Lightweight Secure PUFs

Mehrdad Majzoobi and Farinaz Koushanfar
Electrical and Computer Engineering Department
Rice University
6100 Main, MS380, Houston, TX 77005
Email: {mehrdad.majzoobi,farinaz}@rice.edu

Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
3532G Boelter Hall, Los Angeles, CA 90095
Email: miodrag@cs.ucla.edu

***Abstract*— To ensure security and robustness of the next generation of Physically Unclonable Functions (PUFs), we have developed a new methodology for PUF design. Our approach employs integration of three key principles: (i) inclusion of multiple delay lines for creation of each response bit; (ii) transformations and combination of the challenge bits; and (iii) combination of the outputs from multiple delay lines; to create modular, easy to parameterize, secure and reliable PUF structures. Statistical analysis of the new structure and its comparison with existing PUFs indicates a significantly lower predictability, and higher resilience against circuit faults, reverse engineering and other security attacks.**

I. Introduction

Creation of security mechanisms and protocols is particularly challenging for embedded systems. From one viewpoint, embedded systems are often under strict power, cost, and size constraints and, therefore, must employ only light-weight security protocols. From another viewpoint, they are often mobile and physically unprotected. Thus, they easily permit physical attacks.

Silicon PUFs leverage intrinsic manufacturing variability of deep submicron technology to create single cycle, low power, and area efficient security mechanisms. Since each PUF is unique, the same challenge produces different responses on various chips. The effectiveness of PUFs has been demonstrated for both traditional security tasks such as authentication [5] as well as for digital rights management tasks including FPGA security, remote enabling and disabling, n-variant IC design, and proof of software execution on a certain processor [7], [4], [1], [9], [16], [15], [11], [2]. Several different PUF structures were introduced and realized on both FPGA and ASICs [8], [7]. Unfortunately, it has been demonstrated that the simple delay-based path comparison is easy to reverse engineer, or predict, and is often sensitive to operational conditions and imperfections of the arbiters [5]. A number of fixes have been suggested, including usage of nonlinearity such as feed forward arbiters [8], and interfacing hash functions to the PUF chanllege/responses [5]. The previous solutions have a limited effectiveness, in particular for lightweight embedded systems applications and are suspectable to a range of attacks [13].

To overcome these limitations and to realize the full potential of PUFs as a basis for security of lightweight systems, we have developed a new family of PUF structures and methodology for their design and analysis. We employ three new security principles for designing secure and robust PUFs. First, we use multiple delay lines for creation of each response. Second, we use judicious combination of the challenge input bits to drastically reduce controllability. Finally, we subject the outputs from multiple delay lines to a scrambling lossy transformation to create modular, easy to parameterize, secure, and reliable PUF structures.

II. Background

Gassend et al. proposed the parallel delay-based PUF circuit shown in Figure 1 [5]. Generating one bit of output requires a signal to travel through two parallel paths with multiple segments that are connected by a series of 2-input/2-output switches. Each switch is configured to be either a cross or a straight connector, based on its selector bit. The path segments are designed to have the same nominal delays, but their actual delays differ slightly due to manufacturing variability. The difference between the top and bottom path delays are compared by an *aribter*. The PUF challenges (inputs) are the selector bits of the switches. The output bit of the arbiter depends on the challenges and is permanent for each IC, at least for a range of operational conditions. Parallel PUF's liability to reverse engineering was previously addressed [5] by introducing nonlinearities, such as feed forward (FF) arbiters, in the PUF structure. Figure 1 shows a FF arbiter in dashed line that controls a forward switch selector. Unfortunately, our prior studies showed that even this structure can be reverse engineered using a combination of combinatorial and linear programming techniques [13].

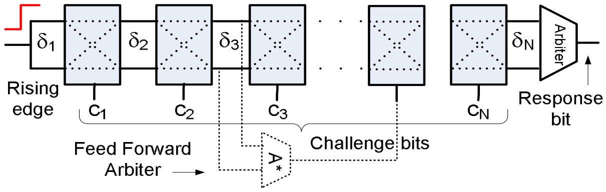

Fig. 1. Parallel PUF structure. The feed forward arbiter (shown in the dashed line) is used to introduce nonlinearity.

III. Analysis of PUF vulnerabilities

In this section, we present the potential PUF security vulnerabilities. We discuss the existing countermeasures along with the merits and drawbacks associated with each. In the interest of brevity, we focus on reverse engineering attack and refer the interested readers to [12] for a more comprehensive study on emulation, statistical modeling and man-in-the-middle attacks.

Reverse engineering: Reverse engineering attacks aim at calculating a group of parameters (e.g., path segment delays) that fully model the PUF behavior, and develop a software counterfeit or emulation model of the PUF. To recover the model parameters, the adversary needs to obtain a polynomial number (with respect to the number of delay elements in PUF) of challenge-response pairs. When considering the delay based PUFs, one can efficiently represent each switch using only two parameters. When the switches are connected in series, the adjacent parameters can be lumped together and be expressed by a new parameter. We use δ to denote the new parameter and refer to it as the (differential) path segment delay. Thus, the PUF challenge/response relationship can be formally expressed by

$$\sum_{i=1}^{N}(-1)^{\rho_i^N}\delta_i + \delta_{N+1} \underset{r=1}{\overset{r=0}{\lesseqgtr}} 0. \tag{1}$$

Where a transformation $\mathbf{T}$ defined as

$$\rho_i^j = \bigoplus_{x=i,i+1,\ldots,j} c_x = c_i \oplus c =_{i+1} \oplus ... \oplus c_j \qquad (2)$$

maps the challenges to ρ's ($i \leq j$). $\bigoplus$ and $\oplus$ denote the parity generation and exclusive-or operations respectively; c_i corresponds to the i_{th} challenge bit in the challenge vector $\bar{c} \in \mathbb{B}^N$, $\mathbb{B}$ ={0, 1}. The inequality direction is determined by the PUF response, r, for the given challenge vector. Therefore, each challenge-response pair (CRP) forms an inequality. By collecting enough CRPs, one can build and solve a system of linear inequalities to estimate the δ's. Reverse engineering of linear PUF has been addressed earlier [6]. Several different methods were proposed to fortify the PUF against such attacks, including use of (i) non-linearity, and (ii) challenge-response hashing [8], [5].

(i) ***Non-linear PUFs:*** The proposed non-linearities are mainly of two types: (a) feed forwarding and (b) MAX (MIN) operations. Detailed analysis of non-linear PUF vulnerabilities was given in [13].

(ii) ***Challenge/Response Hashing:*** To make the parallel PUF responses obfuscated and obscure, a one-way hash function can be placed immediately after the arbiters [5]. To discover the original response, one needs to invert the one-way function that is known to be a hard problem. This process should also be repeated until sufficient number of responses are collected. Another hash function is attached to PUF challenges to prevent from controlling the challenges directly. Due to the confusion and diffusion properties of hash functions, the final system is safe against emulation, reverse engineering and statistical modeling attacks. Note that hash functions have significant hardware and power-overhead and their evaluation takes many clock cycles, imposing a large latency on the system. Table I shows latency (in cycles) and area (in gates) of commonly used hash functions [10]. Also, the adopted key-based hash functions are susceptible to attacks on digitally stored keys, e.g., side-channel attacks [3].

TABLE I
LATENCY AND AREA OF COMMON HASH FUNCTIONS.

Algorithm	Chip area	Clock cycles
SHA-256	10,868	1,128
SHA1	8,120	1,274
MD5	8,400	612
MD4	7,350	456

IV. SECURE PUF

In this section, we introduce the design methodology of the first secure and robust PUF structure. The proposed PUF, shown in Figure 2, consists of four fundamental building blocks: (i) input (logic) network, (ii) output logic network, (iii) wire interconnect network (iv) parallel PUF.

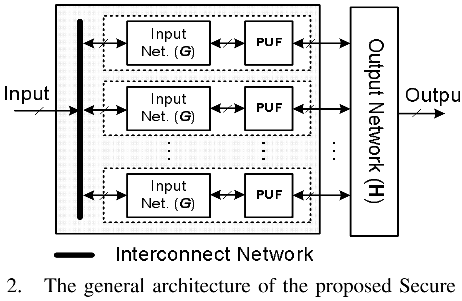

Fig. 2. The general architecture of the proposed Secure PUF.

A. Input Network

We design the input network connected to the parallel PUF's challanges (the dashed box in Figure 2) to satisfy the Strict Avalanche Criterion (SAC) for a parallel PUF circuit. A function is said to satisfy SAC if, whenever a single input bit is complemented, each of the output bits should change with a probability of one half. Later, we will show that the SAC property combined with a scrambling output network and special input interconnections result in a secure and robust PUF structure. We start by analyzing the avalanche property of the linear delay-based PUF. As stated in Section III, the PUF behavior can be represented by Equation 1. Let us assume the differential delay values (δ) in Equation 1 come from independent and identically distributed Gaussian random variables with zero mean, i.e., $\delta_i \sim \mathcal{N}(0, \sigma^2)$. Gaussian and independence assumptions only simplifies the proof and can be removed without altering the results. Our goal is to find the probability that the PUF output flips given that a challenge bit in the PUF input is flipped, i.e., $Prob\{\sim O | \sim c_k\}$. Whenever a challenge bit value flips, some of the terms in Equation 1 change sign (as a result of a change in the corresponding ρ values). We denote the set that contains the indices of ρ's that (do not) flip as result of a flip in the k_{th} challenge bit by Γ_k (Λ_k). If the absolute value of the sum of terms whose indices are in Γ_k is greater than the absolute value of the sum of terms whose indices are in Λ_k, then the summation changes sign (i.e. output flips) whenever c_k flips. It can be proved (see [12]) that if,

$$|\ \Gamma_k\ | = \frac{N+1}{2} \qquad (3)$$

then (almost) half of ρ's in Equation 1 flip as a result of a flip in k_{th} challenge bit (c_k), and the output of the PUF would flip with a probability of 0.5 ($E\{X_k\}$ = 0.5). The result is in accordance with our initial intuitive observation.

We now verify whether this property holds for the parallel PUF structure. The ρ's in Equation 1 are related to challenges by the transformation T defined in Equation 2, i.e., $\mathbf{P} = T(\mathbf{C})$. It can be seen that a flip in c_k causes a flip in ρ_j where $j \leq k$. Thus $|\Gamma_k| = k$. For example, if a flip in c_N happens, all of the ρ's flip as a result. Hence, Equation 3 is not satisfied for the parallel PUF structure. We define a transformation $G(.)$ on challenges that combined with T meets the criterion set by Equation 3.

Goal: Find $G(.)$ so that $\mathbf{P} = T(G(\mathbf{C}))$ satisfies $|\Gamma_k| = \frac{N+1}{2}$ for all k.

Solution: We have derived constraints on the challenge bits for guaranteeing SAC such that whenever a challenge bit flips, another challenge bit at $\frac{N+1}{2}$ locations apart also flips [12]. It is infeasible to impose the exact $\frac{N+1}{2}$ distance constraint on the PUF challenges, however, high quality approximations can be made. For N an even integer and $M = N$, G performs the following transformation:

$$\begin{aligned} c_{\frac{N+i+1}{2}} &= d_i, \quad for \ \ i = 1 \\ c_{\frac{i+1}{2}} &= d_i \oplus d_{i+1}, \quad for \ \ i = 1, 3, 5, ..., N-1 \\ c_{\frac{N+i+2}{2}} &= d_i \oplus d_{i+1}, \quad for \ \ i = 2, 4, 6, ..., N-2 \qquad (4) \end{aligned}$$

The logic network shown in Figure 3 can carry out this transformation. An adversary with the full knowledge of the circuit structure can apply the inverse transformation to make the input network ineffective. We alleviate this issue by introducing a wire interconnecting method that physically binds the inputs of multiple PUF rows. In addition to the expectation of X_k being equal to 0.5, it is desired that the X_k has as small variance as possible. Smaller variation guarantees that SAC is satisfied by a larger number of

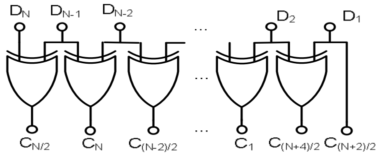

Fig. 3. The input network realization using XOR logic.

PUFs. The variance of X_k is related to the number of switches and the variance of δ that is determined by the technology and process variation. Therefore, one can achieve a smaller variance for X_k by adding to the number of switches or by incorporating multiple rows of the same structure (Section IV-B).

B. Output Network

We introduce an XOR-based output network structure (see Figure 2) that achieves: (i) fortification against reverse engineering attacks, and (ii) higher randomness of responses by combining multiple rows of parallel PUFs with the transformed challenges.

The output network performs a mapping denoted by H(.) from PUF arbiter responses, $\mathbf{R}$, to the output, $\mathbf{O}$. The mapping is defined as $\mathbf{O}$ = $H(\mathbf{R})$, $H : \mathbb{B}^Q \rightarrow \mathbb{B}^{Q'}$ where $\mathbb{B}$ = $\{0, 1\}$ and $Q' < Q$, and

$$o_j = \bigoplus_{i=1,\ldots,x} r_{(j+s+i)\ mod\ Q} \quad for\ j = 1, 2, ..., Q' \tag{5}$$

where $\bigoplus$ denotes the parity generator function and s indicates shifting step. The transformation calculates the parity value for sets of x-adjacent PUF arbiter responses where sets are circularly shifted by s bits with respect to each other. The transformation can be parameterized by s (the shifting step) and x (the parity input size). We will discuss later how these parameters govern a trade-off among security, overhead, and randomness properties.

The proposed transformation can hinder the efforts to reverse engineer the PUF. To reverse engineer a linear PUF structure, the adversary needs to collect a set of challenge-responses from the PUF.

Suppose that the responses of Q parallel PUFs are mapped to a Q'-bit output. Then there are $2^{Q-Q'}$ possible inputs that map to a given output. Therefore, the adversary is faced with solving an ambiguity to discover the real PUF response and has to deal with a branching problem of making assumptions. The number of assumptions grows exponentially, if the adversary is not able to reject some of them at each step. Nevertheless, if one can control (the transition of) the PUF arbiter responses or obtain partial information about arbiter responses, then the number of assumptions can be reduced. We illustrate the problem using an example. In Figure 4 (b), four possible transition hypotheses about the H inputs are shown by arrows. If we knew that only the first bit (or only one bit) of the input has caused such transition in the output, then we could reject the two hypotheses shown by dashed arrows. Also by associating probabilities with transitions and ranking assumptions accordingly, one can guess the PUF responses.

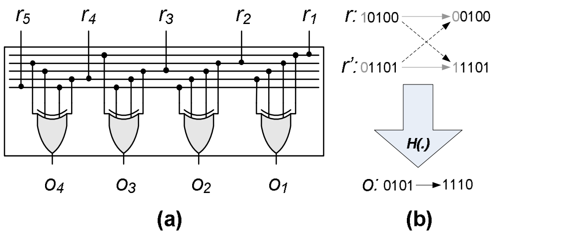

Fig. 4. An example of output network for Q = 5, Q' =4, x = 4 and $s = 1$.

C. Interconnect Network

The interconnect network connects the challenge bits of rows of parallel PUFs (the leftmost solid box in Figure 2). To satisfy the SAC, it is required and sufficient that one challenge bit on each row is connected to another challenge bit on a different row. A challenge bit is broadcasted to all of the PUFs, and since each PUF output flips with a probability of 0.5, the SAC is met. The interconnection rule can be expressed formally as follows.

$$c_i^m = c_j^{m+1} \quad for \quad i, j \in \Omega, \quad m = 1, 2, ..., Q - 1 \tag{6}$$

where c_i^m is the $i - th$ challenge bit in the $m - th$ row, Ω = $\{1, 2, ..., N\}$, and $j = g_m(i)$, g: $\Omega \rightarrow \Omega$ is a one-to-one permutation function. Recall that in Section IV-A we mentioned that the XOR input network can be bypassed by applying the inverse transformation. If the inputs of PUF rows are connected in parallel (with no permutation), i.e., i=j, by applying the inverse transformation (G^{-1}) all of the input networks can be bypassed and, thus, become ineffective. By imposing a constraint on g_m to be non-identity for all m, the attacker can *fully* bypass only one input network. Figure 5 depicts an m-bit circular shift interconnecting method, i.e., j = $g_m(i)$ = $(i + m - 1)$ mod Q.

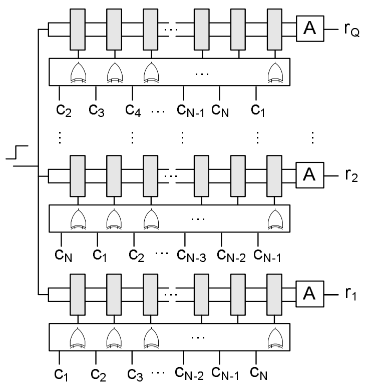
A
r_Q
C_2 C_3 C_4 ··· C_{N-1} C_N C_1
A
r_2
C_N C_1 C_2 ··· C_{N-3} C_{N-2} C_{N-1}
A
r_1
C_1 C_2 C_3 ··· C_{N-2} C_{N-1} C_N

Fig. 5. An m-bit circular shift interconnecting method that interfaces Q rows of parallel PUFs with the transformed challenges.

V. Experimental Results

In the experiments, we model each switch with four delays - two for the straight connection and two for cross connection links and assume that the delay components are samples from independent identical Gaussian distributions with μ = 0.5 ns and σ = 4ps. The mean and variance are for the 65nm technology [14].

For a single row parallel PUF circuit with 64 switches, we simulated the probability of output transition conditioned on each challenge bit transition. Figure 6 shows the value of $E[X_k]$ before and after applying the input XOR transformation (defined in Equation 4) on the PUF challenges. The figure shows that the probability of output flipping conditioned on the k-th challenge bit before input transformation increases monotonically from less than 0.1 to over 0.9, where k = 1,2,...,64. Note that after applying the XOR transformation on the PUF challenges, the output flips with a probability of close to 0.5 after a flip in input bits, which meets the SAC.

Smaller deviation from transition probability of one half is desired for each individual PUF circuit realization. There are two ways to reduce such deviation; (a) to use more switches in the parallel PUF circuit (increasing N); (b) to mix the outputs of higher number of parallel PUF circuits (increasing x in Equation 5). The black solid line in Figure 7 indicates how the variance ($var(X_k)$) decreases as the number of switches in single parallel PUF increases from 8 to

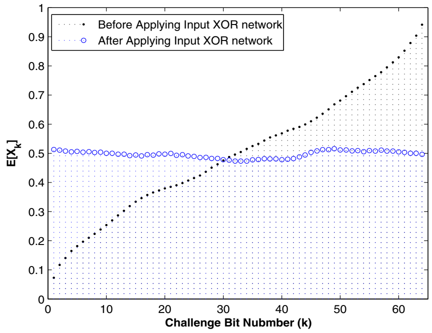

Fig. 6. Transition probability of a 1-bit parallel PUF. The dot (circular) markers show the probability before (after) transforming the challenges.

128. For a fixed number of switches in a row, the variance rapidly drops as 2, 4, and 8 adjacent outputs of rows of parallel PUFs (x = 2,4,8) are mixed. It can be seen that having 32 switches in each row is sufficient for obtaining almost the smallest possible variance.

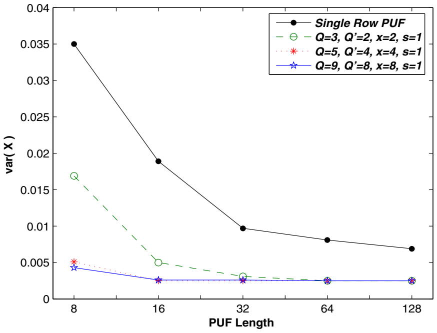

Fig. 7. Deviation of transitional probabilities of individual PUF instances from the SAC.

Delay outliers can cause high predictability, high compressibility of challenge response-pairs, and facilitate building of statistical models. We studied the sensitivity of the secure PUF and the single row parallel PUF structures to outliers. A fault is injected as an outlying delay of 5ns into the 20-th switch of the first row for the secure PUF. Figure 8 shows the expected probability of output transition for both single row parallel PUF and the secure PUF with parameters (Q,Q',x,s) = (9,8,8,1). The expectation is taken over 50 PUF realizations. For the parallel PUF with one row, the transition probability is highly distorted; However the transition probability of output (any of the eight PUF output bits) does not change because of the mixing by the output network.

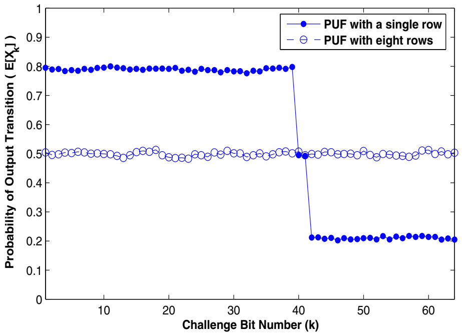

Fig. 8. Sensitivity of PUF transitional behavior to outlier switch delays.

TABLE II
NO. OF XOR GATES IN THE OUTPUT NETWORK

Q	*9*	*9*	*9*	*17*	*17*	*17*	*33*	*33*
Q'	*5*	*8*	*8*	*8*	*16*	*16*	*32*	*32*
x	*8*	*8*	*4*	*16*	*16*	*8*	*16*	*8*
s	*2*	*1*	*1*	*2*	*1*	*1*	*1*	*1*
XOR	*19*	*28*	*17*	*43*	*75*	*52*	*139*	*100*

Table II shows the number of XOR gates needed to construct the output network for different parameters. The number of XOR gates required to build the inputs networks is equal to number of switches times the number of rows ($N \times Q'$). For $N = 64$ and $Q' = 8$, the input net can be made with 512 XOR gates.

VI. CONCLUSION

We have developed a new family of physically unclonable functions (PUFs) that are resilient against reverse engineering, emulation, guessing, and many other security attacks, while they are robust to circuit aging and operational conditions. The task is accomplished by using three key principles: (i) mixing multiple delay lines; (ii) transformations of the challenge bits; and (iii) combination of the outputs from multiple lines. The new structures are low in area, power, and delay overheads. They facilitate easy security versus implementation cost trade-offs. Comprehensive simulations and statistical analysis support our conceptual and synthesis claims.

VII. ACKNOWLEDGEMENT

This work was partly supported by the DARPA/MTO Trust in ICs and Young Faculty Awards (YFA) under grant award W911NF-07-1-0198 and NSF CT-0716674.

REFERENCES

[1] Y. Alkabani and F. Koushanfar. Active hardware metering for intellectual property protection and security. In *USENIX Security*, pages 291–306, 2007.

[2] Y. Alkabani and F. Koushanfar. N-variant IC design: Methodology and applications. In *DAC*, pages 546–551, 2008.

[3] R.J. Anderson. *Security Engineering: A guide to building dependable distributed systems*. John Wiley and Sons, 2001.

[4] L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in RFID systems. In *PERCOM*, pages 211–220, 2007.

[5] B. Gassend et al. Silicon physical random functions. In *CCS*, pages 148–160, 2002.

[6] B. Gassend et al. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience. John Wiley & Sons*, 16(11):1077–1098, 2004.

[7] J. Guajardo et al. FPGA intrinsic PUFs and their use for IP protection. In *CHES*, pages 63–80, 2007.

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

[9] Y. Alkabani et al. Remote activation of ICs for piracy prevention and digital right management. In *ICCAD*, pages 674–677, 2007.

[10] M. Feldhofer and C. Rechberger. A case against currently used hash functions in RFID protocols. In *Workshop on RFID Security*, pages 372–381, 2006.

[11] F. Koushanfar and M. Potkonjak. Cad-based security, cryptography, and digital rights management. In *DAC*, 2007.

[12] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure puf design for embedded systems. In *Technical report* - TREE0803, 2008.

[13] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *ITC*, 2008.

[14] P. Sedcole and P. Y. K. Cheung. Within-die delay variability in 90nm FPGAs and beyond. In *FPT*, pages 97–104, 2006.

[15] G. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14, 2007.

[16] S. Trimberger. Trusted design in FPGAs. In *DAC*, pages 5–8, 2007.