IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 2 May 2013; revised 9 October 2013; accepted 18 December 2013. Date of publication 9 April 2014;
date of current version 30 July 2014.

Digital Object Identifier 10.1109/TETC.2014.2316497

Obtaining Statistically Random Information From Silicon Physical Unclonable Functions

CHI-EN YIN AND GANG QU (Senior Member, IEEE)

Department of Electrical and Computer Engineering, Institute for Systems Research, University of Maryland, College Park, MD 20742, USA

CORRESPONDING AUTHOR: G. QU (gangqu@umd.edu)

This work was supported in part by the National Science Foundation of China under Grant 61228204, in part by the Air Force Office of Scientific Research under Grant FA95501010140, and in part by the University Partnership with the Laboratory of Telecommunications Sciences under Contract H9823013D00560002. The work of C.-E. Yin was supported by the Taiwan Merit Scholarships through the National Science Council of Taiwan under Grant NSC-095-SAF-I-564-056-TMS.

ABSTRACT Silicon physical unclonable functions (PUFs) utilize the variation during silicon fabrication process to extract information that will be unique for each chip. There have been many recent approaches to how PUF can be used to improve security-related applications. However, it is well known that the fabrication variation has very strong spatial correlation[1] and this has been pointed out as a security threat to silicon PUF. In fact, when we apply NIST's statistical test suite for randomness against the random sequences generated from a population of 125 ring oscillator PUFs using classic 1-out-of-8 coding and neighbor coding, none of them can pass all the tests. In this paper, we propose to decouple the unwanted systematic variation from the desired random variation through a regression-based distiller, where the basic idea is to build a model for the systematic variation so we can generate the random sequences only from the true random variation. Applying neighbor coding to the same benchmark data, our experiment shows that second- and third-order polynomials distill random sequences that pass all the NIST randomness tests. So does fourth-order polynomial in the case of 1-out-of-8 coding. Finally, we introduce two generic random sequence generation methods. The sequences they generate fail all the randomness tests, but with the help of our proposed polynomial distiller, all but one tests are passed. These results demonstrate that our method can provide statistically random PUF information and thus bolster the security characteristics of existing PUF schemes.

INDEX TERMS Ring oscillator (RO), physical unclonable functions (PUFs), linear regression, variation decomposition.

I. INTRODUCTION

A. OVERVIEW

One of the most renowned principles for the design of a cryptosystem is Kerckhoffs's law: "A cryptosystem should be secure even if everything about the system, except the key, is public knowledge (1883)." In order to provide a secure storage for cryptographic keys, contemporary tamper-resistant devices such as smart cards arm themselves with a number of countermeasures to defeat various kinds of invasive, semi-invasive and non-invasive physical attacks. [6]–[11]. Nevertheless, it is still possible for attackers to read, and possibly write, the secret bits in the non-volatile memory through the electron beam of a Scanning Electron Microscope (SEM) once the surface of the chip is exposed by, for instance, Focused Ion Beam (FIB) [12], [13]. Physical unclonable functions (PUFs), in contrast, are 'inseparable' because the underlying nano-scale structural disorder will most likely be damaged during the course of physical tampering of the device, so will the keys [14]. Since the first introduction of PUFs in [15], many types of circuitry have been proposed to realize the notion. Most notable are Arbiter PUFs [16], [17] RO PUFs [3], [16] and SRAM PUFs [18], [19]. Many methodologies have been proposed to advance the art in terms of reliability, [4], [5], [20], [21] security, [4], [5], [17], [20], [22]–[28] and hardware efficiency [5], [20], [21], [29]–[31].

Researchers further classify PUFs as 'Strong', 'Controlled', and 'Weak' mainly according to the number of

[1]Spatial correlations and systematic fabrication variations referred hereafter are different for each PUF.

challenge-response pairs (CRPs) a PUF can generate, where the words *weak* and *strong* are irrelevant to the strength of PUF security [27]. Considering that a large number of CRPs can be achieved through a keyed hash function seeded by a Weak PUF, we chooses Weak PUFs as the context of the discussion, though the proposed methodology is expected to work well with Strong PUFs in a similar fashion.

Figure 1 outlines the typical workflow of a Weak RO PUF that involves the following steps.

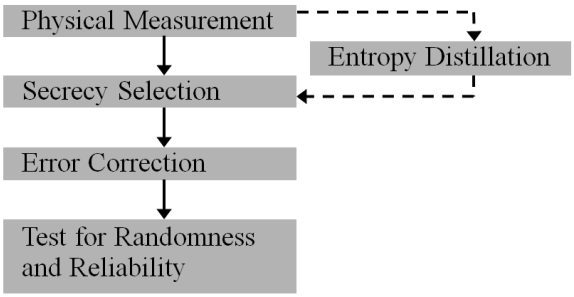

FIGURE 1. The typical workflow of a Weak RO PUF.

1) FABRICATION VARIATION EXTRACTION

The very first task of PUFs is to measure the unique characteristics endowed from the uncontrollable fabrication process. The analog-to-digital transformation is part of the physical entropy source subject to tests and in our case, this step corresponds to a full frequency characterization of a RO array [2].

2) SECRECY SELECTION

This step selects secure and reliable secrecy out of the variation profile measured in the previous step. Existing approaches include the classic 1-out-of-8 Coding [3] and its successor Index-Based Syndrome (IBS) Coding [4], Chain-like Neighbor Coding [2], [5], [32], Temperature-Aware Cooperative (TAC) Coding [29] and Group-Based Coding [30], [31].

3) ERROR CORRECTION

In addition to the above error prevention methods, error correcting codes (ECC) are applied to further enhance reliability. Codes that have been used for RO PUFs include Hamming Code, BCH Code [3], Repetition Code [4], [21], Reed-Muller Code [20], [21], and Kendall Syndrome Code [31].

4) TESTS FOR RANDOMNESS AND RELIABILITY

The security aspects of the PUF secrect can be judged by its statistical characteristics or randomness. Reliability, on the other hand, can be gauged by placing the device under extreme conditions for secret regeneration and failure rate below 1 part per million (ppm) has been reported under severe fluctuation of ambient temperature and supply voltage [4].

B. MOTIVATION

Many cryptography applications such as key generation require random numbers. NIST has established several standards for cryptographically secure pseudo-random number generator (PRNG) as well as a statistical test suite for random and pseudo-random number generators [1]. If the numbers produced by a PRNG fail to pass the NIST test, it is considered vulnerable against cryptanalysis. Therefore, it is critical to verify that the secrecy generated by PUF is random and can pass the NIST test.

We consider the public available RO PUF data obtained from frequency characterization on 125 FPGA devices [2]. To our surprise, none of their random sequence can pass all the NIST tests that are applicable to their sequence length. Table 1 shows the detailed testing results. Column 1 lists the 9 statistical random tests we find applicable to the length of our test sequences.[2,3] Take Frequency Test for example, it examines whether the number of 1's and 0's in a sequence are approximately the same as would be expected for a truly random sequence, for which the number of 1's and 0's in a sequence should be about the same. If a sequence has a very disproportional 1's to 0's such that its *P-value*, the probability for events that at least as extreme as this instance to occur, is smaller than a significant level α, 1% in our case, the event is regarded significant. If it turns out that more than 4% of the total test sequences are significant, the 'PROPORTION' of a test fails; otherwise, it passes. Furthermore, the *P-values* of all the test sequences are expected to be uniformly distributed. To examine this, each of the 9 tests also calculates the *P-value* of the *P-values* using the χ^2 statistic. The 'P-VALUE (OF P-VALUES)' of a test fails if the *P-value* of χ^2 is smaller than 0.0001; otherwise, it passes.

C. CONTRIBUTION

Table 1 clearly indicates that none of the 125 PUF sequences can be deemed 'ideally random' and therefore cannot be used for critical cryptography applications. We will analyze why these PUF secrecy fails to pass the randomness tests in Section III. We argue that the systematic fabrication variation of the semiconductory process causes such failures. We study the architectures of the current RO PUFs and the aforementioned 4-step secret generation workflow. We propose to add one more step before the secret selection, which we refer to as *entropy distillation*, such that we can decouple the unwanted systematic variation from the desired stochastic variation.

There are three main contributions in this work. First, as we have pointed out in [33], this is the first work that evaluates the randomness of the random sequences generated by different PUF schemes, before being randomized by means such as hashing, against the NIST standard test. As researchers have suspected, none of them can pass all the randomness tests (results are detailed in Section V). Second, we propose to decouple the unwanted systematic variation from the desired

[2] 192 bits for the 1-out-of-8 coding. 480 bits for the chain-like neighbor coding. Choice of length may impact results.

[3] Tests like Rank Test, DFT Test, Overlapping Template Matching Test, Linear Complexity Test, Random Excursions Test and Random Excursion Variant Test require a longer random sequence over 1000 bits and thus not applicable to a 'true' random number generator like ours.

random variation through a regression-based distiller, and then generate the random sequences based on the desired random variation. Third, we describe how to design such regression-based distillers and show their effectiveness in enhancing the randomness of the random sequences. Indeed, with the help of our distillers, previously proposed PUF schemes are able to generate random sequences that can pass all the NIST tests.

We demonstrate our approach by the example of RO PUF, but the proposed method can be applied to other PUFs to enhance their security as well (of course, for those silicon PUFs that are more resistant to systematic variation, it will be less effective). As for the implementation of the proposed method, one can either implement the distiller with hardware or rely on a secure ALU for the data processing.

In the remainder of this paper, we will first introduce the basics on RO PUFs in Section II. We analyze the possible causes for the above randomness test failures in Section III. Then in Section IV, we elaborate our regression-based distiller which eliminates the systematic variation and thus fixes the randomness test failures. Finally, we report the detailed experimental results and conclude.

II. PRELIMINARIES

A. BASICS ON RO PUF

A RO PUF extracts fabrication variations through comparing the frequencies of ring oscillators that are identically designed. As depicted in Figure 2, the basic RO PUF consists of two ring oscillators, followed by two counters and one comparator at the end. When the start/stop control signal is asserted, the two ROs start to oscillate until the control signal is negated. The result of the race between the two ROs is determined by fabrication variations. During the course of the race, the two counters count the number of logic cycles run by the respective RO. At the end of race, the comparator outputs a binary result x based on the two counter values, say,

$$x = \begin{cases} 1 & \text{if Counter 1} > \text{Counter 2} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

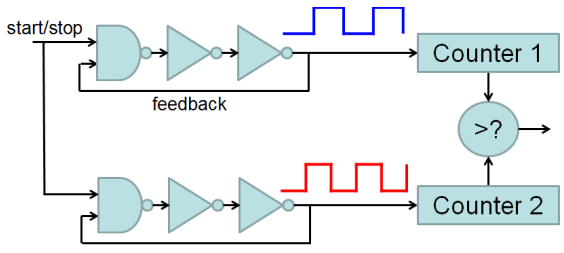

FIGURE 2. **The physical structure of a RO PUF [3].**

To generate a secret in greater length, a RO PUF typically implements hundreds of ROs arranged in a 2-dimensional array. As illustrated in Figure 3, the dataset we use implements ROs in 16 (columns) by 32 (rows) on each their 125 FPGAs [2].

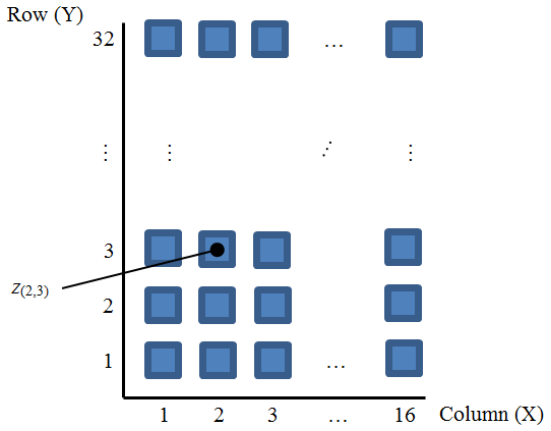

FIGURE 3. **The placement of 512 ROs as a 16 (columns) by 32 (rows) array; for site $RO_{x,y}$, its running frequency is denoted as $z_{x,y}$.**

B. 1-OUT-OF-8 CODING

The result of the race between the same two ROs may differ when the environmental conditions change. For example, a RO running faster than its peer at low temperature can actually be slower than the same peer at high temperature [3]. To prevent this, the 1-out-of-8 coding scheme uses a multiplexer to select the pair with the largest frequency difference out of 8 RO pairs as depicted in Figure 4. For the two dimensional RO array illustrated in Figure 3, we may generate one random bit for each row j from its 16 ROs $RO_{1,j} \ldots RO_{16,j}$. However, in order to generate more random bits to better serve the statistical test purpose, we made a variation by forming two blocks $(RO_{1,j} \ldots RO_{8,j})$ and $(RO_{9,j} \ldots RO_{16,j})$ for each row. The 8 ROs in the same block are referenced by a 3-bit index: 000,001,010 ... 111, and the index to the fastest RO is output as the random bits. This way we generate 6 random bits for each row.

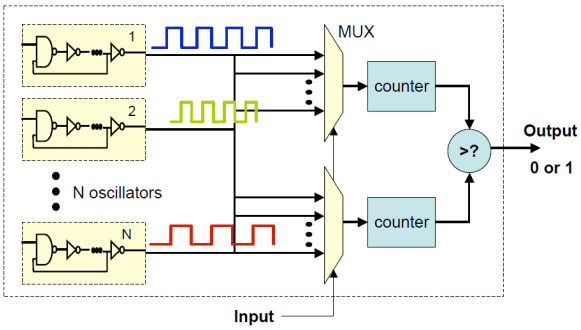

FIGURE 4. **The hardware structure of the 1-out-of-8 RO PUF [3].**

C. CHAIN-LIKE NEIGHBOR CODING

Another well-known pairing strategy is the chain-like neighbor coding, which consists of two design principles: 1) place ROs as close as possible and 2) pair ROs located adjacent to each other. In the two dimensional setting of Figure 3, we may

TABLE 1. NIST test results with respect to the random sequences generated by 1-out-of-8 coding, chain-like neighbor coding and decoupled neighbor coding.

	1-out-of-8		chain-like neighbor		decoupled neighbor	
STATISTICAL TEST	P-VALUE	PROPORTION	P-VALUE	PROPORTION	P-VALUE	PROPORTION
Frequency	0.013689	122/125	0.000072 *	125/125	0.000003 *	115/125 *
BlockFrequency	0.166594	125/125	0.000000 *	125/125	0.050764	120/125
CumulativeSums (m-2)	0.231636	121/125	0.000000 *	125/125	0.000000 *	119/125 *
CumulativeSums (m-3)	0.059743	122/125	0.000000 *	125/125	0.000000 *	118/125 *
Runs	0.002320	117/125 *	0.000000 *	0/125 *	0.302788	120/125
LongestRun	0.000603	123/125	0.000000 *	62/125 *	0.000062 *	124/125
ApproximateEntropy	0.000001 *	117/125 *	0.000000 *	0/125 *	0.000001 *	119/125 *
Serial (forward)	0.004904	124/125	0.000000 *	1/125 *	0.070160	116/125 *
Serial (backward)	0.552185	125/125	0.000000 *	117/125 *	0.192277	123/125

derive 15 random bits for each row j by pairing $(RO_{1,j}, RO_{2,j})$, $(RO_{2,j}, RO_{3,j})$, $(RO_{3,j}, RO_{4,j}) \ldots (RO_{15,j}, RO_{16,j})$.

III. SECURITY ANALYSIS

A. FAILURE CAUSE 1: CHAIN DEPENDENCY

The high failure rate of the chain-like neighbor coding can be attributed to the non-independent comparison chain. Take 3 ROs RO_A, RO_B and RO_C for example, two random bits are generated by comparing RO_A with RO_B and RO_B with RO_C. As we know, to pass NIST test for randomness, the random sequence is expected to demonstrate no significant deviation from the probability mass function (p.m.f) of tossing a fair coin twice, i.e., the 4 possible outcomes '00', '01', '10' and '11' are expected to occur equally with probability 1/4 is not the case for the two bits we generate from the 3 ROs. Let RO_i also denote the running frequency of RO_i. For three ROs, their running frequencies can have six different orders: $RO_A < RO_B < RO_C$, $RO_A < RO_C < RO_B$, $RO_B < RO_A < RO_C$, $RO_B < RO_C < RO_A$, $RO_C < RO_A < RO_B$, $RO_C < RO_B < RO_A$, where each order happens with probability 1/6 when the running frequencies are random and identical and independent distributed (i.i.d.). According to Eqn. (1), both bits x_{AB} and x_{BC} will be equally likely to be '0' or '1'. However, the 2-bit data $x_{AB}x_{BC}$ will be '00', '01', '10', and '11' with probabilities 1/6, 1/3, 1/3, and 1/6 respectively. This means that '01' or '10' occur twice as frequent as '00' or '11', clearly not the p.m.f. of the ideal random sequences.

A simple solution to fix this problem caused by the chain dependency is to break the chain such that each RO will only be paired with its neighbor once as follows: $(RO_1, RO_2), (RO_3, RO_4) \ldots (RO_{2i-1}, RO_{2i}) \ldots$. We refer to this as *decoupled neighbor coding*. Apparently this is less efficient than the original chain-like neighbor coding. For example, when there are n ROs, the chain-like neighbor coding will generate $n-1$ bits, but the decoupled neighbor coding can only generate $\frac{n}{2}$ bits. However, even with this hardware cost, we are unable to produce true random sequence. As the last two columns in Table 1 show, the decoupled neighbor coding scheme helps the original chain-like neighbor coding in passing four out of the nine 'P-VALUE (OF P-VALUES)' tests, and improves the 'PROPORTION' tests too. But it still fails half of the NIST statistical randomness tests.

B. FAILURE CAUSE 2: SPATIAL CORRELATION

Chain dependency does not exist in the 1-out-of-8 coding so to investigate the cause of the failures of the 1-out-of-8 coding, we investigate the raw data from the physical measurement of fabrication variation. Figure 5 shows how the fabrication variation of the semiconductor process portrays: the roughness of the surface (random variation) is superimposed upon a spatial trend (systematic variation). With the existence of systematic variation, the 'random' bits generated by RO PUF arrays may have very low min-entropy[4], which means that they may not be secure for cryptographic purpose. For example, as we can see in Figure 5, along the row (Y), the RO's frequency tends to increase as the Y index increases. Therefore, for the two bits x_{A1A2} and x_{B1B2} generated by two pairs of ROs (A1, A2) and (B1, B2), they are more likely to be '0' at the same time. Similarly, on a different die where the frequency tends to increase along the row (Y), these two bits are more likely to be '1' at the same time. This means that the systematic variation (the spatial correlation for two ROs on the same row (Y) in this case) will render the probability of $x_{A1A2} = x_{B1B2}$ much higher than 0.5, making them not as random nor independent.

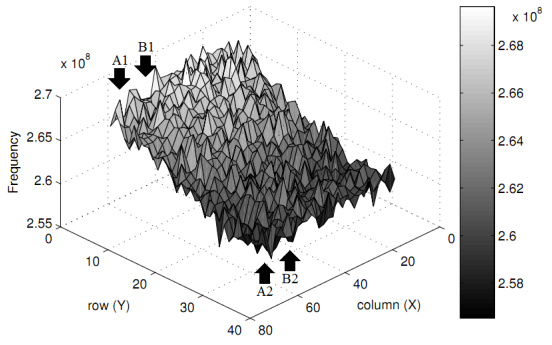

FIGURE 5. The across-die frequency topology of a RO array. The roughness of the surface represents the random variation while the slope represents the systematic variation [34].

While spatial correlation may explain the reason why none of the coding strategies in Table 1 passes all tests, it is

[4]the min-entropy of a discrete random event x with possible outcomes $1 \ldots n$ and corresponding probabilities $p_1 \ldots p_n$ is $H_\infty(X) = \min_{i=1}^{n}(-\log p_i)$.

interesting to note that in fact this threat has been reported in the chain-like neighbor coding, where they attempt to let the systematic effect cancel out with each other, extracting secrecy out of the random effect [5]. Similar principles have been used in [25] and [32]. However, the results we have in Table 1 indicate that such treatment is not sufficient to pass the NIST randomness tests. We postulate that a small remnant of the systematic variation can still be captured by the tests, causing the above failures. To illustrate this, we consider a hypothetical frequency characterization of 16 ROs as shown in Figure 6. Based on the chain-like neighbor coding, these 16 ROs will generate the following 15-bit sequence: 1101,1110,1001,000. The first bit is a '1' because $RO_1 < RO_2$ and the third bit is a '0' because $RO_3 > RO_4$, and so on. If our proposed decoupled neighbor coding is used, we will only have 8-bit data: 1011,1000. Although in both cases we have about the same number of 0's as the number of 1's, there is a clear trend that 1's are more likely to be in the first half of the sequence and the 0's in the second half. When we fit the frequencies into the curve in Figure 6, we see clearly the systematic trend of 'going up slope' first and then 'going down slope', which causes 0's and 1's not distributed uniformly in the sequences. Finally, we mention that this systematic trend can stay undiscovered when one tallies the total number of 0's and 1's or calculates the inter-die uniqueness via Hamming distance, e.g., 46.15-48.51% for RO PUFs [3], [32] and 49.97% for SRAM PUFs [35].

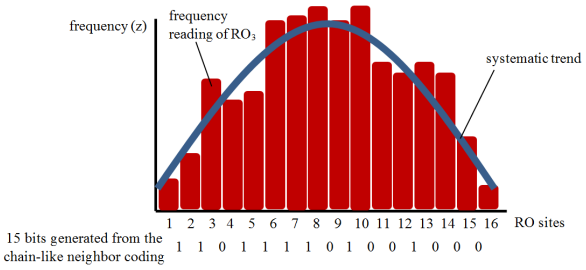

FIGURE 6. Illustration of the impact from systematic variation even after decoupling.

IV. SYSTEMATIC VARIATION ELIMINATION

We believe that one of the main causes of the failures in randomness tests for the RO PUF generated sequences is the systematic process variation. We propose to model such variation and thus remove them to build RO PUF sequences based on the true random part of the process variation. This section shows how the proposed distillation process can strengthen the randomness of the PUF output. Due to its simplicity, we apply polynomial regression to model the systematic trend. Our simulation results show that this simple model is sufficiently good as it can fix all the failures of the 1-out-of-8 coding and the decoupled neighbor coding in Table 1.

A. THE CAUSES OF PROCESS VARIATION

The semiconductor process variation has been modeled as the sum of a systematic component and a random component. The systematic component attempts to capture a deterministic trend and other identifiable patterns through one or a collection of estimators. The main causes of the systematic variation can be attributed to equipment and process non-uniformity such as the focus shift of photolithography, the gradient of thermal annealing, dissimilar interactions between circuit layout and the chemical mechanical polishing process [36], [37].

The random component, on the other hand, accounts for the difference between the model estimates and the observed data; its constituents include atomic-level stochastic phenomena such as random dopant profiles, measurement errors and any unidentified patterns [37], [38]. More discussion and related work on process variation modeling can be found in [39].

It is important to clarify that our goal is not to build a new variation model. Indeed, the proposed distiller does not require the accuracy of the variation model to be as high as those for power or performance driven applications. In the rest of this section, we will illustrate how the distiller can improve PUF data's randomness using the simple polynomial regression model.

Let us first consider an example in Figure 7. It reports the frequency information of the same 16 RO PUF as in Figure 6 except that the Y-axis now shows the difference between each RO's frequency and the systematic trend (the bell-shape curve in Figure 6). When we use the same chain-like neighbor coding, the 15-bit sequence becomes 0100,1010,1011,100. Compared to the original sequence 1101,1110,1001,000 in Figure 6, we don't see the 'predictability' that there are more 1's in the first half and more 0's in the second half of the sequence.

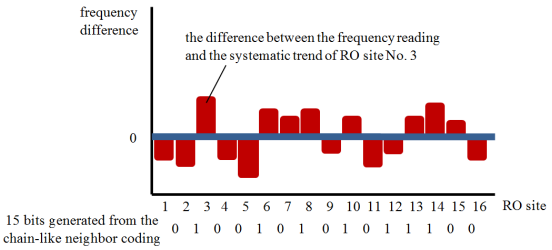

FIGURE 7. The distilled random fabrication variation after the systematic trend is removed.

B. POLYNOMIAL REGRESSION

A k^{th}-order polynomial regression is a form of linear regression in which the relationship between independent variables and a dependent variable by a polynomial of order k, where k is a non-negative integer. For a RO PUF with its m ROs arranged in r rows by c columns, the Cartesian coordinates (x, y) of ROs are regarded as two independent variables and the oscillating frequency z is the single variable dependent on x and y. In such a two dimensional setting, a polynomial regression model of order k takes the following general form

$$z_{x,y} = \sum_{i=0}^{k} \sum_{j=0}^{i} \beta_{k,i,j} {v_x}^{i-j} {h_y}^{j} + \epsilon_{k,x,y}, \qquad (2)$$

where $1 \leq x \leq c, 1 \leq y \leq r; z, \beta, \epsilon \in \mathbb{R}$. On the right hand side of the equation, the summation term models the systematic variation at the physical location (v_x, h_y) on a chip and the residual term $\epsilon_{k,x,y}$ models the random variation. In the k^{th}-order polynomial model, there will be $m = c \times r$ equations in the form of Eqn. (2) as $1 \leq x \leq c$ and $1 \leq y \leq r$. The number of unknowns $\beta_{k,i,j}$ is $n = \frac{(k+1)(k+2)}{2}$ as $0 \leq j \leq i \leq k$. This results in an overdetermined system (i.e. $m > n$), which can be solved by the ordinary least squares (OLS) method.

Equivalently, we can rewrite this in the matrix form

$$\boldsymbol{Z} = \boldsymbol{\Omega_k}\boldsymbol{\beta_k} + \boldsymbol{\epsilon_k} \tag{3}$$

where

$$\boldsymbol{\Omega_k} = \begin{pmatrix} \omega_{k,1,1} & \omega_{k,1,2} & \cdots & \omega_{k,1,n} \\ \omega_{k,2,1} & \omega_{k,2,2} & \cdots & \omega_{k,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{k,m,1} & \omega_{k,m,2} & \cdots & \omega_{k,m,n} \end{pmatrix},$$

$$\boldsymbol{Z} = \begin{pmatrix} z_{1,1} \\ \vdots \\ z_{c,1} \\ z_{1,2} \\ \vdots \\ z_{c,2} \\ \vdots \\ z_{1,r} \\ \vdots \\ z_{c,r} \end{pmatrix}, \quad \boldsymbol{\beta_k} = \begin{pmatrix} \beta_{k,0,0} \\ \beta_{k,1,0} \\ \beta_{k,1,1} \\ \beta_{k,2,0} \\ \beta_{k,2,1} \\ \beta_{k,2,2} \\ \vdots \\ \beta_{k,k,0} \\ \vdots \\ \beta_{k,k,k} \end{pmatrix}, \quad \boldsymbol{\epsilon_k} = \begin{pmatrix} \epsilon_{k,1,1} \\ \vdots \\ \epsilon_{k,c,1} \\ \epsilon_{k,1,2} \\ \vdots \\ \epsilon_{k,c,2} \\ \vdots \\ \epsilon_{k,1,r} \\ \vdots \\ \epsilon_{k,c,r} \end{pmatrix}.$$

And $\omega_{k,p,q}$'s are in the format of ${v_x}^{i-j}{h_y}^j$, where $x = ((p-1) \bmod r) + 1$, $y = \lfloor \frac{p-1}{r} \rfloor + 1$, $i = \lfloor \frac{-1+\sqrt{1+8(q-1)}}{2} \rfloor$, $j = (q-1) - \frac{i^2+i}{2}$, $1 \leq p \leq m$ and $1 \leq q \leq n$. For example, when $k = 2, c = 2, r = 3$, we have

$$\begin{pmatrix} z_{1,1} \\ z_{2,1} \\ z_{1,2} \\ z_{2,2} \\ z_{1,3} \\ z_{2,3} \end{pmatrix} = \begin{pmatrix} 1 & v_1 & h_1 & {v_1}^2 & v_1h_1 & {h_1}^2 \\ 1 & v_2 & h_1 & {v_2}^2 & v_2h_1 & {h_1}^2 \\ 1 & v_1 & h_2 & {v_1}^2 & v_1h_2 & {h_2}^2 \\ 1 & v_2 & h_2 & {v_2}^2 & v_2h_2 & {h_2}^2 \\ 1 & v_1 & h_3 & {v_1}^2 & v_1h_3 & {h_3}^2 \\ 1 & v_2 & h_3 & {v_2}^2 & v_2h_3 & {h_3}^2 \end{pmatrix} \times \begin{pmatrix} \beta_{2,0,0} \\ \beta_{2,1,0} \\ \beta_{2,1,1} \\ \beta_{2,2,0} \\ \beta_{2,2,1} \\ \beta_{2,2,2} \end{pmatrix} + \begin{pmatrix} \epsilon_{2,1,1} \\ \epsilon_{2,2,1} \\ \epsilon_{2,1,2} \\ \epsilon_{2,2,2} \\ \epsilon_{2,1,3} \\ \epsilon_{2,2,3} \end{pmatrix}.$$

The OLS method will find the 'best' estimate $\hat{\boldsymbol{\beta}}$ in terms of the minimum sum of squared errors as Eqn. (4) indicates. By taking partial directives of Eqn. (5) with respect to each $\beta_{k,i,j}$ and letting each gradient to zero, the solution of OLS can be expressed in the matrix form as in Eqn. (6).

$$\hat{\boldsymbol{\beta}}_k = \arg\min_{\boldsymbol{\beta_k}} \left\{ \sum_{x=1,y=1}^{c,r} \epsilon_{k,x,y}^2 \right\} \tag{4}$$

$$= \arg\min_{\boldsymbol{\beta_k}} \left\{ \sum_{x=1,y=1}^{c,r} (z_{x,y} - \sum_{i=0}^{k}\sum_{j=0}^{i} \beta_{k,i,j} x^{i-j} y^j)^2 \right\} \tag{5}$$

$$= (\boldsymbol{\Omega_k^T}\boldsymbol{\Omega_k})^{-1}\boldsymbol{\Omega_k^T}\boldsymbol{Z} \tag{6}$$

C. REGRESSION-BASED DISTILLER

When we apply polynomial regression models to capture the systematic variation trend, higher order models have better accuracy and generate smaller residual terms ($\epsilon_{k,x,y}$'s in Eqn. (2)). While they may lead to sequences that are more random and secure, they incur more computational cost and more importantly, the small magnitude of the residual terms can cause difficulties in error correction phase and damage the efficiency of RO PUF. For example, Figure 8 shows the histograms of the random variation of a data set (see the result section for detailed description of the data) after regression models with polynomials of degrees 0 to 6 are applied. Clearly we see as the order increases, the number of ROs whose frequencies are far from the center decreases quickly, yielding a smaller variance. Nevertheless, they all appear normal distribution and it is difficult to judge which model is the best choice without running the standard randomness tests. Therefore, our goal is to find the polynomial regression model in minimal order that can successfully distill the ideal random variation. We propose to conduct this distillation procedure after the 'fabrication variation extraction' phase. The remaining question is in the next 'secrecy selection' phase that how to build the RO PUF sequence based on the residual terms, or the true random variations.

Suppose we have a 2-dimensional array of ROs placed in r rows and c columns (see Figure 3), there are many ways to define RO PUF bits from the distilled RO frequency information. For example, in our implementation of the 1-out-of-8 coding, each row generates $\frac{c}{8} \times 3$ bits; in the chain-like neighbor coding, we have $c - 1$ bits from each row; in the decoupled neighbor coding, this number reduces to $\lfloor \frac{c}{2} \rfloor$. Of course, instead of focusing on each row, we can define RO PUF bits by comparing the ROs in the same column. In addition to these three coding schemes, we study the following two generic sequences, S and T, to gauge if there is still any trace of spatial correlation in the distilled random component:

$$S = X_1, \ldots, X_{l_X}, \ldots, X_{L_X} \text{ where } X_{l_X} = \begin{cases} 0 & \text{if } z_{u_X,v_X} \leq z_{u_X+\lfloor \frac{c}{2} \rfloor, v_X} \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

$$T = Y_1, \ldots, Y_{l_Y}, \ldots, Y_{L_Y} \text{ where } Y_{l_Y} = \begin{cases} 0 & \text{if } z_{u_Y,v_Y} \leq z_{u_Y,v_Y+\lfloor \frac{r}{2} \rfloor} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

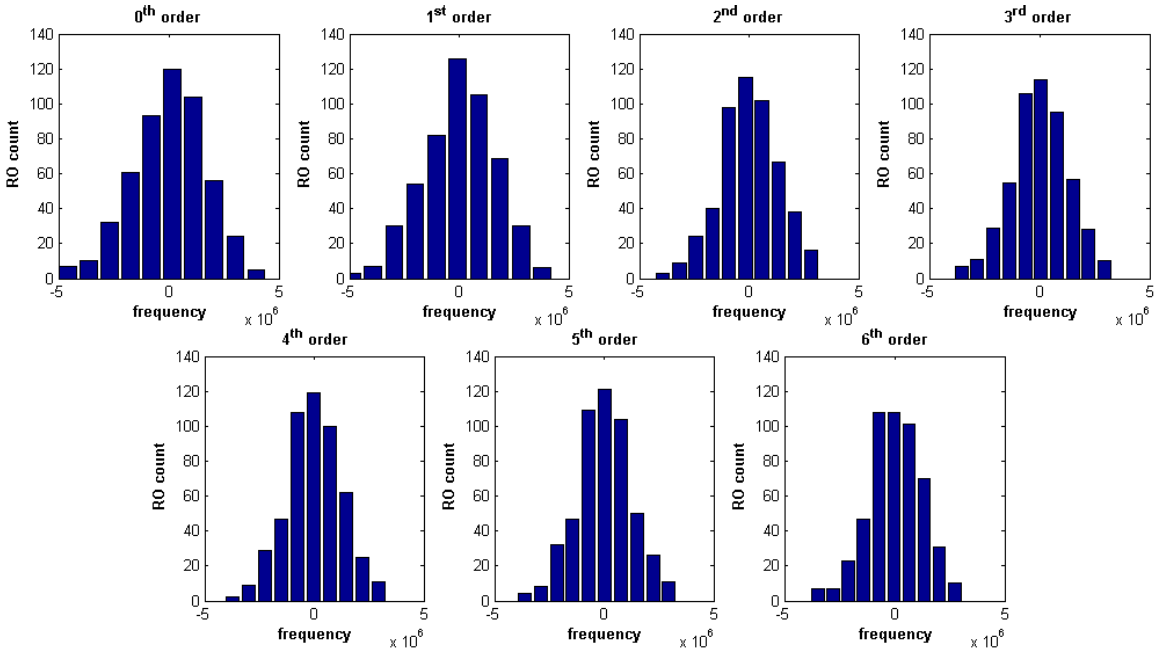

FIGURE 8. The histogram of the distilled random variation after applying 0^{th} through 6^{th}-order polynomial regression to the dataset of Chip No. 1.

where in (7), $u_X = ((l_X - 1) \bmod \lfloor \frac{c}{2} \rfloor) + 1$, $v_X = \lfloor (l_X - 1)/\lfloor \frac{c}{2} \rfloor \rfloor + 1$, $1 \le l_X \le L_X = r \times \lfloor \frac{c}{2} \rfloor$; similarly in (8), $u_Y = \lfloor (l_Y - 1)/\lfloor \frac{r}{2} \rfloor \rfloor + 1$, $v_Y = ((l_Y - 1) \bmod \lfloor \frac{r}{2} \rfloor) + 1$, $1 \le l_Y \le L_Y = c \times \lfloor \frac{r}{2} \rfloor$.

Intuitively, S and T are formulated by cutting each row (or column in T) in the RO array into two equal halves, pairing up ROs in the two halves, and comparing their residual variation terms. Recall that the principle in neighbor coding is to pair up ROs that are next to each other in order to reduce the systematic variation. In S and T, we have purposely done the opposite to pair up ROs that are far from each other to amplify the effect of systematic variation in order to test the effectiveness of the proposed regression-based entropy distiller. In the next section, we will report our detailed findings on such randomness tests.

V. RESULTS AND ANALYSIS ON RANDOMNESS TESTS

In this section, we conduct standard NIST randomness tests to validate that the proposed regression-based entropy distiller will improve the randomness of the RO PUF sequences. We use the test bench in the public domain which consists of the frequency characterization of 125 RO PUFs implemented on 125 Xilinx Spartan-3 FPGAs, where 512 ROs were placed on each FPGA as shown Figure 3 [2].

A. THE POLYNOMIAL REGRESSION MODELS

We first report the systematic variation distillation procedure and then results. For each chip, we apply regression models of different orders to its 512 averages of frequency readings. Figure 9 shows the modeled systematic variation for each RO on the first chip. In the 0^{th} order, the systematic variation is the average of the 512 averages. In the 1^{st} order linear model, we see that the ROs have higher frequency as their Y coordinates decrease. As we use higher order polynomials, it starts to show trend similar to Figure 5.

Figure 10 shows the random variation after distillation. We see the radial pattern close to the center for the 0^{th} and 1^{st} models, which is known as the 'bull's eye' and a clear indication of 'non-randomness'. However, it vanishes in the cases of 2^{nd} model and beyond. This suggests us that polynomials of 2^{nd} degree or higher should be used.

B. NIST RANDOMNESS TESTS

There are nine randomness tests in the NIST statistical test suite applicable to the length of our test sequences: Frequency Test, Block Frequency Test, Cumulative Sums Test (with block size $m = 2$ and $m = 3$), Runs Test, Longest Run Test, Serial Test (both forward and backward) and Approximate Entropy Test. According to [1], empirical results have to be interpreted in two forms of analysis: First, the proportion of sequences passing a test shall be above a minimum rate, 0.96 in our case, i.e., to pass 120 sequences out of a sample size of 125 sequences at significance level $\alpha = 0.01$. Secondly, the *P-values* of all the random sequences shall be uniformly distributed. Based on χ^2 Goodness-of-Fit Test, the underlying distribution is deemed uniform if the *P-value* of the *P-values* is equal or greater than 0.0001 for a population of 125 sequences. Whenever either of these two approaches fails, further tests based on a different sample space will help clarify whether the failure is a statistical anomaly or a clear non-randomness.

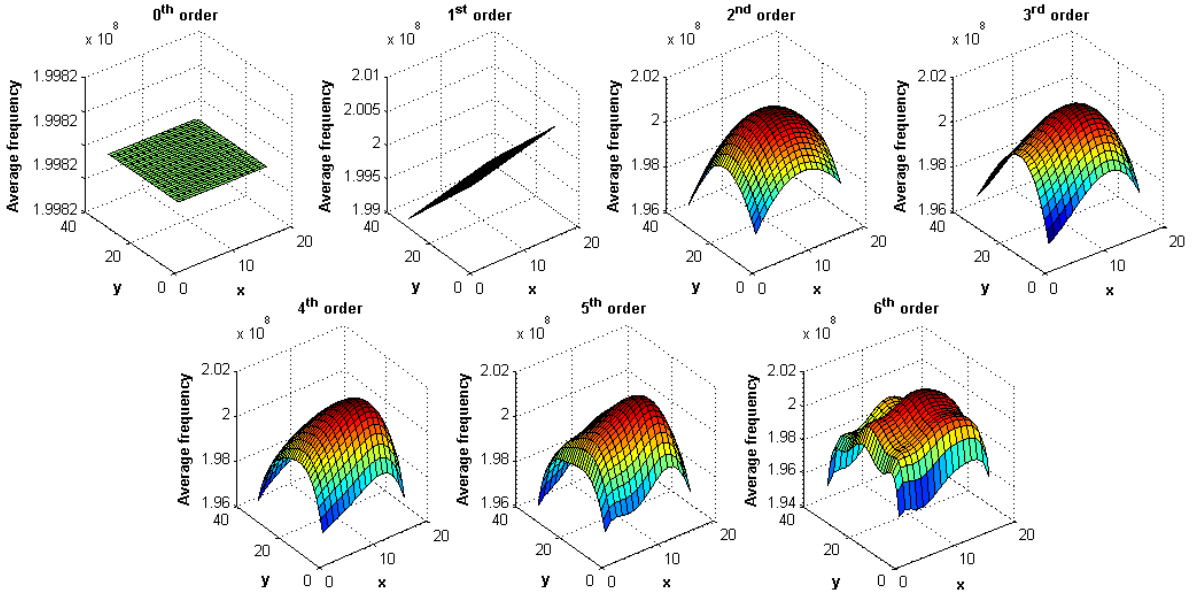

FIGURE 9. The modeled systematic variation after applying 0^{th} through 6^{th}-order polynomial regression to the dataset of Chip No. 1.

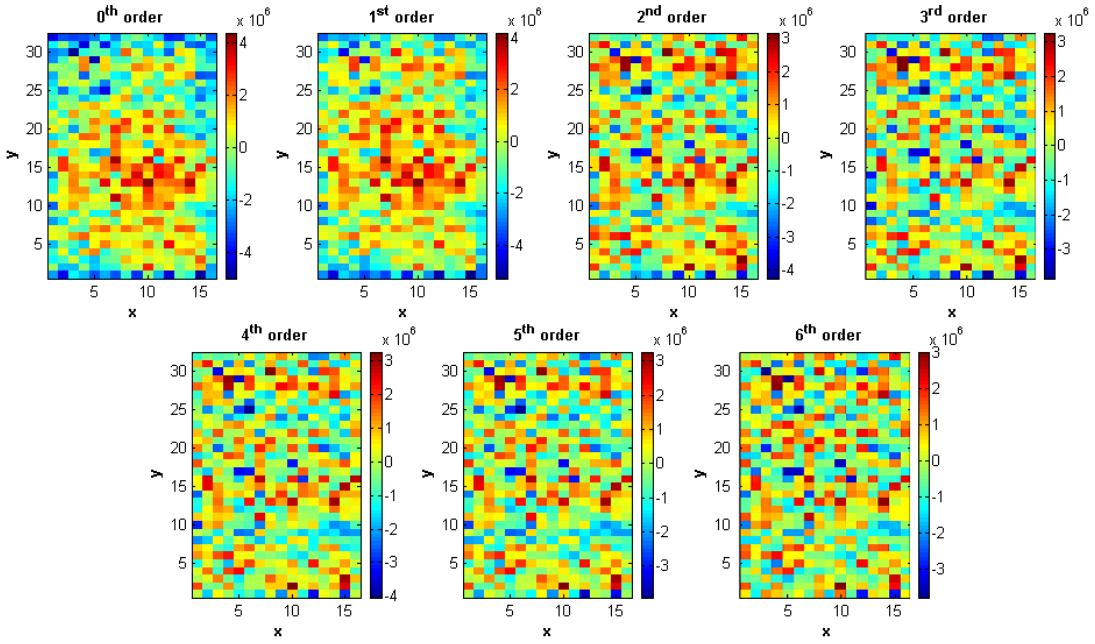

FIGURE 10. The distilled random variation after applying 0^{th} through 6^{th}-order polynomial regression to the dataset of Chip No. 1. Notably, we see the 'bull's eye', i.e., the radial pattern close to the center, vanishing in the cases of 2^{nd} order model and beyond.

Table 2 reports the detailed test results on the generic S-sequence, T-sequence, and the sequences generated by the coding schemes of 1-out-of-8, chain-like neighbor, and decoupled neighbor.

1) S-SEQUENCE AND T-SEQUENCE

The 512 ROs will generate a 256-bit S-sequence and a 256-bit T-sequence. The S-sequence and T-sequence for NIST randomness test are 32000 bits long obtained by concatenating the 125 such 256-bit sequence from the 125 chips.

As the 0^{th}-order section shows, random sequences generated without entropy distillation fail miserably for both forms of analysis 'PROP. (PROPORTION)' and 'P-VAL. (P-VALUE OF P-VALUES)', where '*' marks a failure. This strongly suggests the existence of systematic variation in the raw data. The failure rate decreases sharply when applied

TABLE 2. The results of NIST 'P-VAL. (P-VALUE of P-VALUES)' and 'PROP. (PROPORTION)' analyses with respect to random sequences generated by *S*, *T*, the 1-out-of-8 coding, the chain-like neighbor coding and the decoupled neighbor coding accompanied by 0^{th}- to 6^{th}-order distillers, where '*' marks a failure.

	S		T		**1-out-of-8**		**chain-like neighbor**		**decoupled neighbor**		
	P-VAL.	PROP.	P-VAL.	PROP.	P-VAL.	PROP.	P-VAL.	PROP.	P-VAL.	PROP.	STATISTICAL TEST
0^{th}-order	0.000000 *	45 *	0.000000 *	38 *	0.013689	122	0.000072 *	125	0.000003 *	115 *	Frequency
	0.000000 *	59 *	0.000000 *	49 *	0.166594	125	0.000000 *	125	0.050764	120	BlockFrequency
	0.000000 *	46 *	0.000000 *	39 *	0.231636	121	0.000000 *	125	0.000000 *	119 *	CumulativeSums (m-2)
	0.000000 *	46 *	0.000000 *	38 *	0.059743	122	0.000000 *	125	0.000000 *	118 *	CumulativeSums (m-3)
	0.000000 *	65 *	0.000000 *	31 *	0.002320	117 *	0.000000 *	0 *	0.302788	120	Runs
	0.000000 *	66 *	0.000000 *	44 *	0.000603	123	0.000000 *	62 *	0.000062 *	124	LongestRun
	0.000000 *	53 *	0.000000 *	23 *	0.000001 *	117 *	0.000000 *	0 *	0.000001 *	119 *	ApproximateEntropy
	0.000000 *	65 *	0.000000 *	25 *	0.004904	124	0.000000 *	1 *	0.070160	116 *	Serial (forward)
	0.000000 *	103 *	0.000000 *	74 *	0.552185	125	0.000000 *	117 *	0.192277	123	Serial (backward)
1^{st}-order	0.166594	124	0.003598	122	0.000949	120	0.000072 *	125	0.130323	124	Frequency
	0.000002 *	120	0.889414	121	0.529142	125	0.000000 *	125	0.056599	122	BlockFrequency
	0.000100	120	0.136969	122	0.063046	120	0.000000 *	125	0.082208	124	CumulativeSums (m-2)
	0.405918	122	0.020616	119 *	0.043046	120	0.000000 *	125	0.034444	123	CumulativeSums (m-3)
	0.082208	124	0.000000 *	68 *	0.192277	124	0.000000 *	0 *	0.096097	122	Runs
	0.048059	120	0.000000 *	90 *	0.000067 *	123	0.000000 *	62 *	0.000274	124	LongestRun
	0.025948	120	0.000000 *	75 *	0.002471	120	0.000000 *	0 *	0.130323	122	ApproximateEntropy
	0.112055	122	0.000000 *	80 *	0.262219	124	0.000000 *	1 *	0.956806	122	Serial (forward)
	0.474938	121	0.000000 *	117 *	0.551044	125	0.000000 *	117 *	0.620686	123	Serial (backward)
2^{nd}-order	0.369588	122	0.012159	121	0.000000 *	115 *	0.001228	125	0.086622	121	Frequency
	0.000782	122	0.422488	122	0.059743	124	0.000000 *	125	0.262219	123	BlockFrequency
	0.020616	120	0.086622	120	0.000000 *	118 *	0.000000 *	125	0.073984	123	CumulativeSums (m-2)
	0.575157	122	0.066516	122	0.000000 *	116 *	0.000000 *	125	0.389809	120	CumulativeSums (m-3)
	0.316158	125	0.915772	122	0.552185	123	0.000000 *	0 *	0.493319	124	Runs
	0.000062 *	122	0.000782	120	0.000000 *	123	0.000000 *	58 *	0.000115	124	LongestRun
	0.750075	124	0.474938	120	0.000000 *	120	0.000000 *	0 *	0.316158	122	ApproximateEntropy
	0.874833	124	0.077998	122	0.000123	123	0.000000 *	0 *	0.643139	121	Serial (forward)
	0.231636	123	0.302788	125	0.457002	124	0.000000 *	110 *	0.262219	123	Serial (backward)
3^{rd}-order	0.011457	125	0.136969	124	0.000000 *	111 *	0.000000 *	125	0.389809	123	Frequency
	0.262219	124	0.551044	125	0.003829	123	0.000000 *	125	0.457002	122	BlockFrequency
	0.002320	125	0.000131	125	0.000000 *	112 *	0.000000 *	125	0.551044	124	CumulativeSums (m-2)
	0.002984	125	0.017315	125	0.000000 *	111 *	0.000000 *	125	0.192277	123	CumulativeSums (m-3)
	0.643139	124	0.529142	121	0.889414	124	0.000000 *	0 *	0.529142	124	Runs
	0.000058 *	123	0.012903	124	0.000000 *	120	0.000000 *	48 *	0.000000 *	123	LongestRun
	0.020616	125	0.422488	123	0.000000 *	114 *	0.000000 *	0 *	0.889414	125	ApproximateEntropy
	0.369588	124	0.915772	123	0.000017 *	121	0.000000 *	0 *	0.493319	123	Serial (forward)
	0.439517	124	0.506075	122	0.575157	124	0.000000 *	114 *	0.439517	125	Serial (backward)
4^{th}-order	0.000001 *	125	0.000000 *	125	0.000407	121	0.000000 *	125	0.192277	124	Frequency
	0.166594	125	0.000051 *	125	0.344248	123	0.000000 *	125	0.807956	124	BlockFrequency
	0.000000 *	125	0.000000 *	125	0.143910	121	0.000000 *	125	0.070160	124	CumulativeSums (m-2)
	0.000011 *	125	0.000000 *	124	0.130323	120	0.000000 *	125	0.117876	124	CumulativeSums (m-3)
	0.316158	125	0.903069	122	0.437182	125	0.000000 *	0 *	0.414457	125	Runs
	0.004904	123	0.045489	125	0.007522	120	0.000000 *	54 *	0.000000 *	123	LongestRun
	0.289860	125	0.265309	122	0.708591	122	0.000000 *	0 *	0.143910	122	ApproximateEntropy
	0.571108	125	0.665311	123	0.571108	125	0.000000 *	1 *	0.457002	123	Serial (forward)
	0.405918	123	0.283039	124	0.551044	125	0.000000 *	110 *	0.825875	124	Serial (backward)
5^{th}-order	0.000029 *	125	0.211194	125	0.316158	124	0.000000 *	125	0.096097	125	Frequency
	0.004074	125	0.000000 *	125	0.493319	124	0.000000 *	125	0.552185	124	BlockFrequency
	0.000000 *	125	0.000000 *	125	0.665311	124	0.000000 *	125	0.043046	124	CumulativeSums (m-2)
	0.000000 *	125	0.000000 *	125	0.166594	123	0.000000 *	125	0.036430	125	CumulativeSums (m-3)
	0.493319	124	0.687147	124	0.036430	125	0.000000 *	0 *	0.192277	123	Runs
	0.006661	124	0.001801	125	0.036430	124	0.000000 *	42 *	0.000000 *	124	LongestRun
	0.059743	125	0.302788	124	0.729586	125	0.000000 *	0 *	0.665311	122	ApproximateEntropy
	0.687147	125	0.304210	121	0.096097	125	0.000000 *	0 *	0.512137	124	Serial (forward)
	0.262219	123	0.789315	123	0.457002	125	0.000000 *	114 *	0.474938	125	Serial (backward)
6^{th}-order	0.000009 *	125	0.001586	125	0.001080	125	0.000000 *	125	0.231636	122	Frequency
	0.000000 *	125	0.000000 *	125	0.086622	125	0.000000 *	125	0.437182	123	BlockFrequency
	0.000000 *	125	0.000000 *	125	0.231636	125	0.000000 *	125	0.091249	123	CumulativeSums (m-2)
	0.000000 *	125	0.000000 *	125	0.050764	124	0.000000 *	125	0.211194	123	CumulativeSums (m-3)
	0.101175	125	0.130323	123	0.422488	124	0.000000 *	0 *	0.529142	122	Runs
	0.000643	124	0.007992	124	0.211194	125	0.000000 *	44 *	0.000017 *	122	LongestRun
	0.000006 *	125	0.643139	124	0.130323	124	0.000000 *	0 *	0.598008	124	ApproximateEntropy
	0.552185	124	0.289860	123	0.329976	124	0.000000 *	0 *	0.277369	123	Serial (forward)
	0.874833	124	0.529142	124	0.405918	124	0.000000 *	113 *	0.843024	121	Serial (backward)

with 1^{st}-, 2^{nd}- or 3^{rd}-order distiller in the case of *S* and with 2^{nd}- or 3^{rd}-order distiller in the case of *T*.

Unfortunately, there is at least one failure with respect to *S*, though the failure is only slightly below the cutting value. In such a boarder case where a weak existence of systematic variation is inferred, further investigation with different dataset is necessary to conclude the entropy source, i.e., RO PUF plus the distillation model, 'good' or 'bad'. If simply taking the sum of failure rates with respect to *S* and *T*, either 2^{nd}- or 3^{rd}-order distillers can be considered optimal.

Finally, we mention that the pass rate of the 'P-VALUE OF P-VALUES' analysis drops when applied with a model

of 4^{th} order or higher. A further investigation reveals that this is caused by model over-fitting. When we use high order models, there will be more coefficients $\beta_{k,i,j}$ (as in Eqn. (2)) to describe the underlying systematic variation. In general, this will give us better model. However, considering the limited number of data samples we have (a 256-bit S-sequence or a 256-bit T-sequence), if we use a 6^{th} order polynomial model where 28 unknown $\beta_{k,i,j}$'s need to determined, the model will capture the random variation instead of the systematic variation and causes over-fitting. When that happens, there is little 'true' random variation left and thus randomness test will fail. We can see this from Table 2. This result also indicates that the 2^{nd} or 3^{rd} order model should suffice with the number of data samples we have.

2) 1-OUT-OF-8 CODING

For our implementation of the 1-out-of-8 coding, a 3-bit index '000', '001'... or '111' is generated by pointing to the fastest RO out of 8 consecutive ROs on the same row, i.e., 192 bits per chip or $125 \times 192 = 24000$ bits for the test sequence.

From Table 2, we see that the 1-out-of-8 coding does very well even without the entropy distillation with only one clear 'P-VALUE' failure and two marginal 'PROPORTION' failures. The best linear model can fix all these three failures, but introduced a different 'P-VALUE' failure which is marginal. Also, distillers of 4^{th} order and higher pass all the tests and can be deemed 'good'. However, the 2^{nd} and 3^{rd} order models fail about half of the tests. We suspect that this is caused by the fact that we are collecting three bits at a time from the 3-bit index of the eight ROs. Models of low order may not be able to captuer certain intrinsic correlation behind such selection.

3) NEIGHBOR CODING

In the case of the chain-like neighbor coding, 15 bits are generated per row by pairing up with row neighbors, which yields 480 bits per chip. Thus, the length of the test sequence is $125 \times 480 = 60000$ bits.

As shown in Table 2, none of the polynomial distiller makes meaningful improvement. This phenomenon aligns with our expectation that the failures are caused by the intrinsic chain dependencies of the pairing strategy rather than spatial correlation. Moreover, consider our treatment to this problem, the decoupled neighbor coding, the 1^{st} order linear model is capable of helping it to pass all the randomness tests.

The over-fitting problem for high order polynomial models does not seem to be a concern in this case. The only exception is the 'LongestRun' test which is also the test that the 2^{nd} and 3^{rd} order models fail in the case of S-sequence. Considering the severe over-fitting problem in the case of S-sequence and T-sequence, we believe this is due to the structural difference between the decoupled neighbor coding and the S- or T-sequence. As we have discussed when defining the S-sequence and T-sequence, they are designed to amplify the systematic variation, so over-fitting is more likely to occur. In decoupled neighbor coding, we pair two ROs that are physically close to each other, hence they will have similar systematic variation that can be easily and accurately captured by the distillers. More detailed results can be found in [39].

VI. CONCLUSION

The systematic component of fabrication variation has long posted a security threat to RO PUFs. This work provides experimental data to demonstrate that none of the current coding schemes can pass all the NIST randomness tests. To address the issue, we propose a family of entropy distillers based on polynomial regression. We affirm their effectiveness in improving the randomness of the PUF output.

REFERENCES

[1] A. Rukhin *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST, Gaithersburg, MD, USA, Special Publication 800–22, Revision 1a, Apr. 2010.

[2] A. Maiti and P. Schaumont, "A large scale characterization of RO-PUF," in *Proc. 3rd IEEE Int. Workshop HOST*, Jun. 2010, pp. 94–99.

[3] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE DAC*, Jun. 2007, pp. 9–14.

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

[5] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," in *Proc. 19th IEEE Int. Conf. FPLA*, Sep. 2009, pp. 703–707.

[6] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. New York, NY, USA: Wiley, 2001.

[7] R. Anderson and M. Kuhn, "Tamper resistance—A cautionary note," in *Proc. 2nd USENIX Workshop Electron. Commerce*, Nov. 1996, pp. 1–11.

[8] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices," in *Security Protocols* (Lecture Notes in Computer Science), vol. 1361. Berlin, Germany: Springer-Verlag, 1997, pp. 125–136.

[9] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 1666. Berlin, Germany: Springer-Verlag, 1999, pp. 388–397.

[10] S. Korobogatov and R. Anderson, "Optical fault induction attacks," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 2523. Berlin, Germany: Springer-Verlag, 2002, pp. 2–12.

[11] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving smart card security using self-timed circuits," in *Proc. 8th Int. Symp. IEEE Asynchron. Circuits Syst.*, Apr. 2002, pp. 211–218.

[12] S. Weingart, "Physical security devices for computer subsystems: A survey of attacks and defenses," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 1965. Berlin, Germany: Springer-Verlag, 2000, pp. 302–317.

[13] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 4249. Berlin, Germany: Springer-Verlag, 2006.

[14] P. Tuyls and B. Škorić, "Secret key generation from classical physics," *Hardware Technology Drivers of Ambient Intelligence* (Philips Research Book). Norwell, MA, USA: Kluwer, 2005, pp. 421–447.

[15] R. Pappu, "Physical one-way functions," Ph.D. dissertation, School Archit. Planning, MIT, Cambridge, MA, USA, Mar. 2001.

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

[17] D. Lim, J.-W. Lee, B. Gassend, M. van Dijk, E. Suh, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[18] D. Holcomb, W. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proc. Conf. RFID Security*, Jul. 2007.

[19] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 4727. Berlin, Germany: Springer-Verlag, 2007.

[20] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on FPGAs," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 5154. Berlin, Germany: Springer-Verlag, 2008, pp. 181–197.
[21] R. Maes, P. Tuyls, and I. Verbauwhede, "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs," in *Cryptographic Hardware and Embedded Systems* (Lecture Notes in Computer Science), vol. 5747. Berlin, Germany: Springer-Verlag, 2009, pp. 332–347.
[22] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *Proc. 18th Annu. Comput. Security Appl. Conf.*, Dec. 2002, pp. 149–160.
[23] P. Tuyls, B. Skoric, and T. Kevenaar, *Security With Noisy Data: On Private Biometrics, Secure Key Storage and Anti-Counterfeiting*. New York, NY, USA: Springer-Verlag, 2007.
[24] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. IEEE/ACM ICCAD*, Nov. 2008.
[25] X. Wang and M. Tehranipoor, "Novel physical unclonable function with process and environmental variations," in *Proc. DATE*, Mar. 2010, pp. 1065–1070.
[26] M. Yu and S. Devadas, "Recombination of physical unclonable functions," in *Proc. 35th Annu. GOMACTech Conf.*, Mar. 2010.
[27] U. Ruhrmair, F. Sehnke, J. Soelter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM CCS Conf.*, Oct. 2010, pp. 237–249.
[28] A. Sadeghi and D. Naccache, "Towards hardware-intrinsic security," in *Information Security and Cryptography*. Berlin, Germany: Springer-Verlag, 2010.
[29] C.-E. Yin and G. Qu, "Temperature-aware cooperative ring oscillator PUF," in *Proc. 2nd IEEE Int. Workshop HOST*, Jul. 2009, pp. 36–42.
[30] C.-E. D. Yin and G. Qu, "LISA: Maximizing RO PUF's secret extraction," in *Proc. 3rd IEEE Int. Workshop HOST*, Jun. 2010, pp. 100–105.
[31] C.-E. Yin, G. Qu, and Q. Zhou, "Design and implementation of a group-based RO PUF," in *Proc. DATE*, Mar. 2013, pp. 416–421.
[32] D. Merli, F. Stumpf, and C. Eckert, "Improving the quality of ring oscillator PUFs on FPGAs," in *Proc. 5th Workshop Embedded Syst. Security*, Oct. 2010.
[33] C.-E. Yin, G. Qu, and Q. Zhou, "Improving PUF security with regression-based distiller," in *Proc. 50th ACM/IEEE DAC*, Jun. 2013, pp. 1–6.
[34] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90 nm FPGAs and beyond," in *Proc. 16th IEEE Int. Conf. FPT*, Dec. 2006, pp. 97–104.
[35] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls, "Physical unclonable functions and public-key crypto for FPGA IP protection," in *Proc. 17th IEEE Int. Conf. FPL Appl.*, Aug. 2007, pp. 189–195.
[36] B. E. Stine, D. S. Boning, and J. E. Chung, "Analysis and decomposition of spatial variation in integrated circuit processes and devices," *IEEE Trans. Semicond. Manuf.*, vol. 10, no. 1, pp. 24–91, Feb. 1997.
[37] K. Bernstein *et al.*, "High-performance CMOS variability in the 65-nm regime and beyond," *IBM J. Res. Develop.*, vol. 50, nos. 4–5, pp. 433–449, Jul. 2006.
[38] E. Chang *et al.*, "Using a statistical metrology framework to identify systematic and random sources of die- and wafer-level ILD thickness variation in CMP processes," in *Proc. IEDM*, Dec. 1995, pp. 499–502.
[39] C.-E. Yin and G. Qu. (2014, Mar.). Obtaining Statistically Random Information from Silicon Physical Unclonable Functions. Inst. Syst. Res., Univ. Maryland, College Park, MD, USA [Online]. Available: drum.lib.umd.edu/handle/1903/15000

CHI-EN DANIEL Yin received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1999 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Maryland at College Park, College Park, MD, USA, in 2012. He is currently with eCurrency Mint, Oakland, CA, USA, a virtual currency startup in the Bay Area of California. His research interests include physical unclonable function, smart card, Linux kernel, cloud security, and bitcoin.

GANG QU (S'98–M'01–SM'08) received the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 2000. He is currently a Professor with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland at College Park, College Park, MD, USA. He is also a member of the Maryland Cybersecurity Center and the Maryland Energy Research Center, College Park, MD, USA. He is the Director of the Maryland Embedded Systems and Hardware Security Laboratory and the Wireless Sensors Laboratory. His primary research interests are in the area of embedded systems and VLSI CAD with focus on low-power system design and hardware-related security and trust. He studies optimization and combinatorial problems and applies his theoretical discovery to applications in VLSI CAD, wireless sensor network, bioinformatics, and cybersecurity. He has received many awards for his academic achievements, teaching, and service to the research community. He is serving as the Associate Editor of the IEEE TRANSACTOINS ON COMPUTERS, the IEEE EMBEDDED SYSTEMS LETTERS and INTEGRATION, and the *VLSI Journal*.