact of device aging on its gate delays.

We analyze the security of the digital PUF as it stands alone by applying the NIST randomness benchmark test

ANCS'14, October 20–21, 2014, Los Angeles, CA, USA.

http://dx.doi.org/10.1145/2658260.2658279.

suite [1] and demonstrating that it passes all tests. We also analyze the outputs of our digital PUF using the security principles of confusion and diffusion, as presented by Shannon [2], through demonstration of the avalanche criterion.

However, despite the theoretically sound and mathematically proven security properties of many digital cryptographic systems, there exist many potential side-channels which can effectively bypass these mathematical constructs altogether by reading internal memory or inferring internal procedures through power analysis and memory attacks. Since our digital PUF utilizes memory cells, such as arbiters, SRAM, and flip-flops in its LUTS, it is potentially susceptible to side-channel attacks [3]. We demonstrate that these attacks can be prevented through analysis of modern feature sizes, the use of 3D integrated circuitry, and the use of inspection resistent memory [4].

Finally, we explore two important applications of the digital PUF. Due to the unattended nature of a sensing node, it is possible that an attacker tamper with it in such a way so as to alter the data it is sensing. Thus, our first application is secure data collection, or trusted sensing.

The second application we present is secure information flow through hardware logic obfuscation. Hardware obfuscation is an essential task for the protection of hardware intellectual property (IP). An unattended sensing node can easily be stolen by an attacker with the hopes of learning secure or private information. Through logic obfuscation we prevent the attacker from knowing what the actual functionality of the node is, thus preventing an attacker from being able to learn anything.

The digital PUF enables actual integration with digital circuitry, and most importantly, the actual implementation of logic. The essential idea behind our hardware obfuscation approach is to replace an arbitrary piece of logic with a digital PUF and programmable fabric. The digital PUF serves to obfuscate the inputs in an unpredictable way, while the programmable fabric produces the correct outputs as defined by the original replaced logic. Our PUF is particularly favorable for this application because its digital and combinational nature allow us to obfuscate any arbitrary piece of logic anywhere in a circuit without inducing additional cycles. To analyze our techniques we introduce new metrics for measuring the difficulty in reverse engineering the obfuscated logic.

2. RELATED WORK

Pappu et al. introduced the concept of the first PUF and demonstrated it using mesoscopic optical systems [5]. Devadas' research group at MIT developed the first family of silicon PUFs through the use of intrinsic process variation in deep submicron integrated circuits [6]. Guarardo and his coworkers at Philips Research in Eindhoven demonstrated how PUFs can create unique startup values in SRAM cells [7]. Consequently a great variety of technologies were used for PUF creation including IC interconnect networks, thyristors, memristors, and several nanotechnologies. Although a variety of PUF structures have been proposed, arbiter-based (APUF) [6], ring oscillator-based (RO-PUF) [8], and SRAM PUFs [7] are by far most popular.

PUFs were immediately applied to a number of applications including authentication, cryptographic key generation and secure storage [9], anti-counterfeiting [10], FPGA intellectual property (IP) protection [11], remote enabling and disabling of integrated circuits [12], and remote trusted sensing [13] [14]. PUFs are also used in conjunction with traditional creation and operation of remote secure processors [15]. The security role of the PUF has been greatly enhanced with several proposals for employing PUFs in public key security protocols in systems such as the public PUF (PPUF), SIMPL, and one time pads [16] [17].

There have been two efforts that aim to remove the limitations of analog PUFs. The first is the digital bimodal function (DBF) [18]. The DBF easily passes several security and randomness tests, but is not unclonable and cannot be integrated with regular digital logic without significant time overhead. In the second effort, Fyrbiak et al. proposed the creation of software security primitives using hardware random generators [19]. Hardware-software security primitives require relatively long execution times and depend on unspecified reproducible random generators.

A large number of security attacks on essentially all types of PUFs have been explored. They can be classified into two groups: reverse engineering (also called characterization) and manufacturing or emulation attacks. Non-invasive characterization attacks mainly target the delay-based PUFs (e.g. APUF and RO-PUF). These attacks mainly use numerical algebra and machine learning techniques. For example, Majzoobi et al. demonstrated how linear programming can be used to characterize delay PUFs [20]. By far the most popular statistical attack was reported by Rührmair at al. in which a relatively small number of challenge-response pairs yielded highly accurate prediction models [21]. Most recently, Xu and Burleson proposed coordinated side-channel and machine learning attacks [22].

There are a number of well studied side-channel attacks either on cryptographical protocols and devices or directly on PUFs including timing, power, electromagnetic emanation, optical, and variety of memory reading attacks including the use of focused ion beams [23] [24]. Note that attacks such as cache behavior attacks are not applicable to PUFs. For instance, it has been practically demonstrated that several side-channel attacks can read data stored in DRAM and SRAM cells [25]. For example, the security research group at Technische Universität Berlin reported successful physical cloning of SRAM PUFs [3].

Side-channel attacks use a variety of physical phenomena and sophisticated engineering approaches, often with high effectiveness. Still, there is a strong belief that APUFs and other delay-based PUFs are either safe or at least much more resilient against side-channel attacks due to their small difference in physical signals and dependency on difficult-to-measure threshold voltages that depend on the number of dopants and their distribution in transistor channels along with other physical characteristics of the device.

A multitude of techniques for layout reconstruction and reverse engineering of integrated circuit functionality have been proposed and demonstrated over the last couple decades [26]. The goal of hardware obfuscation is to prevent reverse engineering. One popular approach is to append unique structures to the design in such a way that only the designer of the circuit can enable the correct functional execution [12]. Another approach harnesses the physical structure of gates, specifically differences in small implementation details that cannot be easily deduces using existing reverse engineering techniques [27]. We demonstrate that it is possible with very low overhead to make each integrated circuit of a particular

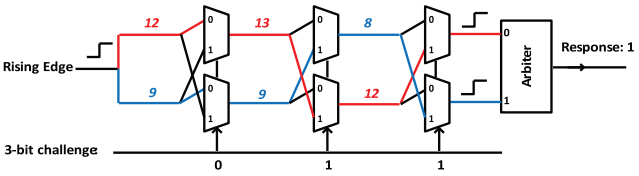


Figure 1: Applying a 3-bit input challenge to a delay-based PUF. The challenge is intentionally chosen in this example in such a way that the delay difference between the two paths (red and blue) are maximized.

Challenge	Delay Difference
000	3
001	-3
010	-11
011	11
100	-3
101	3
110	-5
111	5

Table 1: Delay differences between all possible paths in the example delay-based PUF in Figure 1.

design unique and therefore greatly increase the difficulty of reverse engineering because reverse engineering one IC does not help in reverse engineering a second. Another beneficial side effect is that now energy consumption of each circuit becomes unique.

In this paper we present three primary contributions that differentiate our work from prior art: (i) the digital PUF, (ii) logic obfuscation, and (iii) remote sensing and communication. The current state-of-the-art PUF, in particular, the SRAM PUF, has two problems, the first is that it is susceptible to side-channel attacks, and the second is that it is unstable. Our digital PUF outperforms the SRAM PUF because no one can reverse engineer the digital PUF due to storing its secret key in a stable analog delay-based PUF. In terms of state-of-the-art ring oscillator based and delay based PUFs, we demonstrate that the digital PUF is not only stable, but also digital, which means that we can easily integrate the system into existing logic.

Recently, Zheng et al. has proposed a reconfigurable digital PUF [28]. Our design differs in three regards. The first is that Zheng's digital PUF requires significantly more resources and real-time configuration mechanisms which limits its implementation to only FPGA systems, while our digital PUF can be implemented on any platform, including ASIC and programmable processors, and is much smaller and faster. The second difference is that Zheng focuses on code obfuscation and protection of software which cannot be done using our system because we assume that one will change the primary inputs to the system. However, our digital PUF design can be operated externally as any other system, the only difference is that internally it is very different. The final difference is that we enable two distinctly different applications that the reconfigurable PUF cannot implement without very large overhead.

Regarding our second main contribution, logic obfuscation, existing solutions utilize special implementations, and often obfuscate using techniques external to the actual logic mechanisms. Our system allows for obfuscation of any system regardless of what kinds of gates it uses and does so by putting the mechanism inside of the logic.

As for our third contribution, no one else has implemented remote secure trusted communication using PUFs that enables that the source of the transmissions can be trusted. For example, this could be essential for secure remote access of files on the web through FTP. Furthermore, the use of the digital PUF enables for low overhead and, more importantly, resiliency with respect to operational and environmental conditions and aging.

3. PRELIMINARIES

3.1 Delay-based PUF Stability

Figure 1 depicts an example of a 3-bit delay-based PUF. Each challenge bit controls the inputs of two multiplexers. An output bit is generated by assigning a challenge vector and sending a rising edge through the PUF. The two paths traverse the three delay segments, swapping positions (top and bottom) depending on the input bit at each segment, before arriving at the arbiter which determines the final output. For example, an input challenge of 011 generates the blue and red paths depicted. An arbiter will set its value to 0 or 1 depending on which path (top or bottom) arrives first, effectively selecting the path that has the smaller delay. Table 1 consists of the delay differences between the top and bottom paths for all possible paths in the example PUF in Figure 1.

A key observation is that for each unique delay-based PUF there exists a set of challenges that produce stable outputs. Consider the situation in which environmental conditions affect the physical characteristics of the circuit. For example, variations in temperature cause variations in individual gate delays, thereby affecting the overall path delays in the analog PUF. Since challenges 011 and 010 result in a large difference in delay between the two racing paths, it is still with high possibility that the red path will have a larger delay compared to the blue path despite the effects temperature may have on the individual gate delays. We label this challenge, and any other challenges that are resilient to such environmental changes, as stable inputs.

For path delay analysis we introduce a *delay ratio* metric, which is defined as the delay differences of two paths divided by the delay of the shorter path. For the purposes of testing, we assume that gate delays follow a normal distribution due to the effects of process variation.

For different original delay ratios and varying temperatures we use the Hotspot tool [29] to simulate the standard delay-based PUF and measure the probability that its output is stable. Table 2 shows the results of a 32-bit delay-based PUF. As expected, a higher original delay ratio yields higher probabilities for stable outputs. For example, for an original delay ratio of 0.1, the probability that the PUF output remains stable across temperatures ranging from 250K to 400K remains 1.

The results of our 64-bit PUF tests are shown in Table 3. Compared to the 32-bit test case, the 64-bit test case demonstrates a similar trend and exhibits even better stability under the same conditions. As long as the original delay ratio reaches a particular threshold (e.g. 0.1 in this experiment), the outputs remain stable for a wide range of temperatures. Hence, we select those challenges that satisfy this delay ratio threshold as the stable challenges.

Temperature	Delay Ratio (T=300K)						
	0.04	0.05	0.06	0.07	0.08	0.09	0.1
250K	0.979	0.987	0.994	0.997	1	1	1
350K	0.969	0.975	0.989	0.994	0.997	1	1
400K	0.937	0.951	0.959	0.977	0.988	0.996	1

Table 2: Probability that outputs of the 32-bit PUF are stable over varying temperatures for different delay ratios.

Temperature	Delay Ratio (T=300K)						
	0.04	0.05	0.06	0.07	0.08	0.09	0.1
250K	0.984	0.986	0.996	0.998	1	1	1
350K	0.982	0.986	0.993	0.998	1	1	1
400K	0.954	0.974	0.986	0.991	0.997	1	1

Table 3: Probability that outputs of the 64-bit PUF are stable over varying temperatures for different delay ratios.

	$P(R \geq 0.04)$	$P(R \geq 0.06)$	$P(R \geq 0.08)$	$P(R \geq 0.1)$
32-bit PUF	12.51%	4.27%	1.07%	0.21%
64-bit PUF	9.34%	2.44%	0.43%	0.05%

Table 4: Probability that the delay ratio (R) is larger than the labelled threshold value for a 32-bit and 64-bit PUF.

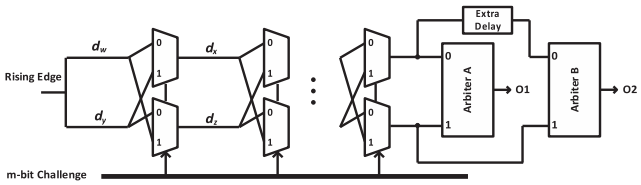


Figure 2: Architecture for stable challenge-response testing.

An important issue to address is how to obtain these stable challenges. Gate level characterization is possible but may require expensive efforts and costs. We have proposed an easy but feasible alternative. As shown in Figure 2, before the two paths reach the arbiter, we intentionally place extra delays within one of the paths. Assuming we are measuring an m-bit PUF, and assuming the expected delay of each stage is D (since delays are distributed normally in the presence of process variation), the expectation of the total delay for each path is $m \times D$. Therefore, we intentionally add $0.1 \times m \times D$ delay to one of the paths using our additional architecture. When applying a challenge, if the path with extra delay still reaches Arbiter B earlier than the other path reaches both Arbiter A and B (i.e. both O_1 and O_2 are 0), then we can claim with certainty that the challenge coupled with this particular PUF produces a path difference of 0.1 or greater and can thus be regarded as a stable input.

We search for stable inputs in this manner by applying many random challenges to the delay-based PUF. Table 4 shows the probability that for a random input challenge the corresponding *Delay Ratio* is larger than a particular threshold value. Although the portion of stable challenges is small, the time required to build a reasonable amount of stable challenges is negligible due to the fact that each test only requires a single clock cycle.

3.2 Digital Bimodal Function

The concept of the digital bimodal function (DBF) was first proposed by Xu et al. [18]. The essential idea behind the DBF is to represent a set of binary functions in two forms, one which is fast and compact ($f_{compact}$) and the other which is slow and complex ($f_{complex}$). Both forms have exactly the same functionality, in other words, given the same inputs, both forms produce the same outputs.

Equation 1, 2, and 3 illustrate an example of a DBF. As a prerequisite, a_i, b_i, and c_i are binary values, and the function sets f and g are Boolean functions in the form of sums of products (SOP) and/or products of sums (POS) representing $f_{compact}$ and $f_{complex}$, respectively. Equation 1 represents the relationship between a_i and b_i and Equation 2 represents the relationship between b_i and c_i. Note that each function f has 4 binary inputs assigned in a random and permanent order.

Equation 3 is generated by substituting 1 into 2, yielding a direct relationship between a_i and c_i. Note that substitutions are expanded and simplified so that each sub function in g is in the form of a SOP or a POS. The key observation here is that while both f and g implement the same functionality, f can be computed much more rapidly than g since it is in a compact format in which each subfunction requires only four inputs, while g is in an expanded format in which each subfunction requires up to n variables. It has been shown that the size difference between $f_{compact}$ and $f_{complex}$ increases exponentially with an increase in input variables and additional levels of substitution [18].

Figure 3 depicts the FPGA-based implementation of the DBF example defined in Equations 1, 2, and 3. The architecture is composed of two levels of 4-input LUTs. Note that each 4-input LUT implements a 4-input Boolean function from f. A hierarchy structure is constructed by feeding the outputs of the previous level of LUTs to the inputs of the next level of LUTs which is equivalent to the function substitution. Therefore, the LUT network directly implements $f_{compact}$ in the DBF. As the number of inputs and number of levels in the LUT network grows, the expanded form of $f_{complex}$ becomes very difficult to implement in hardware (grows exponentially) while $f_{compact}$ remains in a relatively compact form (grows linearly).

Unfortunately, one significant drawback of the DBF is that it can easily be reverse engineered once an attacker gains access to the configuration of the LUT network. Our digital PUF solves this problem by integrating the delay-based PUF into the design and using only stable challenge-response pairs to initialize a random subset of LUTs comprising the DBF.

Inputs: $a_i \in \{0,1\}, i \in \{0,1,2...n-1\}$
Outputs: $c_i \in \{0,1\}, i \in \{0,1,2...n-1\}$
Variables: $b_i \in \{0,1\}, i \in \{0,1,2...n-1\}$
$r_j \in \{0,1,2...n-1\}, j \in \{0,1,2...8n-1\}$

$$\begin{cases} b_0 = f_0(a_{r_0}, a_{r_1}, a_{r_2}, a_{r_3}) \\ b_1 = f_1(a_{r_4}, a_{r_5}, a_{r_6}, a_{r_7}) \\ b_2 = f_2(a_{r_8}, a_{r_9}, a_{r_{10}}, a_{r_{11}}) \\ \dots \\ b_{n-1} = f_{n-1}(a_{r_{4n-4}}, a_{r_{4n-3}}, a_{r_{4n-2}}, a_{r_{4n-1}}) \end{cases} \quad (1)$$

$$\begin{cases} c_0 = f_n(b_{r_{4n}}, b_{r_{4n+1}}, b_{r_{4n+2}}, b_{r_{4n+3}}) \\ c_1 = f_{n+1}(b_{r_{4n+4}}, b_{r_{4n+5}}, b_{r_{4n+6}}, b_{r_{4n+7}}) \\ c_2 = f_{n+2}(b_{r_{4n+8}}, b_{r_{4n+9}}, b_{r_{4n+10}}, b_{r_{4n+11}}) \\ \dots \\ c_{n-1} = f_{2n-1}(a_{r_{8n-4}}, a_{r_{8n-3}}, a_{r_{8n-2}}, a_{r_{8n-1}}) \end{cases} \quad (2)$$

$$\begin{cases} c_0 = g_0(a_0, a_1, a_3, \dots, a_{n-1}) \\ c_1 = g_1(a_0, a_1, a_3, \dots, a_{n-1}) \\ c_2 = g_2(a_0, a_1, a_3, \dots, a_{n-1}) \\ \dots \\ c_{n-1} = g_{n-1}(a_0, a_1, a_3, \dots, a_{n-1}) \end{cases} \quad (3)$$

4. DIGITAL PUF

4.1 Architecture

Figure 4 depicts the architecture of the digital PUF. At startup, the user selects and applies stable challenge vectors, supplied by the digital PUF manufacturer, to an array of delay-based PUFs. The resultant stable outputs are then used to initialize and configure individual LUT cells in the DBF. This procedure is applied to a random subset of LUT cells, while the remaining cells are initialized by the user. This bifurcation in initialization enables self trust by preventing malicious manufacturers from completely controlling the DBF configuration process.

After PUF initialization, the user generates an input-output mapping for the DBF which serves as a specification of $f_{complex}$. This is easily done by traversing all the possible inputs and generating the corresponding output. The mapping is stored as Boolean functions in both SOP and POS forms.

By applying only stable challenges to the delay-based PUF at initialization we ensure that the entire digital PUF system is completely stable. Furthermore, the intrinsic unclonability of the delay-based PUF along with its integration with the DBF guarantees that the overall architecture is unclonable. Since the delay-based PUF is used only at initialization and is subsequently disregarded and the rest of the digital PUF operation is delegated to the DBF, we inherit the small power, area, and low delay properties of the DBF as discussed by Xu et al. [18].

4.2 Side-channel Attacks

In this section we discuss solutions for protecting the digital PUF against side-channel attacks on the LUT memory cells.

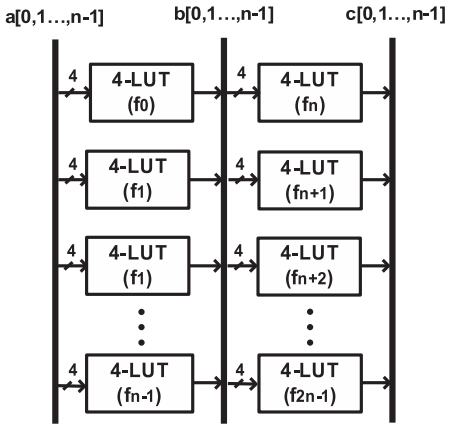


Figure 3: An example of the FPGA-based DBF LUT network.

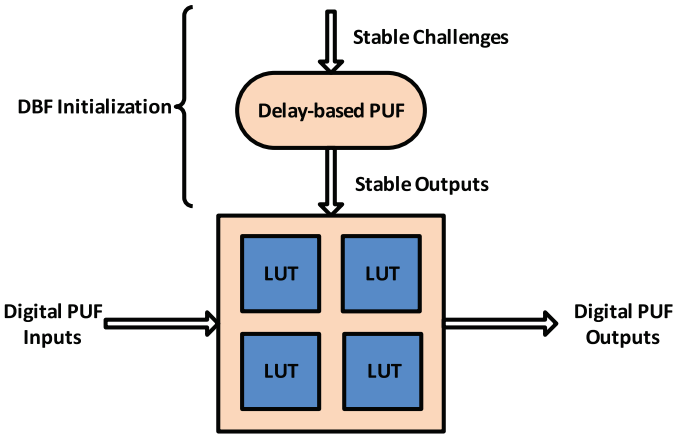


Figure 4: Architecture of the digital PUF. Note that the stable outputs from the analog PUF are used only once at startup to initialize and configure the LUTs in the DBF.

MicroSemi/Actel's antifuse-based FPGA employs one-time programmable connections that are non-volatile [30]. After each fuse is programmed, its probe and programming interface is automatically disabled. Actel fuses are smaller than the regular feature size of the FPGA and therefore are much less susceptible to destructive reverse engineering techniques. They have a very small power footprint (below 40 µJ) that is significantly smaller than the footprint of a transistor. There are many millions or fuses and recovering their values is at best a very time consuming task. Note that in our approach, the fuses would be programmed in a unique way for each circuit. While dynamic reprogramming of fuses is not feasible, one can easily organize several combinations of fuses in such a way that their software activated combination produces a unique digital PUF.

The second potential approach to side-channel prevention is the use of inspection resistant memory proposed by Valamehr et al. [4]. They employ a combination of secret sharing and secure hashing to reduce the probability of correct key or device recovery to 10^{-12} even if the probability of incorrectly recovering a value from a particular location is only 5%. The overall hardware overhead is equal to slightly more than 7 SRAM cells. Note that as observed by Valamehr

et al. their techniques can be combined with antifuse mechanisms.

Our most preferable solution against side-channel attacks is the use of 3D integrated circuit technology [31]. Recently, 3D integrated circuits have emerged as a practical industrial option that reduces some design constraints, such as long interconnect, yield, and levels of integration. 3D has been proposed several times for security application [32] [33], but only very recently has it been advocated as a platform for the detection of intrusive side-channel attacks [34]. Also, the use of configurable shields against side-channel attacks has been explored [35].

We propose the use of 3D implementation in which the security device, in our case, the digital PUF, is placed in the middle-most layers. Devices with the same architecture but with randomly selected parameters are placed both above and below the actual device. Another alternative is that all these devices are used in a standard secret sharing mode. Therefore, no backside access is possible, and the performance of electromagnetic attacks are drastically reduced or even eliminated. Finally, to prevent attacks through the same layers we can employ an active shield [35]. Our final observation is that all three discussed techniques are orthogonal and can be combined. Each technique does introduce some overhead but is relatively small. 3D techniques are applicable only if 3D technology is used.

4.3 Security Properties

In this section, we adopt a set of standard statistical tests to analyze the security properties of the digital PUF. We describe possible statistical attacks and test the resilience of our digital PUF against such attacks. We use the standard digital PUF structure with 64-bit inputs and outputs and 32 levels of substitution. We assume that the digital PUF is initialized randomly.

4.3.1 Output Randomness

We quantify the output randomness of the digital PUF by applying the industry standard statistical test suite provided by the National Institute of Standards and Technology (NIST). We generate a stream of outputs in the following way: a random seed is used as the primary inputs to the digital PUF after random configuration and the corresponding outputs are generated. In each subsequent clock cycle, the outputs are XORed with the previous inputs to generate the inputs for the next clock cycle. We repeat the process until we collect enough outputs required by the benchmark suite. The results in Table 5 indicate that the output stream of the digital PUF passes the NIST randomness tests.

Statistical Test	Avg. Success Ratio
Frequency	100%
Block Frequency (m=128)	98.7%
Cusum-Forward	97.8%
Cusum-Reverse	97.9%
Runs	98.4%
Longest Runs of Ones	97.9%
Rank	99.3%
Spectral DFT	97.5%
Non-overlapping Templates ($m = 9$)	97.5%
Overlapping Templates ($m = 9$)	97.5%
Universal	100%
Approximate Entropy ($m = 8$)	98.1%
Rand. Excursions ($x = 1$)	98.8%
Rand. Excursions Variant ($x = -1$)	97.6%
Serial ($m = 16$)	99.3%
Linear Complexity ($M = 500$)	98.0%

Table 5: NIST randomness test results on the digital PUF. 1,000 bitstreams of 10,000 bits are provided to each test. Each test passes for p-value$\geq \sigma$, where $\sigma = 0.01$.

4.3.2 Avalanche Effect

In this attack, an adversary attempts to predict the outputs of the digital PUF using the knowledge of outputs for similar inputs. This attack is only dangerous for systems in which output vectors of similar inputs are highly correlated with one another. In cryptography, cipher diffusion is achieved if a change in the input by one bit results in a dramatic change in the outputs in an unpredictable manner. This is otherwise known as the avalanche effect. To test this, we measure the hamming distance between two output vectors whose input vector differ by one bit. Ideally, the distribution should be in the form of a binomial distribution with the peak at half of the number of output bits. The result in Figure 5a shows an almost perfect binomial distribution which indicates our matched device satisfies the avalanche criterion and is highly resilient against this type of attack.

4.3.3 Input-based Correlation

Another type of attack utilizes correlations between individual output bits, O_i, and input bits, I_j, for prediction. The goal in this attack is to predict the conditional probability, $P(O_i = c_1|I_j = c_2)$, where c_1 and c_2 are either 1 or 0. For example, if the attacker observes that output O_i is equal to 1 when the input I_j is 1 a large majority of the time, then he can guess with a high probability that output O_i is 1 when I_j is 1. The ideal situation is when all conditional probabilities are 0.5. Figure 5b depicts the distribution of conditional probabilities, $P(O_i = 1|I_j = 1)$, for the digital PUF. The majority of probabilities cluster around 0.5, thus indicating low potential for prediction.

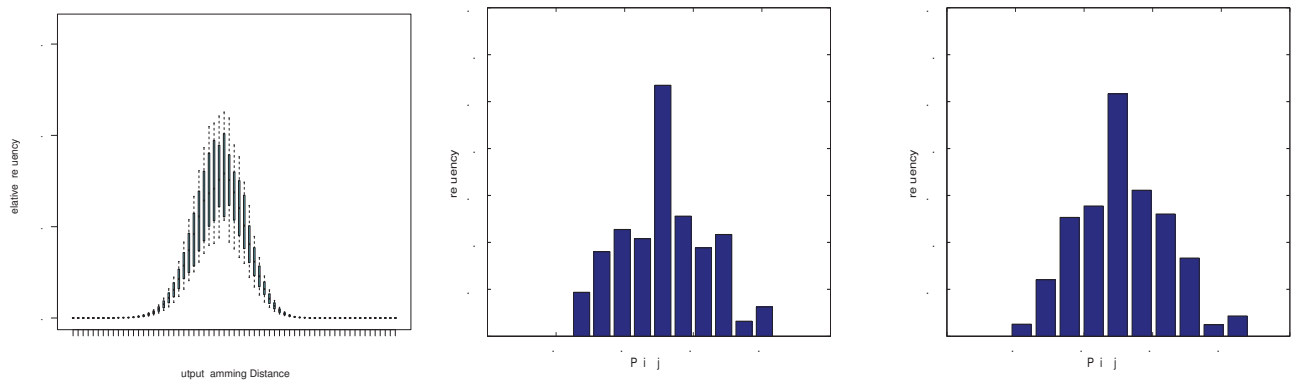
4.3.4 Output-based Correlation

Similar to the previously described attack, this attack attempts to predict an output bit O_i according to the value of a corresponding output bit O_j. In this case, if two output bits have a strong correlation, then the attacker can deduce the output vector through knowledge of a subset of output bits. We present the distribution of conditional probabilities, $P(O_i = 1|O_j = 1)$, in Figure 5c which depicts low potential for prediction based on output to output correlation.

4.4 Structure Exploration

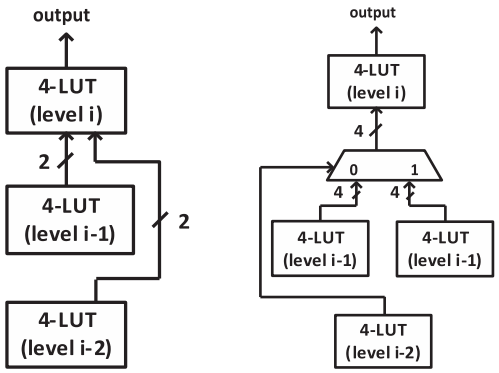
The core architecture of the digital PUF is the randomly connected LUT network. In this section we address how to connect the LUTs to achieve optimal security, while ensuring that the the structure has a small area and delay overhead. The following factors can directly influence the LUT structure:

- **Number of Inputs.** The size and the complexity of $f_{complex}$ is directly dependent on the number of inputs to $f_{compact}$. Thus, the number of inputs should be



(a) (b) (c)

Figure 5: (a) Distribution of output hamming distances testing the avalanche effect. The error bars depict the max, 0.75 quantile, mean, 0.25 quantile, and min frequencies. (b) Probability distribution of conditional probabilities between output bits O_i and input bits I_j. (c) Probability distribution of conditional probabilities between output bits O_i and output bits O_j.



(a) (b)

Figure 6: Examples of feed forward structures applied to the digital PUF. (a) Inputs arrive from all previous levels. (b) Inputs arrive from and are controlled by previous levels.

selected in such a way so as to satisfy the application requirements for security, delay, and area.

- **Number of Levels.** Adding more LUT levels to the digital PUF causes more diffusion, however, also increases the delay and area costs.
- **LUT Connections.** In Figure 4, LUTs are connected in such a way that all the outputs feed into the inputs of the next level. However, this is not a mandatory requirement for the digital PUF. For example, feed forward structures can be used. Specifically, the inputs to a specific level of LUTs can come from or be controlled by the outputs from any previous level up to and including the primary inputs. Figure 6 illustrates two examples of feed forward structures for the digital PUF.

We explore the impact of input size on the digital PUF by measuring the success of the NIST tests for varying input sizes. Only when the digital PUF's output stream can pass all NIST tests do we claim that the configuration is acceptable. We begin with a digital PUF with 8 bit inputs and increment its input size by 8 after each test. We find that once the digital PUF reaches an input size of 32 bits it can consistently pass all NIST randomness tests.

We compare the different structures depicted in Figure 6 against the original digital PUF with varying levels of LUTs. We use the output hamming distance test from the avalanche criterion to compare the structures. Since the input size of each structures is 32 bits, the best structure will yield an average output hamming distance of close to 16.

Table 6 shows the results across the three types of structures. Two observations can be drawn from this table. The first is that the average hamming distance increases with more LUT levels for smaller sized digital PUFs but eventually stabilizes as the digital PUF grows. It can be concluded that after some growth, more levels would not significantly increase overall input diffusion. The second observation is that the feed forward structure in Figure 6b is the most secure of the three structures (in terms of satisfying the avalanche criterion) since its average output hamming distance reaches closest to 16.

5. APPLICATIONS

In this section we present two applications of the digital PUF. The first application is logic obfuscation. In this application we replace a portion of logic within a circuit with a digital PUF and a supporting programmable fabric in order to hide the functionality of the circuit and thus eliminate the possibility of reverse engineering attacks. The second application is trusted remote sensing which enables that the base station of a sensor network can fully trust that the data collected and transmitted from each node is valid and not tampered with, and furthermore, that the node itself has not been tampered with.

5.1 Hardware Logic Obfuscation

Reverse engineering attacks are so advanced nowadays that even integrated circuits containing on the order of 10^9 transistors can be reverse engineered in a matter of weeks. Techniques for hardware obfuscation attempt to prevent reverse engineering attacks by obscuring the functionality of

Levels	4	8	12	16	20	24
Original Structure	8.8 ± 0.6	12.7 ± 0.2	12.8 ± 0.5	13.1 ± 0.4	13.3 ± 0.5	12.9 ± 0.7
Feed Forward in Figure 6a	6.7 ± 0.4	10.9 ± 0.7	12.6 ± 0.6	13.3 ± 0.4	12.9 ± 0.5	13.1 ± 0.9
Feed Forward in Figure 6b	9.9 ± 0.9	13.9 ± 0.8	14.8 ± 0.8	15.1 ± 0.5	15.1 ± 0.5	14.8 ± 0.7

Table 6: Output hamming distance averages and standard deviations across 20 random instances of each digital PUF structure. The input size is 32 bits. Each column corresponds to a given number of LUT levels in the PUF structure.

a portion of logic within the circuit from an attacker while maintaining that the circuit performs its intended function.

Through the use of the digital PUF we demonstrate that we can obfuscate the functionality of a circuit by obscuring a portion of circuitry in such a way that the original circuit functionality is maximally difficult to reverse engineer. Specifically, we combine the digital PUF with a programmable fabric that, together, implement the originally intended functionality of the original circuitry while its function remains unknown. What is most unique about our approach in comparison to previous obfuscation techniques is that our digital PUF is able to integrate directly into the circuit and actually directly obfuscate logic.

5.1.1 Architecture

We obfuscate a piece of arbitrary logic by completely replacing it with a digital PUF and a supporting programmable fabric using the architectures shown in Figure 7. Obfuscation is accomplished by connecting the original logic inputs as the challenge to the digital PUF. The configurable fabric is necessary since the actual function of the digital PUF is configured post-fabrication. This is done by first characterizing the supporting delay-based PUFs that determine the digital PUF's LUTs. Characterization of these initialization PUFs is carried out as described previously in Section 3.1. Then, the digital PUF is configured following the procedure outlined in Section 4.1.

It is important to note that it is feasible to use only the standard delay-based PUF for logic obfuscation. However, given its limitations, it can only be applied using the architecture depicted in Figure 7a. This is due to two reasons. The first is because the standard delay-based PUF is unstable for some set of inputs. If the post-logic architecture from Figure 7b is used, it is possible that an input vector for which the analog PUF has an unstable output could arrive at the PUF in which case the obfuscated block would fail to produce the correct circuit functionality due to the PUF's instability. The second reason the analog PUF must be placed after the programmable fabric is because of its required arbiter which effectively acts as a flip-flop, thus ending the flow of logic for the given clock cycle. Assuming that an attacker can read this flip-flop and an attacker knows the structure of the configurable fabric, then his task is simplified to recording input-output pairs to build a representation of the PUF's functionality.

Since the digital PUF can be employed just like any other combinational component, it can be applied to the post-logic architecture depicted in Figure 7b. The biggest benefit of this architecture is that the PUF outputs cannot be measured directly as they can be in the pre-logic case. In the post-logic design, we can select the output wires of the obfuscated circuit in such a way that the remaining circuitry that the signals propagate through are difficult for an attacker to reverse engineer.

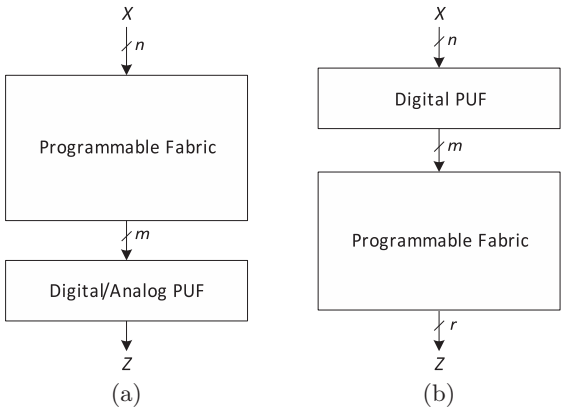


Figure 7: Hardware logic obfuscation architecture. (b) Pre-logic is required for the analog PUF to ensure input-output stability. (c) Post-logic is enabled through the use of the digital PUF since it is stable for all inputs.

Note that utilizing the digital PUF enables us to place the PUF anywhere in the circuit without the need of flip-flops or arbiters. Also note that flip-flops in this case are not primary inputs or outputs and cannot be directly controlled. Hence, we can obfuscate any connected subset of combinational circuitry anywhere in the design. In order to make the reverse engineering task difficult for an attacker, we select for replacement a portion of circuitry whose inputs that are difficult for the attacker to control as well as whose outputs are are difficult for the attacker to reverse engineer. We discuss the specifics of our heuristics in Section 5.1.3

In Figure 8 we present a motivational example using the s27 circuit from the ISCAS'89 benchmark suite [36]. In this example we obfuscate the circuitry consisting of the G9 and G11 gates. Note that by selecting this portion of circuitry for obfuscation we affect a portion of flip-flops which cannot be directly controlled, G5 and G6. In this case G6 is simply a direct output of the obfuscated block.

This is a small circuit with as many primary inputs as their are flip-flops (specifically, flip-flops that cannot be directly controlled). In larger circuits we find it is much easier to find portions of circuitry that are influenced by a larger majority of flip-flops than primary inputs and also affect a larger number of flip-flops than primary outputs.

Once the digital PUF is configured in Figure 8b we synthesize the configurable fabric to map the PUF outputs to the original replaced circuitry outputs using traditional FPGA design tools.

5.1.2 Attack

We assume that an adversary has complete knowledge of the design of the circuit even including knowledge of the design of the supporting configurable fabric. We assume

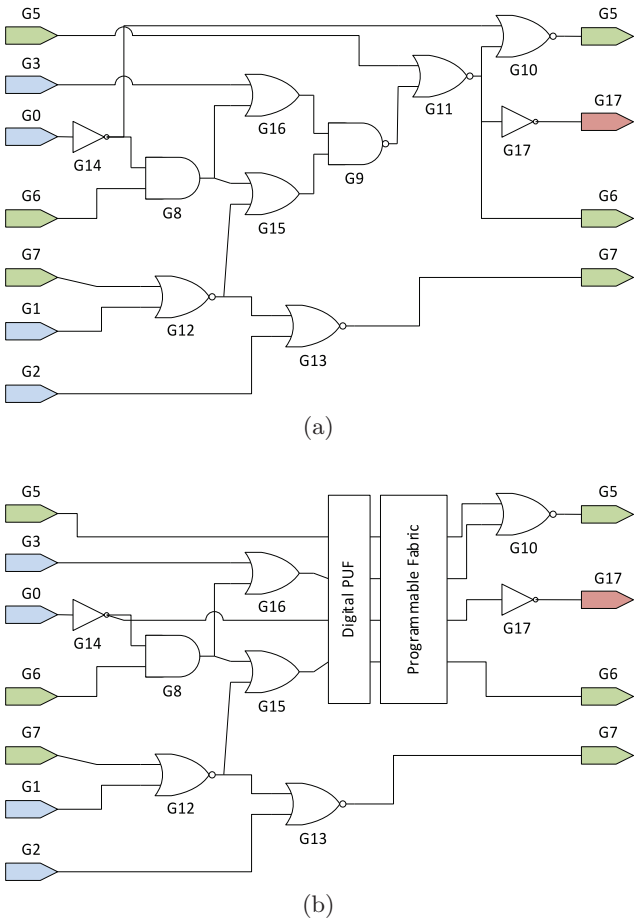


Figure 8: Motivational example using the (a) s27 circuit from the ISCAS'89 benchmark suite [36]. (b) Obfuscated form using the post-logic architecture from Figure 7b. The blue pins denote primary inputs. The red pin denotes a primary output. The green pins represent flip-flops.

that he has read access to all flip-flops in the circuit, but only write access to those flip-flops which are primary inputs to the system.

The job of the attacker is to reverse engineer the functionality of the entire circuit. Specifically, he will focus on the part that is obfuscated. Since we allow him to know the design of the configurable fabric, this leaves him with the task of fully characterizing the digital PUF.

This task is made more difficult by strategically selecting logic for obfuscation in such a way to reduce the attacker's ability to control the inputs to the obfuscating PUF as well to diffuse the outputs of the PUF before they arrive at a readable flip-flop.

5.1.3 Logic Selection

The digital nature of our PUF enables us to treat it as a combinational component. This gives us almost complete freedom to select any piece of arbitrary combinational logic for obfuscation. In assigning placement of obfuscated circuitry we consider the attack outlined above. Ultimately, the obfuscated logic is a black box in which the attacker can measure the inputs and outputs but is unaware of the internal functionality (e.g. digital PUF configuration). Thus, in logic selection for obfuscation we purposefully select a portion of logic for obfuscation whose inputs are difficult to control and whose outputs are as difficult as possible to measure. In this way we prevent an attacker from reconstructing a complete input-output switching expression of the obfuscated block.

Choosing inputs is accomplished by selecting wires that are dependent upon many flip-flops. By selecting the inputs to the obfuscated block in this manner we ensure that an attacker cannot directly control its input vectors. For example, in the obfuscated circuit example in Figure 8b, the obfuscated block is dependent upon six flip-flops, three of which cannot be directly controlled (G5, G6, G7) and two of which are also obfuscated (G5, G6). In order to reduce delay overhead we select sets of input wires which contain positive slack and whose ASAP and ALAP delays overlap.

The outputs of the obfuscated block are selected in such a way so as to maximize the reverse engineering task of the attacker. We assume that an attacker has full knowledge of the netlist of the circuit as well as read access to all flip-flops. In order to hide the outputs of the obfuscated block from the attacker we select output wires that combine together through regular circuitry into one flip-flop. This forces the attacker to reverse engineer the original output through the circuitry from a minimal amount of information

Note that the reverse engineering task is equivalent to the satisfiability (SAT) problem and is thus NP-complete. Hence, we increase the level of difficulty by selecting n obfuscated logic block outputs that combine maximally to k flip-flops, thus increasing the total number of clauses and variables comprising the SAT instance that must be solved by the attacker.

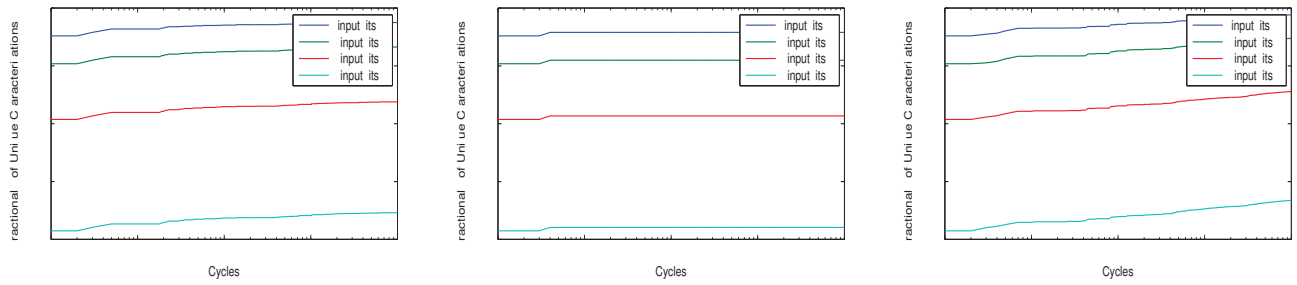
5.1.4 Obfuscation and Overhead

In this section we analyze and measure the overhead requirements and feasibility of attacking obfuscated circuits from the ICSACS'89 benchmark suite [36].

Figure 9 depicts the number of clock cycles required to fully characterize a fraction of input-output mappings of the pertinent obfuscated logic for three example benchmarks. In each case we analyze 100 different obfuscation configurations with the corresponding input bit size and plot the average number of characterized input-output mappings over time.

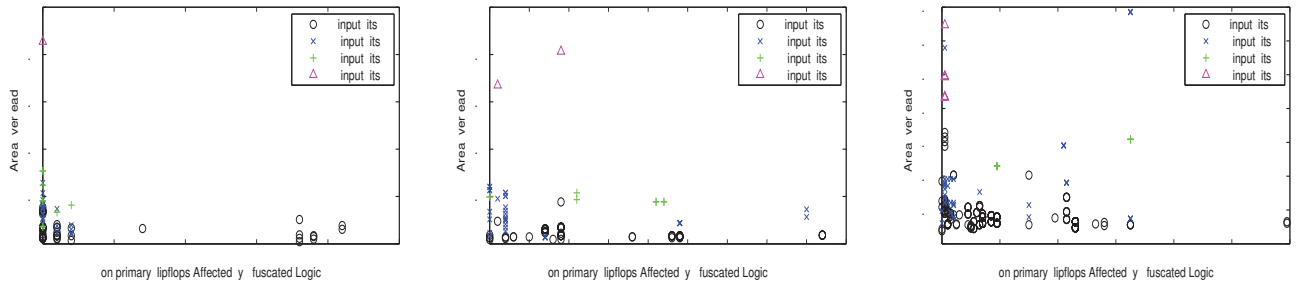
In these examples we assume an even more powerful attack than described above in which the attacker knows the output of the obfuscated block without the need to reverse engineer it. Note that even with this knowledge the number of characterized input-output mappings increases only linearly with an order of magnitude increase in cycles observed. Furthermore, by increasing the input size of the obfuscated logic block we reduce the absolute fractional number of input-output mapping characterizations by the same order of magnitude increase in input size, rendering complete specification of the obfuscated logic block infeasible.

We measure the area overhead required by the varying input sizes on different gates and depict the results in Figure 10 and Table 7. Our technique ensures that area overhead remains approximately the same order of magnitude for a given input set size in absolute terms, and thus, decreases tremendously with the size of the obfuscated circuit.



(a) (b) (c)

Figure 9: Fraction of correctly characterized PUF obfucated logic input-output mappings for the (a) s5378, (b) s9234, and (c) s38417 circuits from the ISCAS'89 benchmark suite [36].



(a) (b) (c)

Figure 10: Area overhead of circuit obfuscation as a fraction of the original size of a 90nm circuit for the (a) s5378, (b) s9234, and (c) s38417 circuits from the ISCAS'89 benchmark suite [36].

Circuit	Gates	Average Area Overhead			
		8	16	32	64
s1488	653	13.55 %	58.96 %	-	-
s5378	2,779	5.48 %	11.25 %	16.09 %	85.46 %
s9234	5,597	4.14 %	11.42 %	18.87 %	74.15 %
s35932	16,065	2.27 %	2.68 %	-	-
s38417	22,179	0.82 %	2.34 %	2.69 %	4.85 %

Table 7: Average area overhead for obfuscated logic with input sizes of 8, 16, 32, and 64 for the pertinent benchmark circuits. The dashed placeholders represent input set sizes that could not be found for the corresponding circuit.

5.2 Remote Trust

Trust is an essential component for many remote systems. It is even more essential for sensor networks which are often left unattended and installed in potentially hostile environments. The notion of trust in such systems enables that a communicating party know with certainty that a sensor node's data being transmitted has indeed been collected by that sensor which has not been tampered or compromised in any way. While public key cryptography ensures that no information is snooped over an insecure line, it does not protect against physical attacks to the sensing node. For example, if an attacker were to move a sensing node from its intended location, the node will continue to record and send its data over a secure channel to the base station, while the base station is unaware of the attack.

The key to enabling remote trust is through the integration of the system's core functionality along with pertinent parts of trustworthy circuitry with a PUF. The idea is that by combining the PUF with these data collecting elements (i.e. sensors, GPS, clock), any tampering of the PUF and/or data elements will affect the PUF outputs, effectively changing its functionality.

Previous approaches to trusted remote sensing utilize analog PUFs as the trust mechanism [13]. In addition to a susceptibility to environmental and operational variations, these devices are also susceptible to glitching. Since these devices are analog in nature, they rely on signal path propagation races throughout the PUF network. Between clock cycles and applications, some signals remain inside the PUF, ultimately affecting the next clock cycle. A zeroing procedure has yet to be presented for these architectures that is low in latency and effective at removing glitching between uses, however it is assumed that at least half of the throughput of the device is lost in practical operation since in at least every other clock cycle it is necessary that the PUF be zeroed, possibly more.

The digital PUF requires only a single cycle to initialize at power-up and only a single cycle to function and requires no additional clock cycles for resetting. Not only is it a low latency, high throughput, and low energy primitive, but it is also completely integrable with digital logic. Similar to our

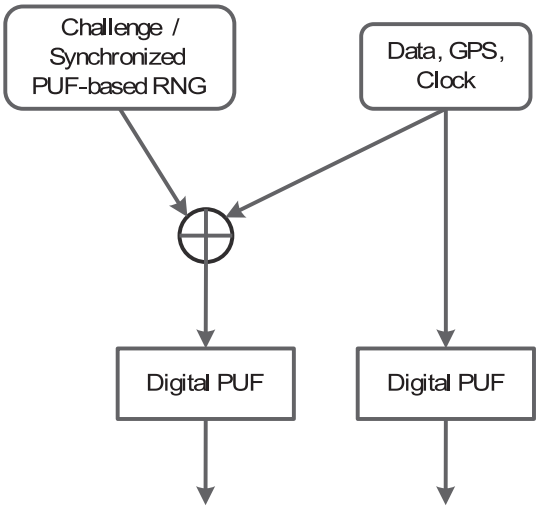


Figure 11: Trusted remote sensing computation flow at the sensor node. The base station provides the challenge.

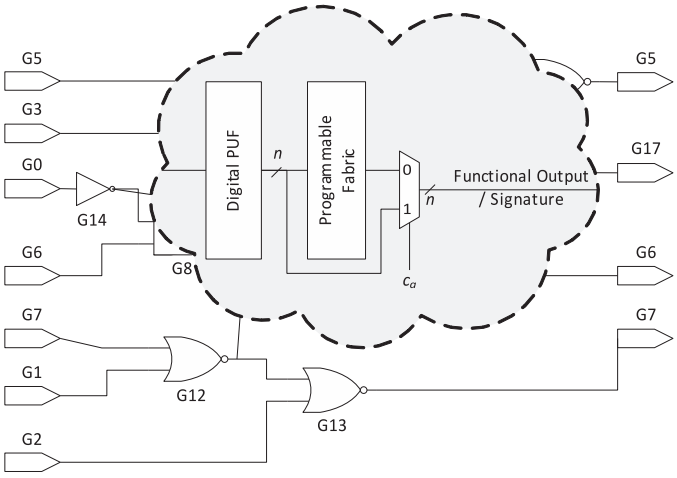


Figure 12: Variant of the hardware obfuscation architecture applied to the s27 benchmark suite enabling hardware attestation. The control signal c_a determine whether the circuit operates in normal functional mode or in attestation mode.

logic obfuscation application, we can place digital PUFs in the middle of digital logic, completely integrating with data collection elements such as sensory circuitry.

Figure 11 depicts the trusted remote sensing computation flow performed by each sensor node. The challenge provided to the sensor node can either be sent directly by the base station or—since the digital PUF passes all NIST tests—can be supplied using a digital PUF as a synchronized random number generator (RNG). At installation, the digital PUF-based RNG is synchronized with the digital PUF-based RNG at the base station. Since the system operator is the only entity that knows the functionality of both PUFs, only he can select seeds for each PUF that synchronizes their functionality. Note that the seeds are in fact challenge vectors to the analog parts of the digital PUF for initialization and hence, are unique. Thus, the seeds can even be made public since their use in initializing any other digital PUF would produce a completely different random number generator.

The remaining digital PUFs are also configured at installation and their functionality is recorded at the base station. At transmission time, the sensor node sends its data, timestamp, GPS coordinates, and two digital PUF outputs as illustrated in Figure 11. Since the base station knows the configuration of the sensor, it validates the data in a single cycle. A man-in-the-middle attack is easily caught since any alterations to any of the transmitted data will cause a vastly different PUF output which cannot be computed by an attacker without having reverse engineered the pertinent digital PUF.

5.2.1 Hardware Attestation

A requirement of the trusted remote sensing computation flow depicted in Figure 11 is that the digital PUF and circuitry (e.g. sensing, GPS, clock) must be physically coupled together to prevent physical attacks. Specifically, the boundaries between these two components should not be easily read or, more importantly, written to by an attacker.

We enable physical coupling by incorporating a variant of our hardware obfuscation architecture into the pertinent circuit as depicted in Figure 12. The control signal c_a selects whether the circuit operates in functional mode, in which the circuit operates as normal, or attestation mode, in which the circuit outputs a unique signature for verification.

The most important property of the generated signature is that it simultaneously and uniquely entangles the input data, logic, and digital PUF into one signal, and does so in a single clock cycle. Since the digital PUF functionality is known only by the remote operator, only he can verify the output signature given the inputs. By integrating this architecture into pertinent components of the device (e.g. sensing circuitry, GPS, clock), we enable that these components and their outputs can be remotely trusted.

6. CONCLUSION

We have presented a digital PUF which leverages a stable delay-based PUF for initializing its connected network of LUTs. Stability in the delay-based PUF is ensured by selecting challenges that have a delay ratio of 10% which ensures that the output is always stable for temperatures ranging from 250K to 400K. We demonstrated the security properties of the digital PUF by passing all benchmark tests from the NIST randomness suite, passing the avalanche criterion, and subjecting the digital PUF to a host of statistical attacks. Finally, we have demonstrated the application of the digital PUF in a remote secret key exchange protocol in which both communicating parties experience very low overhead in terms of both time and energy. Furthermore, while previous PUFs have been designed for remote enabling and communication tasks, the digital PUF enables the actual implementation of logic. We use this capability in a novel logic obfuscation technique and demonstrate that for large circuits we can successfully obfuscate circuit functionality while imposing less than 5% overhead in terms of area.

7. ACKNOWLEDGEMENTS

This work was supported in part by the NSF under award CNS-0958369, award CNS-1059435, and award CCF-0926127, and by Samsung under award GRO-20130123.

8. REFERENCES

[1] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," tech. rep., DTIC Document, 2001.

[2] C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[3] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *HOST*, pp. 1–6, 2013.

[4] J. Valamehr *et al.*, "Inspection resistant memory: architectural support for security from physical examination," in *ACM SIGARCH Computer Architecture News*, vol. 40, pp. 130–141, 2012.

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

[6] B. Gassend *et al.*, "Silicon physical random functions," in *Computer and Communications Security*, pp. 148–160, 2002.

[7] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *CHES*, pp. 63–80, 2007.

[8] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *DAC*, pp. 9–14, 2007.

[9] J. W. Lee *et al.*, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Symposium on VLSI Circuits*, pp. 176–179, 2004.

[10] S. Devadas *et al.*, "Design and implementation of PUF-based 'unclonable' RFID ICs for anti-counterfeiting and security applications," in *IEEE International Conference on RFID*, pp. 58–64, 2008.

[11] E. Simpson and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *CHES*, pp. 311–323, 2006.

[12] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *USENIX Security Symposium*, pp. 291–306, 2007.

[13] M. Potkonjak, S. Meguerdichian, and J. L. Wong, "Trusted sensors and remote sensing," in *IEEE Sensors*, pp. 1104–1107, 2010.

[14] J. B. Wendt and M. Potkonjak, "Nanotechnology-based trusted remote sensing," in *IEEE Sensors*, pp. 1213–1216, 2011.

[15] G. E. Suh *et al.*, "Design and implementation of the AEGIS single-chip secure processor using physical random functions," in *ACM SIGARCH Computer Architecture News*, vol. 33, pp. 25–36, 2005.

[16] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Information Hiding*, pp. 206–220, 2009.

[17] U. Rührmair, "SIMPL systems, or: can we design cryptographic hardware without secret key information?," in *SOFSEM*, pp. 26–45, 2011.

[18] T. Xu, J. B. Wendt, and M. Potkonjak, "Digital bimodal function: an ultra-low energy security primitive," in *ISLPED*, pp. 292–296, 2013.

[19] M. Fyrbiak, C. Kison, and W. Adi, "Construction of software-based digital physical clone resistant functions," in *International Conference on Emerging Security Technologies*, pp. 109–112, 2013.

[20] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 2, no. 1, p. 5, 2009.

[21] U. Rührmair *et al.*, "Modeling attacks on physical unclonable functions," in *Computer and Communications Security*, pp. 237–249, 2010.

[22] X. Xu and W. Burleson, "Hybrid side-channel/machine-learning attacks on PUFs: a new threat?," in *DATE*, p. 349, 2014.

[23] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side-channel(s)," in *CHES*, pp. 29–45, 2003.

[24] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *CHES*, pp. 2–12, 2003.

[25] J. A. Halderman *et al.*, "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009.

[26] Y. Ren, Y. Shi, and B.-H. Gwee, "A novel gate-level to behavior-level conversion algorithm with high microcell identification rate," in *IASTED International Conference*, vol. 712, p. 138, 2010.

[27] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *DAC*, pp. 83–89, 2012.

[28] J. Zheng and M. Potkonjak, "DPUF: A reconfigurable IP protection architecture for embedded systems," in *ANCS*, pp. 1–2, 2014.

[29] W. Huang *et al.*, "Hotspot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[30] "Implementation of security in Actel's ProASIC and ProASICPLUS flash-based FPGAs." `http://www.actel.com/documents/Flash_Security_AN.pdf`, 2003.

[31] D. H. Kim, K. Athikulwongse, and S. K. Lim, "A study of through-silicon-via impact on the 3D stacked IC layout," in *ICCAD*, pp. 674–680, 2009.

[32] T. Huffmire *et al.*, "Hardware trust implications of 3-D integration," in *Proceedings of the 5th Workshop on Embedded Systems Security*, p. 1, 2010.

[33] J. Valamehr *et al.*, "A qualitative security analysis of a new class of 3-D integrated crypto co-processors," in *Cryptography and Security: From Theory to Applications*, pp. 364–382, 2012.

[34] S. Briais *et al.*, "3D hardware canaries," in *CHES*, pp. 1–22, 2012.

[35] U. Guvenc, "Active shield with electrically configurable interconnections," in *SECUREWARE*, pp. 43–45, 2013.

[36] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *ISCAS*, pp. 1929–1934, 1989.