Secure and Robust Error Correction for Physical Unclonable Functions

Meng-Day (Mandel) Yu
Verayo

Srinivas Devadas
Massachusetts Institute of Technology

Editor's note:
Physical unclonable functions (PUFs) offer a promising mechanism that can be used in many security, protection, and digital rights management applications. One key issue is the stability of PUF responses that is often addressed by error correction codes. The authors propose a new syndrome coding scheme that limits the amount of leaked information by the PUF error-correcting codes.
—Farinaz Koushanfar, Rice University

An important aspect of improving the trustworthiness level of semiconductor devices, semiconductor-based systems, and the semiconductor supply chain is enhancing physical security. We want semiconductor devices to be resistant not only to computational attacks but also to physical attacks. Gassend et al. described the use of silicon-based physical random functions,[1,2] also called *physical unclonable functions* (PUFs), to generate signatures based on device manufacturing variations that are difficult to control or reproduce. Given a fixed challenge as input, a PUF outputs a response that is unique to the manufacturing instance of the PUF circuit. These responses are similar, but not necessarily bit exact, when regenerated on a given device using a given challenge, and are expected to deviate more in Hamming distance from a reference response to the extent that environmental parameters (e.g., temperature and voltage) vary between provisioning and regeneration. This deviation occurs because circuit delays do not vary uniformly with temperature and voltage.

PUFs have two broad classes of applications.[1,3-5] In certain classes of authentication applications, the silicon device is authenticated if the regenerated response is close enough in Hamming distance to the provisioned response. To prevent replay attacks, challenges are never repeated. This means the PUF must be resistant to software model-building attacks (e.g., learning attacks like those Lim described[6]) to be secure. Otherwise, an adversary can create a software model or clone of a particular PUF.

If, instead of Hamming-based authentication as we've described, the PUF is to serve as a secret-key generator, only a fixed number of secret bits need to be generated from the PUF. These bits can serve as symmetric key bits or as a random seed to generate a public-private key pair in a secure processor.[3] However, in order for the PUF outputs to be usable in cryptographic applications, the noisy bits must be error corrected, with the aid of helper bits; these helper bits are commonly referred to as a *syndrome*. The greater the environmental variation a PUF is subject to, the greater the possible difference (noise) between a provisioned PUF response and a regenerated response.

Software model-building attacks are not a concern when a fixed number of independent secret bits are generated from the PUF. These bits, if noise-free, need not be exposed (for example, these bits may be one-way hashed prior to being exposed; in this case, model building of the PUF requires inverting the one-way hash), and therefore an adversary cannot construct a model of the PUF.

In perhaps the earliest reference to error correction in silicon PUFs, Gassend cited the use of 2D Hamming codes for error correction.[1] (For more information on PUFs and error correction, see the "Related Work" sidebar.) Suh et al. had a more realistic view of noisy properties of PUFs and suggested the use of Bose-Chaudhuri-Hochquenghen (BCH) code, specifically BCH (255, 63, $t = 30$) code for error correction,[3] where the PUF generates 255 bits, but because 192 syndrome bits are exposed in public storage, the actual key size is no more than 63 bits. This code can be used to correct 30 errors out of 255 bits but is expensive to implement. Maximum error rates for PUFs across environmental variations reflecting variations from real-life deployments can be as high as 25%, making straightforward use of BCH impractical—the codeword sizes required would be too large for practical realizations. For these high error rates, error reduction techniques must be applied prior to error correction. For example, PUF bits that are less likely to be noisy can be selected, and/or repetition coding can be used. Error reduction requires additional helper or syndrome bits to be publicly stored. These bits could leak information. For example, by using these syndrome bits, the adversary might obtain bias information that can be used to reduce the search space required to obtain the secret key. Information leakage via syndrome coding has not received much attention in practical PUF-based key generation systems.

Accordingly, in this article, which focuses on the use of a PUF and error correction techniques to generate cryptographic keys, we propose a new syndrome coding scheme called *index-based syndrome* coding. IBS differs from conventional syndrome coding methods, such as the code-offset construction using linear codes,[7] in two main respects. First, by its very nature it leaks less information than conventional methods or other variants that use bitwise XOR masking. The key idea is to generate pointers to values in a PUF output sequence so that the syndrome bits no longer need to be a direct linear mathematical function of PUF output bits and parity bits. Under the assumption that PUF outputs are independent and identically distributed (IID), IBS can be shown to be what is known as *information-theoretically secure* (i.e., security can be derived entirely from information theory) from the standpoint that IBS does not contribute to additional min-entropy loss. In applying National Institute for Science and Technology (NIST) statistical tests for randomness, experimental results of a Xilinx FPGA-based implementation show that IBS has a high pass rate that is consistent with pass rates of NIST-recommended reference random bits, validating the IID assumption.

The second way in which IBS differs from conventional syndrome coding is that IBS coding, when used with certain classes of PUFs (specifically, those with real-valued outputs), has a coding gain associated with the soft-decision encoding and decoding native to IBS. Soft-decision coding yields a higher coding gain than its hard-decision counterpart because the coder takes advantage of the confidence information of the bits presented at its input to make better coding decisions. Experimental results with a Xilinx FPGA-based implementation show that IBS reduces error-correcting code (ECC) complexity by approximately 16× to 64×, given certain design assumptions, while preserving the ability to correct errors across varied environmental conditions. A Xilinx Virtex-5 implementation showed no error correction failures when provisioned at 25°C and 1.0 V, and regenerated at –55°C and 1.1 V. Based on the number of tests run, the error rate is bounded well below 1 ppm (parts per million). We ran other conditions from –55°C to 125°C, at 1.0 V ± 10% as well, showing consistent results pointing to an error rate below 1 ppm.

Index-based syndrome coding

Consider a noisy *pseudorandom* source (one in which, for a given seed, the bitstream generated is predictable). Here, "noisy" means that the predictable bit stream could have some bit corruptions when regenerated. Examples of noisy pseudorandom sources include PUFs and biometric sources.

PUF with real-valued output

Now consider a noisy pseudorandom source with real-valued outputs. Each output value, rather than being a single bit (of 1 or 0), is instead real valued in the sense that the output value contains both polarity information (1 or 0) as well as confidence information (strength or confidence level of 1 or 0). One way to represent a real-valued output is to have each output value in 2s-complement representation. A + sign bit (1′b0) represents a 1-bit PUF output, and a – sign bit (1′b1) represents a 0-bit PUF output. The strength (or confidence level) of the 1 or 0 PUF output is represented by the remaining non-most-significant bits. Another representation of real-valued output is to show the PUF output bit in its native form (0 for a PUF output 0, 1 for a PUF output 1), and have a unary number of 1s representing output strength. Examples of PUFs with real-valued outputs include PUFs producing outputs resulting from oscillator comparisons with possibly selectable paths through each

Related Work

Several recent papers have cited the use of error correction with physical unclonable functions to generate cryptographic keys.[1-5]

Physical unclonable functions (PUFs)

Physical one-way functions were implemented using microstructures and coherent radiation, and an authentication application has been described.[1] Gassend et al. coined the term *physical unclonable function* and showed how PUFs could be implemented in silicon and used for authentication and cryptographic applications.[2] Many other silicon realizations of PUFs have been proposed.[6-9] It has been shown that some proposed PUFs can be modeled or reverse-engineered,[9] precluding their use in unlimited authentication applications. However, the focus of our work is on generating a fixed number of independent bits from a PUF, which are kept secret, and therefore these modeling attacks are not relevant. The security of the error correction scheme that ensures reliability of these bits is the important consideration and focus of our work.

Efficient and robust error correction

Bosch et al. suggested using two-stage coding to reduce error correction complexity through heavy use of repetition coding and conventional syndrome generation using XOR masking.[4] However, this work didn't directly address the case in which a PUF has DC bias and how that affects information leakage through repeat XOR masking of the same bit across multiple PUF output bits, nor do the error correction calculations directly account for voltage effects. In contrast, our work includes characterizing information leakage through repetition coding, ways to mitigate that via indexing, subjecting the syndrome through NIST tests and other correlation tests, and establishing a formal proof to show why *index-based syndrome* (IBS) does not contribute to additional min-entropy leakage. Additional contributions of our work include an IBS-only codec without the complexity of a conventional decoder; characterization across voltage variation and wider temperature variation; and empirical results showing no error correction failures across a wide range of temperature and voltage conditions, with an IBS-ECC (BCH (63, 30, $t = 6$)) configuration (IBS used in addition to Bose-Chaudhuri-Hochquenghen code BCH (63, 30, $t = 6$)) empirically producing error-free results.

Maes et al. also described soft-decision decoding with respect to PUFs using conventional soft-decision

oscillator ring (see Figure 1 for an example of PUF using an oscillator/arbiter hybrid approach).

Alternative approaches include synthesizing real-valued outputs from a PUF that outputs single-bit values. An example would be to take multiple readings of single-bit PUF output to obtain confidence information for that output value. The use of IBS with a real-valued PUF (RV-PUF) allows IBS to minimize information leak while increasing coding gain. If an RV-PUF is not used, information leak is still minimized, but coding gain benefits might be more limited.

Soft decision IBS encoder

Now consider a soft-decision encoder as Figure 2 shows. For each secret bit B, the encoder takes RV-PUF outputs R_i, $0 <= i <= q - 1$ and represents B as an s-bit index (pointer) P, which points to an R_i. Each R_i is a w-bit value that contains both polarity and confidence information—for example, a w-bit 2s complement number. $R_0, \dots, R_{q-1}$ forms a response segment R. So each response segment R consists of q R_i values, and each R_i value is w-bits wide. An index P that is s-bits wide points to one of these R_i values in the response segment to represent B.

For discussion purposes, each B bit is *enmapped* using nonoverlapping sets of $R_0, \dots, R_{q-1}$. Output of the IBS enmapper $P^{(B)}(.)$ depends on RV-PUF outputs $R_0, \dots, R_{q-1}$, as well as on B. The enmapper applies one of a family of functions, $P^{(B)}(.)$—which is indexed by the value B being encoded—to the sequence of device-specific values. For example, a 1-bit input for B has two functions, $P^{(0)}(.)$ and $P^{(1)}(.)$. Each function takes as input the sequence of pseudorandom values, $R = (R_0, \dots, R_{q-1})$, and provides an s-bit index as an output—for instance, where $q <= 2^s$.

One example of an index-based enmapping function is based on the indices of the extreme values in the sequence:

$$P^{(B)}(R_0, \dots, R_{q-1}) \begin{cases} = \arg\min_i R_i & \text{if } B = 0 \\ = \arg\max_i R_i & \text{if } B = 1 \end{cases}$$

decoders.[5] Contributions of the work described in the main text include performing soft-decision decoding without explicit use (and added complexity) of a conventional soft-decision decoder, and characterizing results across voltage variation and a wider temperature variation.

Information leakage

The work we describe in this article is among the first to explicitly construct syndrome that leaks no information (specifically, syndrome that does not contribute to additional min-entropy loss) from an information-theoretic viewpoint. This work complements the work done by Dodis.[10] In particular, when IBS is used as a replacement for the code-offset method using XOR masking, it can be shown formally that the syndrome does not leak additional min-entropy. This result, for example, can be used to derive m' in a secure sketch (see Dodis' work for definition of m' and secure sketch).

References

1. P.S. Ravikanth, ''Physical One-Way Functions,'' PhD thesis, Department of Media Arts and Sciences, Massachusetts Inst. of Tech., 2001.
2. B. Gassend et al., ''Silicon Physical Random Functions,'' *Proc. ACM Computer Communication Security Conf.,* ACM Press, 2002, pp. 148-160.
3. G.E. Suh, ''AEGIS: A Single-Chip Secure Processor,'' PhD thesis, Electrical Eng. and Computer Science Dept., Massachusetts Inst. of Tech., 2005.
4. C. Bosch et al., ''Efficient Helper Data Key Extractor on FPGAs,'' *Proc. Workshop Cryptographic Hardware and Embedded Systems* (CHES 08), LNCS 5154, Springer, pp. 181-197.
5. R. Maes, P. Tuyls, and I. Verbauwhede, ''A Soft Decision Helper Data Algorithm for SRAM PUFs,'' *IEEE Int'l Symp. Information Theory* (ISIT 09), IEEE Press, 2009.
6. R. Maes, P. Tuyls, and I. Verbauwhede, ''Intrinsic PUFs from Flip-flops on Reconfigurable Devices,'' *Benelux Workshop Information and System Security* (WISSec 08), 2008.
7. D.E. Holcomb, W.P. Burleson, and K. Fu, ''Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags,'' *Conf. Radio Frequency Identification Security* (RFID 07), 2007.
8. M. Majzoobi, F. Koushanfar, and M. Potkonjak, ''Lightweight Secure PUFs,'' *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 08), IEEE CS Press, 2008, pp. 670-673.
9. M. Majzoobi, F. Koushanfar, and M. Potkonjak, ''Testing Techniques for Hardware Security,'' *Proc. Int'l Test Conf.* (ITC 08), IEEE CS Press, 2008, pp. 1-10.
10. Y. Dodis, L. Reyzin, and A. Smith, ''Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data,'' *Proc. Eurocrypt 2004,* LNCS 3027, Springer, 2004, pp. 523-540.

As an example of IBS encoding, let $q = 8$, $B = 1$, $R_0, \ldots, R_{q-1} = -3, -10, 25, 80, -94, -3, 8, -2$. In this example, $P = 3$ (pointing to 80), if the max/min criteria as we've described is used for $P^B(.)$.

B is generated in a manner such that it is independent of $R_0, \ldots, R_{q-1}$. For example, B can be derived from

- the same distribution that generates $R_0, \ldots, R_{q-1}$, with each response values IID. For example, $B = 1$ if $R_j >= 0$, else $B = 0$, with R_j generated independently from $R_0, \ldots, R_{q-1}$ (e.g., $j = -1$, or another value where $j \neq 0, \ldots, q-1$).
- a random number generator (RNG).
- any source independent of the source generating $R_0, \ldots, R_{q-1}$.

We conducted some use cases using the PUF to generate k output values. The polarity bits of these values are used (and magnitudes ignored) to form a k-bit secret. This k-bit secret is then fed into a conventional ECC encoder to produce $n - k$ bits of parity, where n is the ECC block size.

In one use case, the $n - k$ parity bits serve as the B bits, and additional PUF output bits (treated as real values) along with these B bits are fed into an IBS encoder to generate the indices to represent these $n - k$ parity bits. In a second case, the k bits serve as the B bits, and conventional ECC is not used. Again, additional PUF output bits (treated as real values) along with these B bits are fed into an IBS encoder to generate the indices to represent k bits from the PUF. In a third use case, both the k bits and $n - k$ bits serve as B, and additional PUF output bits (treated as real values) along with B are fed into an IBS encoder to generate the indices to represent the entire n bits, of which the first k bits are from the PUF and the last $n-k$ bits are from the parity encoder. [The second and third use cases are described later in "IBS standalone (IBS-S)" and "IBS with additional ECC (IBS-ECC)."] The point is that B can be any value, as long as it is independent of

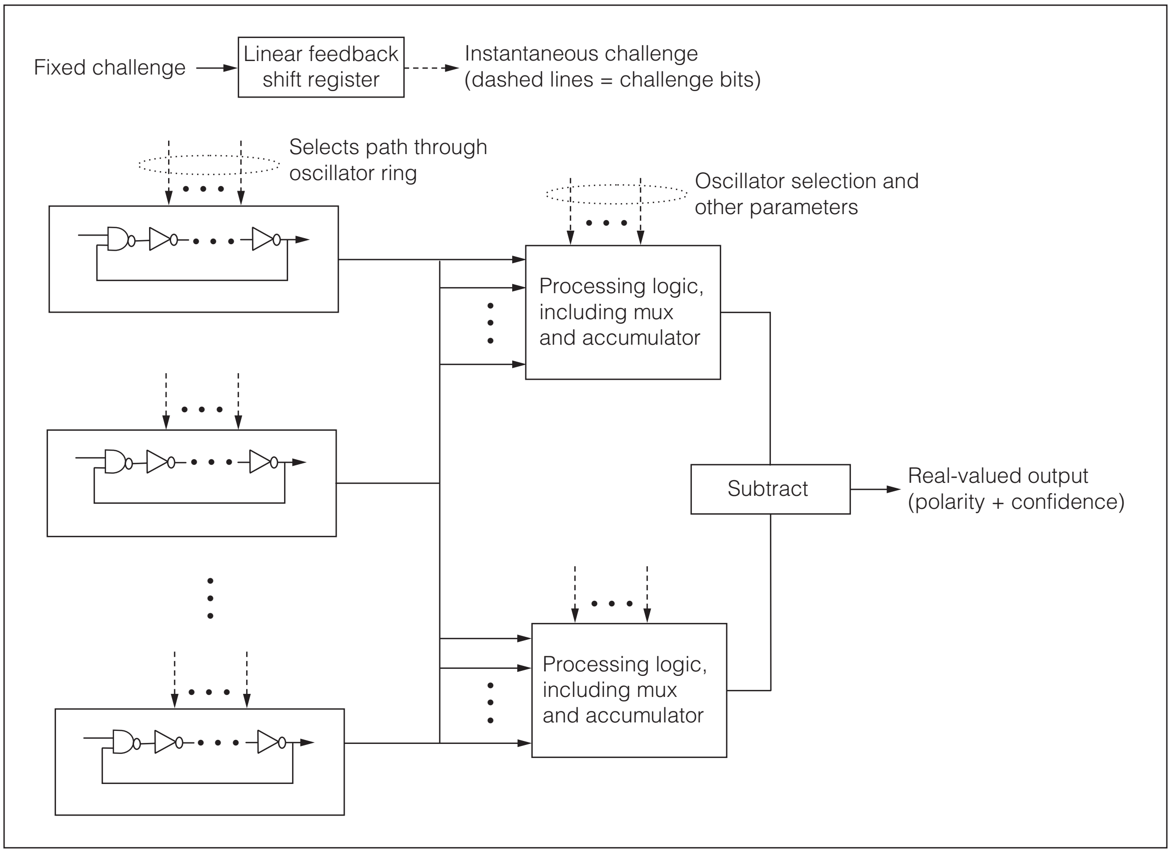

Fixed challenge
Linear feedback shift register
Instantaneous challenge (dashed lines = challenge bits)
Selects path through oscillator ring
Oscillator selection and other parameters
Processing logic, including mux and accumulator
Subtract
Real-valued output (polarity + confidence)
Processing logic, including mux and accumulator

Figure 1. Real-valued physical unclonable function (RV-PUF) using an oscillator and arbiter hybrid approach.

the additional PUF outputs generated that serve as $R_0, \ldots, R_{q-1}$. In fact, B can be derived from an RNG or any other source completely independently of the PUF generating $R_0, \ldots, R_{q-1}$, if an application can benefit from such an arrangement.

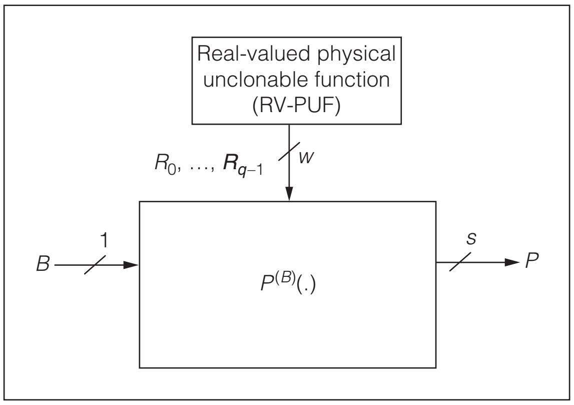


Figure 2. Soft-decision index-based syndrome encoder, where each PUF output response value is *w*-bits wide, and each index (pointer) value is *s*-bits wide.

Soft-decision IBS decoder

Now consider the soft-decision IBS decoder shown in Figure 3. The RV-PUF regenerates the device-specific values as $R' = (R'_0, \ldots, R'_{q-1})$. The values are not exactly equal, but it is expected that the values R_i and R'_i are at least approximately equal. As a result, the ordering by value is approximately the same, but not necessarily identical, as used in the encoder.

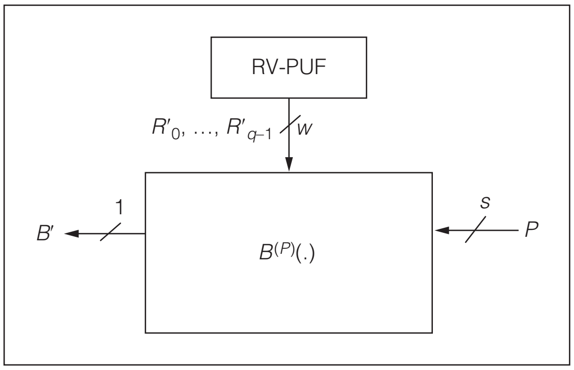
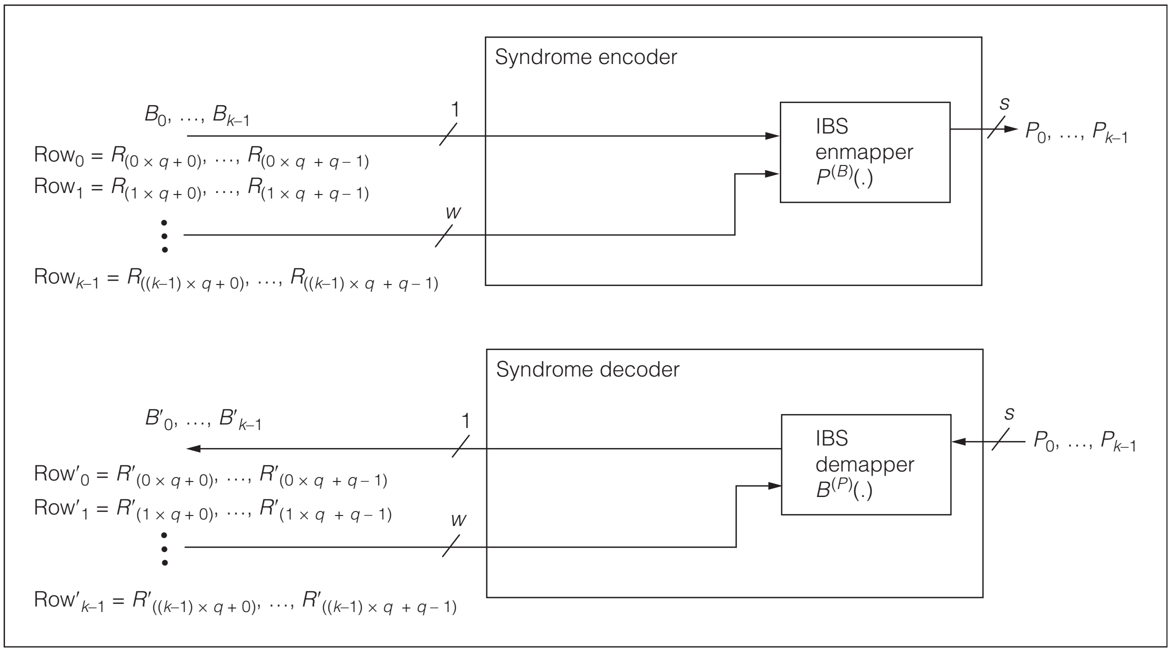


Figure 3. Soft-decision index-based syndrome decoder.



Syndrome encoder
$B_0, \ldots, B_{k-1}$
1
$\mathrm{Row}_0 = R_{(0\times q+0)}, \ldots, R_{(0\times q+q-1)}$
$\mathrm{Row}_1 = R_{(1\times q+0)}, \ldots, R_{(1\times q+q-1)}$
w
$\mathrm{Row}_{k-1} = R_{((k-1)\times q+0)}, \ldots, R_{((k-1)\times q+q-1)}$
IBS enmapper $P^{(B)}(.)$
s
$P_0, \ldots, P_{k-1}$
Syndrome decoder
$B'_0, \ldots, B'_{k-1}$
1
$\mathrm{Row}'_0 = R'_{(0\times q+0)}, \ldots, R'_{(0\times q+q-1)}$
$\mathrm{Row}'_1 = R'_{(1\times q+0)}, \ldots, R'_{(1\times q+q-1)}$
w
$\mathrm{Row}'_{k-1} = R'_{((k-1)\times q+0)}, \ldots, R'_{((k-1)\times q+q-1)}$
IBS demapper $B^{(P)}(.)$
s
$P_0, \ldots, P_{k-1}$

Figure 4. IBS-S (index-based syndrome-standalone) configuration.

The IBS decoder includes an IBS demapper $B^P(.)$, which accepts the index value P and outputs an estimate B', which in normal operation is expected to regenerate the original value B. This regeneration is done by first applying a function $B^{(P)}(.)$ to the sequence of values, $R' = (R'_0, \ldots, R'_{q-1})$, to produce a reconstruction of the value B.

One example of the demapping function $B^P(.)$ that is compatible with the maximum and minimum encoding function is as follows:

$$B' = B^{(P)}(R'_0, \ldots, R'_{q-1}) = \text{sign of } (R'_p)$$

and

$$\text{sign of } (R'_p)\begin{cases} = 0 & \text{if } R'_p < 0 \\ = 1 & \text{if } R'_p \geq 0 \end{cases}$$

In some examples, the decoding function is

$$B^{(P)}\ (R'_0, \ldots, R'_{q-1}) = Pr(B = 1 |\ P, R'_0, \ldots, R'_{q-1})$$

based on a probabilistic model of the encoding process, thereby generating a soft-bit regeneration of the original data.

Continuing with the example from the soft-decision IBS encoder, let $q = 8$, $P = 3$, $R'_0, \ldots, R'_{q-1} = -4, -11, 77, 84, -92, -8, 2, -1$. In this example, B' = sign of $(84) = + = 1$, which equals the original encoded B of 1.

Note that these encoding and decoding functions can be understood to be compatible on the basis of the observation that in encoding, the device-specific maximum value is almost always the most positive (and in some rare cases, if all values are negative, the least negative), and therefore, that value's regeneration is expected to at least remain positive, even if it's not the maximum of the regenerated sequence. Similarly, the minimum value in encoding is expected to remain negative when it is regenerated. If further error correction is required, a conventional error correction codec can be instantiated with the IBS enmapper and demapper. In other applications, errors might be ignored, and comparisons based on a Hamming threshold could be used.

Two IBS configurations

We evaluate two IBS configurations here: IBS in a standalone mode (IBS-S), without other forms of error correction, and IBS with additional error correction (IBS-ECC).

IBS standalone (IBS-S). In this use case, the IBS enmapper/demapper is used without other forms of error correction. Secret bits are broken into $B_0, \ldots, B_{k-1}$ where k is the total length of secret bits, each of which is encoded respectively using $P_0, \ldots, P_{k-1}$, by finding the max or min value [assuming max/min criteria is used for $P^{(B)}(.)$] in each of the k rows of R_i values, as shown in Figure 4. If the PUF does not output confidence information (e.g., RV-PUF is

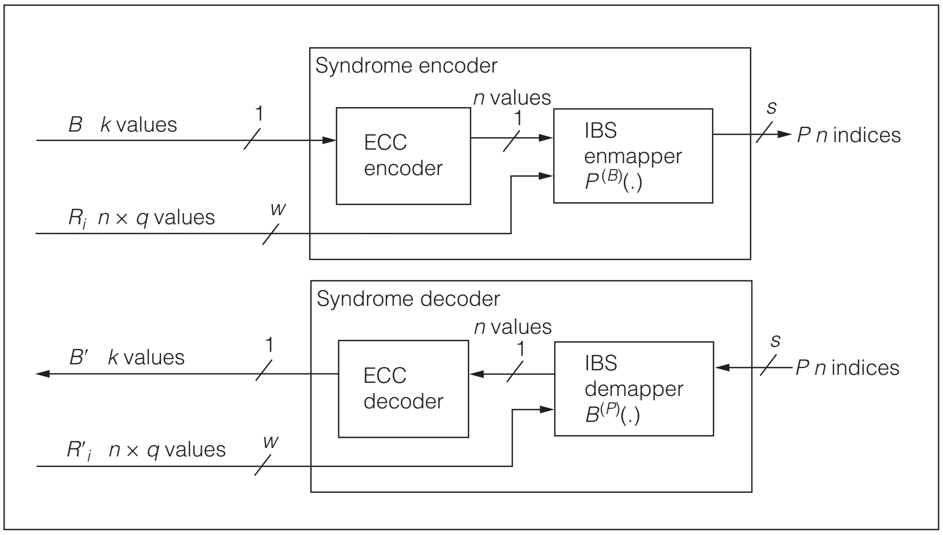


Figure 5. IBS-ECC (index-based syndrome error-correcting code) configuration.

not used), a random selection among matching bits in the row of R_i values is then made, and if none of the bits match, a random selection among all the (nonmatching) bits in the row is made. In a simple decoder example, $P_0, \ldots, P_{k-1}$ is used to reconstruct $B'_0, \ldots, B'_{k-1}$ by using each index P to select the appropriate R_i in each row, and analyzing its sign bit.

As we will explain, IBS in standalone configuration (without additional error correction mechanisms), has error correction and error reduction capabilities that might be sufficient for certain classes of applications.

As an example, assume $q = 8$, $k = 2$, use of max/min criteria, and a simple decoder that looks at the sign bit of a PUF output value being pointed to by P to extract B'. Suppose that at the encoder input we have

$$B_0, B_1 = 1, 0$$

$$\begin{aligned}\mathrm{Row}_0 &= \mathrm{R}_0, \ldots, \mathrm{R}_7\\ &= -3, -10, 25, 80, -94, -3, 8, -2\end{aligned}$$

$$\begin{aligned}\mathrm{Row}_1 &= R_8, \ldots, R_{15}\\ &= 12, 8, -21, -3, -9, -30, 85, 34\end{aligned}$$

The output of the encoder will be as follows:

$P_0 = 3$ (looks up max value of 80)

$P_1 = 5$ (looks up min value of -30)

Now, suppose that at the decoder input we have

$$P_0, P_1 = 3, 5$$

$$\begin{aligned}\mathrm{Row}'_0 &= R'_0, \ldots, R'_7\\ &= -4, -11, 77, 84, -92, -8, 2, -1\end{aligned}$$

$$\begin{aligned}\mathrm{Row}'_1 &= R'_8, \ldots, R'_{15}\\ &= 16, 12, -25, -1,\\ &\quad -13, -24, 81, 45\end{aligned}$$

The output of the decoder will be

$$B'_0 = \mathrm{sign}(84) = + = 1$$

$$B'_1 = \mathrm{sign}(-24) = - = 0$$

thus recovering B_0, $B_1 = 1, 0$.

IBS with additional ECC (IBS-ECC). If additional error correction is required, IBS can be instantiated with one (or more) stages of other ECC (see Figure 5 for an example). A total of $n \times q$ PUF outputs are generated, to derive n indices. Note that the configuration in Figure 5 results in a soft-decision *syndrome* encoder/decoder system, even if we use a conventional hard-decision ECC encoder/decoder. IBS therefore produces coding gain associated with soft-decision decoders without explicit use (and added complexity) of conventional soft-decision ECC, as Maes et al. proposed.[8]

As an example, let us assume $q = 8$, $k = 4$, $n = 7$, the use of max/min criteria, and a simple decoder that looks at the sign bit of the PUF output value being pointed to by P to extract B'. The BCH (7, 4, $t = 1$) ECC used has the following generator matrix:

$$G = \begin{bmatrix} 1\,0\,0\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,1\,1 \\ 0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,1\,1 \end{bmatrix}$$

Suppose that at the ECC encoder input we have

$$B_0, B_1, B_2, B_3 = 1, 0, 0, 0$$

At the ECC encoder output, we then have

$$1, 0, 0, 0, 1, 0, 1$$

This result is input to the IBS enmapper, along with

$$\begin{aligned}\mathrm{Row}_0 &= R_0, \ldots, R_7\\ &= -3, -10, 25, 80, -94, -3, 8, -2\end{aligned}$$

$$\begin{aligned}\mathrm{Row}_1 &= R_8, \ldots, R_{15}\\ &= 12, 8, 0, -2, -1, -3, 85, 34\end{aligned}$$

$$\ldots$$

$$\text{Row}_6 = R_{48}, \ldots, R_{55} = 3,\ 5,\ 8,\ -15,\ -31,\ 45,\ -15,\ 102$$

The output of the IBS enmapper will be

$P_0 = 3$ (looks up max value of 80)
$P_1 = 5$ (looks up min value of -3)
...
$P_6 = 7$ (looks up max value of 102)

Now, suppose that at the IBS demapper input we have the following:

$$P_0, P_1, \ldots, P_6 = 3,\ 5,\ \ldots,\ 7$$
$$\text{Row}'_0 = R'_0, \ldots, R'_7 = -4,\ -11,\ 77,\ 84,\ -92,\ -8,\ 2,\ -1$$
$$\text{Row}'_1 = R'_8, \ldots, R'_{15} = 16,\ 12,\ 1,\ -1,\ 2,\ 3,\ 81,\ 45$$
...
$$\text{Row}'_6 = R'_{48}, \ldots, R'_{55} = -1,\ 1,\ 2,\ -12,\ -38,\ 43,\ -13,\ 99$$

The output of the IBS demapper will be (note the intentional bit error for B'_1):

$$B'_0 = \text{sign}(84) = + = 1$$
$$B'_1 = \text{sign}(3) = + = 1$$
...
$$B'_6 = \text{sign}(99) = + = 1$$

The output of the ECC decoder will be (as a result of error correction):

1, 0, 0, 0

thus recovering $B_0, B_1, B_2, B_3 = 1, 0, 0, 0$.

Design implementation

We implemented a design using IBS to derive secure and robust keys from PUFs in Xilinx Virtex-4 LX25/LX60 and Xilinx Virtex-5 LX50 FPGAs. The design consisted of three major components. First was a programmable IBS enmapper/demapper supporting 0th-order to 5th-order indices and built-in repetition-coder supporting 1×, 3×, and 5× repetition coding. The second component was a programmable BCH (63) encoder/decoder supporting BCH codes from $t = 1$ to $t = 6$. The third component was a physical unclonable function circuit (Figure 1) with 64 challenge bits.

Also included in the design were various debug facilities, including the ability to obtain raw oscillator frequencies. We used these to collect error statistics at various points in the processing pipeline, and to provide a direct user-chosen stimulus in various parts of the processing pipeline and to observe downstream behavior. The design complexity was dominated by the BCH (63) ECC decoding core, which had on the order of 400 registers plus supporting combinatorial logic. In one fully programmable and fully instrumented version of the design, the combined PUF-based key generator consisting of all three major components and debug facilities used approximately 1,000 slices in a Xilinx-5 LX50 device, or roughly 14% of the slice count.

This design has a large error correction margin. A design that does not require as large an error correction margin, and having fewer debug facilities and less programmability, will be smaller.

Security analysis

Here, we first show the shortcomings of conventional syndrome generation methods. Then we demonstrate how index-based syndrome is superior using information theoretic arguments. The theoretic results are then affirmed by empirical test results using a FPGA implementation and NIST statistical tests for randomness.

Conventional methods using XOR masking

Consider conventional syndrome generation methods, which typically perform logical bitwise XOR of parity information with PUF outputs. Assume the PUF generates n bits of information. The first k bits are fed into a conventional ECC encoder to produce $n-k$ bits of parity. Syndrome bits are then formed by performing a bitwise XOR operation of the $n-k$ parity bits output by the ECC encoder with the last $n-k$ bits from the n PUF bits. If a PUF systematic bias exists, information is leaked via the syndrome. This is especially true if repetition code is used. Published experimental systematic bias values include 46.15% for a ring oscillator PUF, 23% for an early version of an arbiter PUF,[6] 49.97% for an SRAM-based PUF,[9] and approximately 10% for a flip-flop PUF.[8] The percentage represents bias toward 1. The ideal bias value is 50%, meaning 1s and 0s are statistically equally likely.

The simplest example of repetition code is a binary repetition (3, 1, 3) code. It repeats each bit three times, so that a 0 is encoded onto the vector

(000) and a 1 onto the vector (111), using a generator matrix:

$$G = [1\ 1\ 1]$$

To produce a syndrome using XOR masking, a syndrome encoder performs logical bitwise XOR of these three bits with three PUF output bits.

Consider the case in which the PUF has a bias of 51%, meaning that out of 100 PUF output bits, an average of 51 bits are 1s. If a syndrome encoder performs logical bitwise XOR of a secret bit with a single bit of PUF output, and the result is 0, there is a 51% chance that the secret bit is 1 and a 49% chance that the secret bit is 0, and vice versa if the result is 1. Now if a syndrome encoder performs logical bitwise XOR of that same secret bit with a single bit of a PUF output, and generates additional syndrome bits by performing logical bitwise XOR of the same secret bit with an additional 99 PUF output bits, then if the secret bit is 1, the mask would statistically have 49 1s and 51 0s. Alternatively, if the secret bit is 0, the mask would statistically have 51 1s and 49 0s. Clearly, with repetition coding, in which the repetition encoder repeats the secret bit, bitwise XOR masking of the repeated bits with PUF output bits reveals an additive increase in information the longer the repetition code. In a binary repetition (33, 1, 33) code, where a secret bit is repeated 33 times, the resulting 33-bit syndrome produced using bitwise XOR with 33 PUF output bits would statistically leak information every time if, out of 33 bits produced by the PUF, 17 bits are 1s and 16 bits are 0s (51.52% bias) or vice versa.

In general, if a binary repetition (r, 1, r) code is used, a PUF bias of more than [ceiling(r/2)]/r or a PUF bias of less than [floor(r/2)]/r would cause the syndrome (XOR mask) to leak a secret bit each time, statistically speaking. Bits leaked can be calculated as follows, as a function of PUF bias and the repetition code used:

$$\begin{aligned}&\text{bits leaked per secret bit} =\\ &\quad \text{abs}\,(\text{PUF bias} - 0.5)/\text{abs}\{[\text{ceiling}(r/2)]/r - 0.5\}\end{aligned}$$

where abs is the absolute value operator.

For example, if a PUF has a bias of 0.51 (or a bias of 0.49), a binary repetition (9, 1, 9) code would leak 1 bit out of 5.6 bits encoded. In other words, more than 1/6 of the information would be leaked though the syndrome.

IBS approach, theoretical result

Looking again at the IBS enmapper, we want to know whether P leaks any information about B. P is assumed to be public information, and B is a secret bit (which, along with other secret B bits, is used to derive keys or seeds to generate keys).

$R_0, \ldots, R_{q-1}$ is assumed to be private information (for example, a biometric reading, or outputs of a silicon PUF inside a chip). We assume that although the reading itself is not known to the adversary, he or she has possible access to other readings from other biometric hosts or other devices (his or her fingerprints and those of his or her cohorts, for instance, or reverse-engineered PUF devices obtained from, say, eBay). Accordingly, *systematic* (population) statistics (such as systematic DC bias), but not the statistical properties of a particular *individual* biometric reading or PUF silicon device, can be inferred.

Based on prior research, PUFs have interchip or interclass variations in that cross correlation of outputs from different PUFs are quite different, but possibly with some systematic bias. Therefore, PUF output response bits are often modeled as an IID normal distribution with mean μ and standard deviation σ. Use of an IID normal distribution to model PUF responses based on empirical data can be found for the arbiter-based PUF[6] and for the SRAM PUF.[10] It is assumed, for the purpose of proof, that each of the $n \times q$ R_i values generated is independent of one another. For example, each R_i could be derived from disjoint oscillator pairs, with no oscillator reuse in deriving different R_i values. We also assume that each chip provisions only one secret key or secret seed, and provisioning is disabled once the secret has been provisioned. Alternately, the chip can be built with a fixed challenge to generate only one or a few secrets.

Now, consider the case in which IBS is used. It can be proved mathematically, under certain assumptions, that there is no reduction in min-entropy, even if a progressive application of an otherwise leaky code such as repetition code is used. Similar arguments can be made when IBS coding is applied with other forms of error correction. In this sense, IBS is superior in security compared to conventional XOR masking.

According to results in the "Mathematical Model for IBS Enmapper and Proofs" sidebar, revealing index $\underline{P}$ does not lead to a reduction in min-entropy in $\underline{B}$. This was proven without any assumptions about the distribution for PUF outputs, except that it is IID, meaning that even if there is a DC bias (or for that matter, any higher-order bias), the proof still holds true. Furthermore, no assumption about $\underline{B}$ is made except that it is independent from $R_0, \ldots, R_{q-1}$, implying that $\underline{B}$ values can be correlated with one another (as is the case when repetition code or other forms of conventional error correction are used), and the proof still holds true.

IBS approach, empirical data

Experimental results derived from a Xilinx Virtex-4 implementation affirm the formal mathematical results. We used NIST statistical tests for randomness to analyze the randomness of syndrome encoder outputs (P) given certain sequences of input bits (B). Three-bit indices were used. For each input sequence, we analyzed 100 million syndrome bits (from 100/3 = 34 million indices). These syndrome bits were formed by serializing concatenated 3-bit syndrome indices. Four test sequences, each containing 34 million bits, were injected into the B input of an IBS encoder.

Specifically, the 0th-order DC sequences of all 0s and all 1s, and the 1st-order AC sequences of alternating 1010s and 0101s were used for B. For each sequence, 34 million indices (P values) were generated. Each P chose either the maximum value from the PUF output values if B was 1, or the minimum value from the PUF output if B was 0. In all, 367 million PUF output bits (33 $\times$ 8 million) were generated for each test sequence, all from the same starting 64-bit challenge seed. So, each bit B was encoded as a 3-bit index P, choosing the best of nonoverlapping 8 PUF output values.

We performed the analysis explained here before we had completely built the IBS hardware in the FPGAs, as a part of the design derivation and refining process. So, a large part of the design was emulated in software but using PUF information derived across four Xilinx Virtex-4 LX25 FPGAs. Each FPGA was used to derive a single RV-PUF, corresponding to Figure 1. Success rates for each of the 15 NIST tests across the four LX25 chips were comparable with the success rates derived from a NIST-recommended set of random numbers as input (see Table 1).

Results showed that syndrome bits are tested to be random, and thus it is difficult to infer input bits B from indices P. More specifically, take note of the results of the 0th-order (DC) sequences:

- an input sequence of B consisting of all 0s produces random syndrome bits, implying $Pr(P \mid B = 0) = 1/8$.
- an input sequence of B consisting of all 1s produces random syndrome bits, implying $Pr(P \mid B = 1) = 1/8$.

Furthermore,

$$Pr(P) = Pr(P \mid B = 0) \times Pr(B = 0) + Pr(P \mid B = 1) \times Pr(B=1) = 1/8 \times (Pr(B = 0) + Pr(B = 1)) = 1/8$$

Therefore,

$$Pr(P \mid B) = Pr(P)$$

showing from empirical results that indices P and input bits B are statistically independent.

From the empirical results, we could also gain a level of confidence in the belief that the PUF output bits can be treated as IID. If they were not, it's highly unlikely that the syndrome produced (e.g., by applying B consisting of all 0s) would be random unless somehow the silicon-based PUF could arrive at a strange distribution to make that the case. The empirical results therefore affirm the assumption (held by, for example, Lim[6] and Maes[10]) that the PUF output can be treated as IID.

Custom-constructed correlation tests analyzing syndrome encoder input B versus output P confirm results from the standardized NIST randomness tests. More than 95% of the correlation results were within two standard errors of the ideal uncorrelated value, and the few outliers that were present did not stray far from two standard errors away from the ideal.

Coding gain associated with IBS

Here we present empirical results in applying index-based syndrome coding to PUFs in devices representing two process geometries. Specifically, 90-nm Xilinx Virtex-4 and 65-nm Xilinx Virtex-5 devices were used. The results show that coding gain associated with IBS is significant. This has the effect of reducing total error correcting code logic complexity, and increases stability of regenerated PUF-derived secret bits across varied environmental conditions.

Results on Xilinx Virtex-4

Figure 6 shows the coding gain associated with using 3-bit indices and 3$\times$ repetition coding. The PUF was provisioned under a nominal temperature of 20°C but under voltage variations of 1.2 V $\pm$ 10%. The left plot shows the error curve, black representing regeneration under nominal voltage and temperature (20°C, 1.2 V), and other curves showing regeneration under the four corners (voltage and

Mathematical Model for IBS Enmapper and Proofs

We can define several random variables to mathematically represent elementary operations of the IBS enmapper. The mathematical properties for random variables are as follows.

For secret bit $\underline{B}$ (underlining denotes a random variable):

$\underline{B} \in \{0, 1\}$ is generated such that it is independent of $\underline{R}_0, \ldots, \underline{R}_{q-1}$.

Note: The case where $\underline{B}$ takes on the sign of (or is otherwise derived from and therefore not independent from) one of $\underline{R}_0, \ldots, \underline{R}_{q-1}$ values (e.g., one pointed to by $\underline{P}$) can also be proved to be information theoretically secure. For brevity, we do not include this case.

For response segment ($\underline{R}$):

$\underline{R} \in \{\text{signed integer}\}^q$ consists of q signed integers $\underline{R}_i$, $0 <= i < q$, where $\underline{R}_i$ is independent and identically distributed (IID).

For pointer ($\underline{P}$):

$\underline{P} \in \{0, 1 \ldots q-1\}$,

If $\underline{B} = 1$, $\underline{P}$ = index of f_1 $(\underline{R}_0 = r_0, \ldots, \underline{R}_{q-1} = r_{q-1})$
If $\underline{B} = 0$, $\underline{P}$ = index of f_0 $(\underline{R}_0 = r_0, \ldots, \underline{R}_{q-1} = r_{q-1})$

$f_1(.)$, $f_0(.)$ are functions that output one of the following values: $r_0, \ldots, r_{q-1}$.
f_1, f_0 are such that $f_1(.) \neq f_0(.)$, except in the trivial case when $r_0 = r_1 = \ldots = r_{q-1}$.
f_1, f_0 are such that

$$\begin{aligned} f_1(\underline{R}_0 = r_0, \ldots, \underline{R}_{q-1} = r_{q-1}) &= f_1(r_0, \ldots, r_{q-1}) \\ &= f_1(r_{\pi(0)}, \ldots, r_{\pi(q-1)}) \\ &= f_1(\underline{R}_0 = r_{\pi(0)}, \ldots, \underline{R}_{q-1} \\ &\quad = r_{\pi(q-1)}), \text{ and similarly} \\ f_0(\underline{R}_0 = r_0, \ldots, \underline{R}_{q-1} = r_{q-1}) &= f_0(r_0, \ldots, r_{q-1}) \\ &= f_0(r_{\pi(0)}, \ldots, r_{\pi(q-1)}) \\ &= f_0(\underline{R}_0 = r_{\pi(0)}, \ldots, \underline{R}_{q-1} \\ &\quad = r_{\pi(q-1)}), \end{aligned}$$

where π is an index permutation function.

Examples of f_1 and f_0 include

max and min functions;
2nd-most max and 2nd-most min functions;
max deviation from mean and min deviation from mean; and
max deviation from 0 and min deviation from 0.

For "index of" operator, if multiple indices produce the same f_1 result, a random index among those indices is chosen; likewise, for f_0.

Now, consider an adversary who has possession of a certain pointer P. The claim, stated simply, is that P leaks no information about B. That is, $H(\underline{B}) = H(\underline{B} \mid \underline{P})$, where H is a Shannon entropy measure. However, to be conservative and to allow results to be more readily adapted to the research by Dodis et al. in constructing secure sketches and fuzzy extractors,[1] we also express our results using min-entropy:

$$H_\infty(.) \equiv -\log_2(Pr_{\max}(.))$$

(where $\equiv$ means definition), since we want to account for a worst-case "guessing" probability.

We also define average min-entropy per the work of Dodis et al.[1] as

$$\tilde{H}_\infty(\underline{X} \mid \underline{Y}) \equiv -\log_2(E_{y \leftarrow \underline{Y}}[2^{-H_\infty(\underline{X} \mid \underline{Y}=y)}])$$

Theorem 1: $\underline{P}$ and $\underline{B}$ are independent, assuming the PUF has IID outputs and $\underline{B}$ is independent of $\underline{R} = \underline{R}_i$, $0 <= i < q$.

Proof: To prove that $\underline{B}$ and $\underline{P}$ are independent, it is equivalent to proving that the probability of $\underline{P} = p$ remains the same regardless of the value of $\underline{B}$:

$$Pr(\underline{P} = p \mid \underline{B} = b) = Pr(\underline{P} = p \mid \underline{B} = b'), \tag{A}$$

since Equation A implies that $\underline{B}$ and $\underline{P}$ are independent.

$$\begin{aligned} Pr(\underline{P} = p) &= \sum_{\text{all b}} Pr(\underline{P} = p, \underline{B} = b) \\ &= \sum_{\text{all b}} Pr(\underline{P} = p \mid \underline{B} = b)\, Pr(\underline{B} = b) \\ &= Pr(\underline{P} = p \mid \underline{B} = b') \sum_{\text{all b}} Pr(\underline{B} = b) \\ &\qquad \text{(using Equation A)} \\ &= Pr\,(\underline{P} = p \mid \underline{B} = b') \end{aligned}$$

We first note that

$$\begin{aligned} Pr(\underline{P} = p \mid \underline{B} = b) &= \sum_{\text{all r}} Pr(\underline{P} = p,\ \underline{R} = r \mid \underline{B} = b) \\ &= \sum_{\text{all r}} Pr(p = f(b,\ r),\ \underline{R} = r \mid \underline{B} = b) \end{aligned} \tag{B}$$

Here, f unifies f_1 and f_0 into a single function, with an additional input b to direct f to select either f_1 or f_0.

As stated in Theorem 1, $\underline{B}$ is independent of $\underline{R}$; therefore, Equation B becomes

$$\sum_{\text{all r}} Pr(p = f(b, r), \underline{R} = r \mid \underline{B} = b)$$
$$= \sum_{\text{all r}} Pr(p = f(b, r), \underline{R} = r)$$
$$= \sum_{r \text{ s.t. } p=f(b, r)} Pr(\underline{R} = r)$$
$$= \sum_{r_0, \ldots, r_{q-1} \text{ s.t. } p=f(b, r_0, \ldots, r_{q-1})} Pr(\underline{R} = r_0, \ldots, r_{q-1})$$

(expressing r explicitly as q values) (C)

We have now transformed the problem into proving that Equation C is independent of b.

Now, let's create a permutation function π, where the indexed entries for f_1 and f_0 are swapped. For example, if f_1 = max and f_0 = min, the max and min entries are swapped. If there are multiple max entries, for example, a random one is chosen to be swapped; this is true also if there are multiple min entries. In the unlikely event that all values in $r_0, \ldots, r_{q-1}$ are equal, a random swap is performed.

Therefore,

$$p = f(b, r_0, \ldots, r_{q-1}) = f(b', r_{\pi(0)}, \ldots, r_{\pi(q-1)}), \qquad \text{(D)}$$

where the substitution of b by b' corresponds to the permutation π.

Now let τ be the inverse of permutation π (to reverse the index swap).

Recall that Equation B = Equation C. Adding results from Equation D, we get

$$Pr(\underline{P} = p \mid \underline{B} = b)$$
$$= \sum_{r_0, \ldots, r_{q-1} \text{ s.t. } p=f(b, r_0, \ldots, r_{q-1})} Pr(\underline{R} = r_0, \ldots, r_{q-1})$$
$$= \sum_{r_0, \ldots, r_{q-1} \text{ s.t. } p=f(b', r_{\pi(0)}, \ldots, r_{\pi(q-1)})} Pr(\underline{R} = r_0, \ldots, r_{q-1})$$
$$= \sum_{r_0, \ldots, r_{q-1} \text{ s.t. } p=f(b', r_0, \ldots, r_{(q-1)})} Pr(\underline{R} = r_{\tau(0)}, \ldots, r_{\tau(q-1)})$$
$$= \sum_{r_0, \ldots, r_{q-1} \text{ s.t. } p=f(b', r_0, \ldots, r_{(q-1)})} Pr(\underline{R} = r_0, \ldots, r_{q-1})$$

(since individual response r_i are IID)

$$= Pr(\underline{P} = p \mid \underline{B} = b')$$

thus proving Equation A and the independence of $\underline{P}$ and $\underline{B}$. Note that the result also holds for $\underline{B}$ representing a code word, and $\underline{P}$ representing the sequence of helper information (indices) for each code word bit. In this case, $\underline{R}$ represents all the PUF output bits necessary to generate all the indices, and π and τ represent f_1/f_0 swaps corresponding to each code word bit.

Theorem 1 implies the following:

First, additional knowledge of index $\underline{P}$ does not leak additional information about $\underline{B}$ (that is, $H(\underline{B} \mid \underline{P}) = H(\underline{B})$). This is true even when the adversary is given the knowledge of statistical distribution for PUF output $\underline{R}$ and input bit $\underline{B}$. It's also true when $\underline{B}$ values are correlated with one another (e.g., as k and parity values in the context of traditional error correction). Finally, this is true even if the PUF output distribution is not perfectly normal (or not normal at all), as long as IID still holds.

Second, given a population of PUFs, and an adversary who has broken those PUFs and gathered systematic (population) information, additional information of $\underline{P} = p$ on a new PUF does not aid the adversary, from an information-theoretic standpoint, in recovering $\underline{B}$ on that new PUF.

Third, $\underline{P}$ does not induce min-entropy loss on $\underline{B}$:

$$\tilde{H}_\infty(\underline{B} \mid \underline{P}) = H_\infty(\underline{B})$$

Proof:

$$H_\infty(\underline{B} \mid \underline{P} = p) \equiv -\log_2(Pr_{\max}(\underline{B} \mid \underline{P} = p))$$
$$= -\log_2(\max\{Pr(\underline{B} = 1 \mid \underline{P} = p), Pr(\underline{B} = 0 \mid \underline{P} = p)\})$$
$$= -\log_2(\max\{Pr(\underline{B} = 1), Pr(\underline{B} = 0)\})$$
$$= H_\infty(\underline{B}) \quad \text{(due to Theorem 1)} \qquad \text{(E)}$$

Now, consider how much information about $\underline{B}$ in terms of min-entropy is revealed when $\underline{P}$ is revealed, when $\underline{P}$ is taken all possible values of p. We use average min-entropy measure as defined by Dodis et al.:[1]

$$\tilde{H}_\infty(\underline{X} \mid \underline{Y}) \equiv -\log_2(E_{y \leftarrow \underline{Y}}[2^{-H_\infty(\underline{X} \mid \underline{Y} = y)}])$$

In our example, $\underline{X} = \underline{B}$, $\underline{Y} = \underline{P}$, and $y = p$. Now, we compute the expected value over $\underline{P}$:

$2^{-H_\infty(\underline{B} \mid \underline{P}=0)} = 2^{-H_\infty(\underline{B})}$ since $\underline{P}$ is independent of $\underline{B}$, as shown in Equation E.

The same would apply for all other values of $\underline{P}$, (e.g., $\underline{P} = 1, \underline{P} = 2, \ldots, \underline{P} = q - 1$).

Therefore,

$$\tilde{H}_\infty(\underline{B} \mid \underline{P}) = -\log_2(E_{p \leftarrow \underline{P}}[2^{-H_\infty(\underline{B} \mid \underline{P}=p)}])$$
$$= -\log_2([q \times 2^{-H_\infty(\underline{B})}]/q)$$
$$= -\log_2([2^{-H_\infty(\underline{B})}]) = H_\infty(\underline{B})$$

Reference

1. Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *Proc. Eurocrypt 2004,* LNCS 3027, Springer, 2004, pp. 523-540.

Table 1. National Institute for Science and Technology (NIST) statistical tests for randomness: success ratio for PUF syndrome indices.

Statistical test	**NIST input parameters**	**Success ratio (%)* (Chip A)**	**Success ratio (%) (Chip B)**	**Success ratio (%) (Chip C)**	**Success ratio (%) (Chip D)**	**Reference random bits from George Marsaglia's *Random Number CDROM*** (%)**
Frequency	—	98	100	98	99	98
		97	97	99	97	
		99	98	98	97	
		98	98	97	99	
BlockFrequency	128	99	98	100	98	97
		99	100	100	100	
		98	100	100	100	
		98	96	99	100	
CumulativeSums	—	98–99	99–100	98–99	97–99	98–99
		97–98	97–98	99–99	97–97	
		98–98	98–98	99–99	98–99	
		97–98	99–99	98–99	99–99	
Runs	—	98	100	99	98	100
		97	98	98	99	
		98	99	100	88	
		98	99	100	97	
LongestRun	—	100	100	97	96	97
		99	100	99	100	
		99	97	100	100	
		98	100	98	98	
Rank	—	100	97	96	99	100
		99	99	100	100	
		99	99	98	99	
		99	100	98	100	
FFT	—	100	100	100	100	100
		100	99	100	100	
		100	100	100	100	
		100	100	100	100	
NonOverlappingTemplate	9	95–100	94–100	95–100	95–100	95–100
		97–100	97–100	97–100	95–100	
		97–100	97–100	97–100	97–100	
		97–100	97–100	95–100	97–100	
OverlappingTemplate	9	98	99	98	99	97
		99	98	99	98	
		99	98	98	100	
		100	100	100	99	

(Continued)

Table 1. (Continued)

Statistical test	NIST input parameters	Success ratio (%)* (Chip A)	Success ratio (%) (Chip B)	Success ratio (%) (Chip C)	Success ratio (%) (Chip D)	Reference random bits from George Marsaglia's *Random Number CDROM*** (%)
Universal	—	99	100	100	100	100
		98	99	100	96	
		99	99	97	98	
		99	98	99	99	
ApproximateEntropy	10	99	99	98	100	100
		100	100	98	98	
		97	99	100	99	
		99	100	99	99	
RandomExcursions	—	96–100	94–100	96–100	98–100	98–100
		97–100	98–100	99–100	97–100	
		100–100	97–100	96–100	96–100	
		98–100	98–100	96–100	100–100	
RandomExcusionVariant	—	96–100	97–100	98–100	98–100	93–100
		98–100	96–100	97–100	97–100	
		97–100	95–100	96–100	95–100	
		95–100	94–100	95–100	96–100	
Serial	16	99–99	97–100	98–100	98–99	98–100
		98–99	97–97	98–99	98–99	
		98–100	97–98	100–100	98–100	
		99–100	98–100	98–99	98–98	
LinearComplexity	500	100	100	97	98	100
		99	98	100	100	
		98	97	96	99	
		99	99	99	100	
Cumulative *p*-values		100	100	100	99	100
		(752/752) pass	(752/752) pass	(752/752) pass	(750/752) pass	(188/188) pass
Cumulative proportions		99	99	99	99	98
		(748/752) pass	(747/752) pass	(749/752) pass	(747/752) pass	(184/188) pass

*The first line is for the DC all 1s case; the second line is for the DC all 0s case; the third line is for the AC 1010s case; and the last line is for the AC 0101s case.

**http://www.stat.fsu.edu/pub/diehard/

temperature extremes). It's possible to infer from the data that for a block size of 63, application of IBS with simple 3× repetition coding reduces the errors to correct from 23 bits (35.9%) to 6 bits (9.4%) for provisioning at 1.2 V ± 10%, 20°C, generation at four corners. Over 2 million autocorrelation runs were used to compile the right curve, indicating that the probability of seven or more bit errors is less than 0.5 ppm. This means that a BCH (63, 30, $t = 6$) corrects errors across

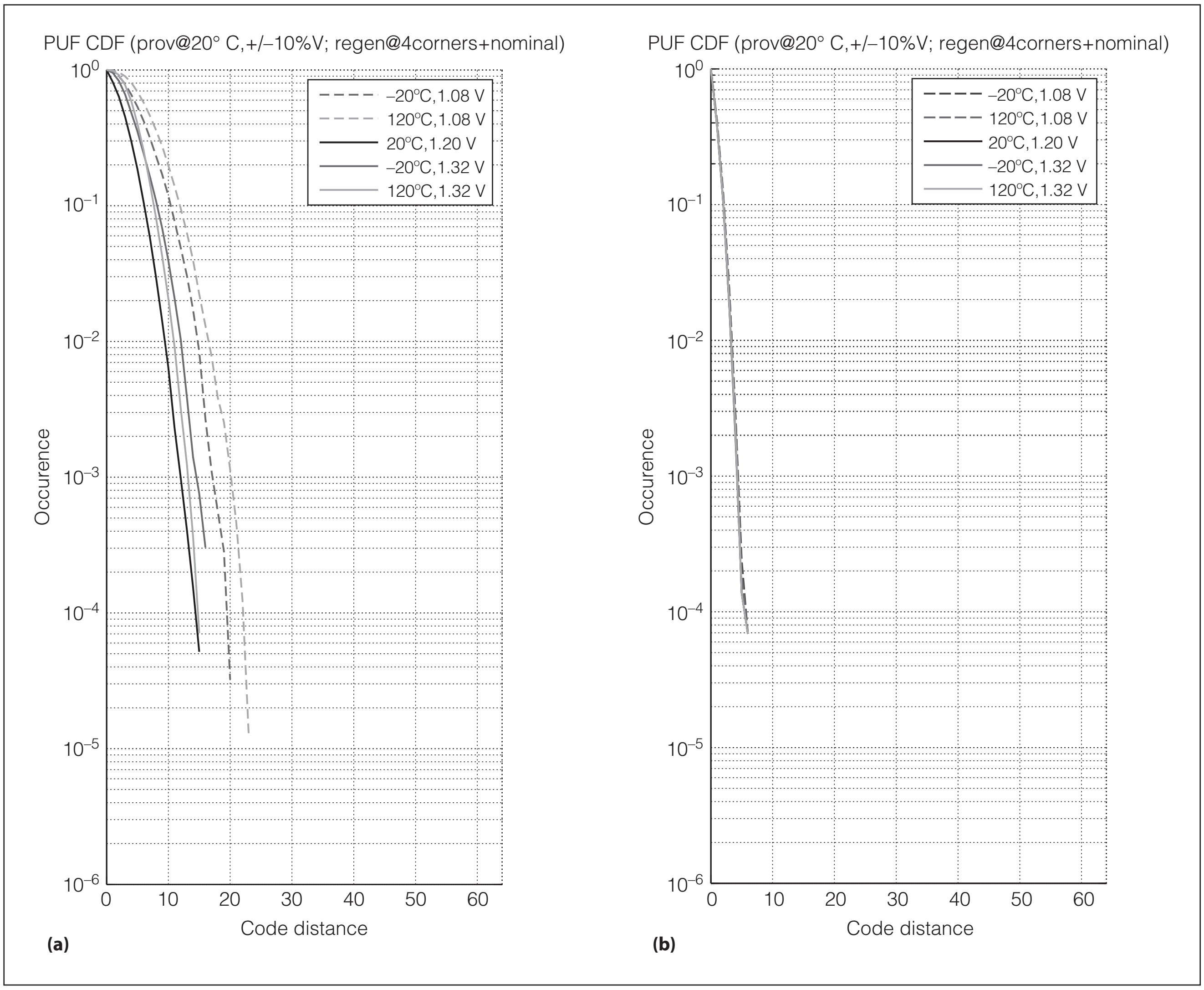


Figure 6. Coding gain of IBS. Number of errors before index-based syndrome decoding is 23 bits for a block size of 63 (a), and after applying IBS, the number of bit errors is reduced to 6 (b). This results in a significant reduction in total error correcting code logic complexity, and greatly improves the stability of PUF-derived secret bits across wide environmental conditions.

these environmental conditions at an error rate less than 0.5 ppm.

We performed the analysis described here before we completely built the IBS hardware in FPGA as a part of the design derivation and refining process; consequently, much of the design was emulated in software but using PUF information derived across 12 Xilinx Virtex-4 LX25 FPGAs, each with a single PUF. An all-hardware design was then derived, after this analysis, to confirm the results of data obtained from partial software emulation. This (early) all-hardware design contained an IBS enmapper/demapper with fixed 3-bit indices and 3× repetition coding, a BCH (63) decoder hardwired to $t = 6$ bits of error correction, and a PUF circuit.

This design was tested in extreme temperature conditions and never failed over tens of millions of error correction blocks. We performed more extreme temperature and voltage stressing tests using an even more mature hardware design on Virtex-5 FPGAs.

This scheme, by using IBS with simple 3× repetition coding and 3× majority decoding, reduces the BCH complexity requirement. Instead of using a BCH (255) code, for example (as Suh suggested[3]), we used a BCH (63) code, which has a 16× reduction in complexity. (BCH decoder complexity grows

Table 2. IBS test results for Xilinx Virtex-5.

Bit errors	I(3)* P(25°C, 1.0 V)** R(25°C, 1.0 V)*** (Nominal)	I(3) P(25°C, 1.0 V) R(–55°C, 1.0 V)	I(3) P(25°C, 1.0 V) R(–55°C, 1.1 V) (Fast, Fast)	I(3) P(25°C, 1.0 V) R(125°C, 0.9 V) (Slow, Slow)	I(3R3) P(25°C, 1.0 V) R(25°C, 1.0 V) (Nominal)	I(3R3) P(25°C, 1.0 V) R(–55°C, 1.0 V)
0	54.0%	59.5%	47.0%	55.5%	93.3%	94.8%
1	37.0%	23.9%	40.0%	35.6%	6.7%	5.2%
2	8.9%	16.6%	13.0%	5.3%	<= 0.27 ppm	<= 172 ppm
3	133 ppm	<= 6.24 ppm	4.8 ppm	3.6%		
4	<= 33.3 ppm		<= 0.6 ppm	<= 21.2 ppm		
5						
6						
Total samples	30,015	160,185	1,650,060	47,160	3,668,205	5,820
Block failures with BCH (63, 30, $t = 6$)	None ($\ll$ 33.3 ppm)	None ($\ll$ 6.24 ppm)	None ($\ll$ 0.6 ppm)	None ($\ll$ 21.2 ppm)	None ($\ll$ 0.27 ppm)	None ($\ll$ 172 ppm)

***I**(3) = 3-bit **I**ndex ; I(3**R**3) adds 3 × **R**epetition Coding.
****P**(.) = **P**rovisioning conditions
*****R**(.) = **R**egeneration conditions

approximately with the square of the block size.) However, as Suh's research showed,[3] the code does not correct across as extreme temperature conditions, and corrects across smaller voltage variations. If a BCH (511) code is required to produce an equivalent environmental robustness, there's a 64× reduction in complexity using IBS with simple repetition coding.

Results on Xilinx Virtex-5

IBS also reduces error correction requirements in designs implemented for Virtex-5 FPGAs. The design used was more mature than the one used with Virtex-4. Specifically, the Virtex-5 version had a higher degree of programmability and more sophisticated debug and data-gathering facilities.

Table 2 shows representative results obtained from one Virtex-5 device for a 63-bit block size. We performed provisioning at 25°C, at a nominal core voltage of 1.0 V. When regeneration was done at the fast corner (–55°C, 1.1 V), three bit errors occurred at 4.8 ppm, and four bit errors were not observed for more than 1.65 million blocks, indicating that the probability of four bit errors was less than 0.6 ppm. An ECC to correct three bit errors out of 63, based on this data, has a block failure rate less than or equal to 0.6 ppm. Alternatively, retry mechanisms can be used, or errors can be forgiven, depending on the application. Regeneration at the slow corner (125°C, 0.90 V) showed comparable results, but only 47,160 sample blocks were taken, so our results lacked resolution. We can safely say that if a $t = 6$ corrector is used, block error rates are likely to be well below 1-ppm levels. Note that the result of applying 3 × repetition coding with IBS is even more dramatic, with a maximum of a 1-bit error in the data set (see the two rightmost columns in Table 2).

IBS HAS TWO major advantages. First, the technique can be shown formally (and affirmed by NIST test results) to be information-theoretically secure in that the syndrome does not leak additional min-entropy on the hidden secret bits. Second, the index-based coding technique, by its very nature, is robust. Error correction block failure rates can be easily driven below 1 ppm, which was empirically demonstrated in Xilinx Virtex-4 and Virtex-5 FPGAs. Future work includes characterizing the stability of PUF-derived secret bits across a wider range of environmental parametrics, for example, to account for aging and radiation effects, as well as adapting IBS to a wider range of PUF and biometric sources. ■

Acknowledgment

We thank Marten van Dijk of the Massachusetts Inst. of Tech. for his valuable input to the draft, especially for his contribution to the proofs in the "Mathematical Model for IBS Enmapper and Proofs" sidebar.

■ References

1. B. Gassend, "Physical Random Functions," master's thesis, Department of EECS, Massachusetts Inst. of Tech., 2003.
2. B. Gassend et al., "Silicon Physical Random Functions," *Proc. ACM Computer Communication Security Conf.,* ACM Press, 2002, pp. 148-160.
3. G.E. Suh, "AEGIS: A Single-Chip Secure Processor," PhD thesis, Electrical Eng. and Computer Science Dept., Massachusetts Inst. of Tech., 2005.
4. M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight Secure PUFs," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 08), IEEE CS Press, 2008, pp. 670-673.
5. M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing Techniques for Hardware Security," *Proc. Int'l Test Conf.* (ITC 08), IEEE CS Press, 2008, pp. 1-10.
6. D. Lim, "Extracting Secret Keys from Integrated Circuits," master's thesis, Department of EECS, Massachusetts Inst. of Tech., 2004.
7. Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *Proc. Eurocrypt 2004,* LNCS 3027, Springer, 2004, pp. 523-540.
8. R. Maes, P. Tuyls, and I. Verbauwhede, "Intrinsic PUFs from Flip-flops on Reconfigurable Devices," *Proc. Benelux Workshop Information and System Security* (WISSec 08), 2008.
9. D.E. Holcomb, W.P. Burleson, and K. Fu, "Initial SRAM State as a Fingerprint and Source of True Random Numbers for RFID Tags," *Proc. Conf. Radio Frequency Identification Security* (RFID 07), 2007; http://rfidsec07.etsit.uma.es/slides/slides.htm.
10. R. Maes, P. Tuyls, and I. Verbauwhede, "A Soft Decision Helper Data Algorithm for SRAM PUFs," *IEEE Int'l Symp. Information Theory* (ISIT 09), IEEE Press, 2009.

Meng-Day (Mandel) Yu is a senior design engineer at Verayo, and is a technical lead in the development of PUF-based IP for ASICs and FPGAs. His research interests include coding theory, signal processing, and computer security. He has an MS in electrical engineering from Stanford University.

Srinivas Devadas is a professor and associate head of the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology. His research interests include CAD, computer architecture, and computer security. He has a PhD in electrical engineering from the University of California, Berkeley. He is a Fellow of the IEEE.

■ Direct questions and comments about this article to Meng-Day (Mandel) Yu, 1054 S. De Anza Blvd., Ste. 201, San Jose, CA 95129; myu@verayo.com, and to Srinivas Devadas, Rm. 32-G844, Massachusetts Institute of Technology, Cambridge, MA 02139; devadas@mit.edu.

IEEE computer society

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.
MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.
COMPUTER SOCIETY WEB SITE: www.computer.org
OMBUDSMAN: To check membership status or report a change of address, call the IEEE Member Services toll-free number, +1 800 678 4333 (US) or +1 732 981 0060 (international). Direct all other Computer Society-related questions—magazine delivery or unresolved complaints—to help@computer.org.
CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.
AVAILABLE INFORMATION: To obtain more information on any of the following, contact Customer Service at +1 714 821 8380 or +1 800 272 6657:

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES
Computer: The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.
Periodicals: The society publishes 13 magazines, 18 transactions, and one letters. Refer to membership application or request information as noted above.
Conference Proceedings & Books: Conference Publishing Services publishes more than 175 titles every year. CS Press publishes books in partnership with John Wiley & Sons.
Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.
Technical Committees: TCs provide professional interaction in more than 45 technical areas and directly influence computer engineering conferences and publications.
Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.
Certifications: The society offers two software developer credentials.
For more information, visit www.computer.org/certification.

IEEE

revised 30 Nov. 2009

COMPUTER SOCIETY OFFICES
Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036
Phone: +1 202 371 0101 • **Fax:** +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314
Phone: +1 714 821 8380
Email: help@computer.org
Membership & Publication Orders:
Phone: +1 800 272 6657 • **Fax:** +1 714 821 4641
Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • **Fax:** +81 3 3408 3553
Email: tokyo.ofc@computer.org

Next Board Meeting:
5 Feb. 2010, Anaheim, CA, USA