Security and Reliability Properties of Syndrome Coding Techniques Used in PUF Key Generation

Meng-Day (Mandel) Yu
David M'Raïhi
{myu, david}@verayo.com
Verayo, Inc.
San Jose, CA, USA

Srinivas Devadas
devadas@mit.edu
MIT
Cambridge, MA, USA

Ingrid Verbauwhede
ingrid.verbauwhede@esat.kuleuven.be
ESAT/SCD-COSIC, KU Leuven
Leuven, Belgium

Abstract: *A Physical Unclonable Function (PUF) uniquely identifies identically manufactured silicon devices. To derive keys, a stability algorithm is required. Unlike conventional error correction used in communication systems, a PUF stability algorithm has a dual mandate of accounting for environmental noise while minimally disclosing keying material; the latter, security, aspect is generally not a concern for conventional error correction use cases. For the purpose of comparison, we classify PUF stability algorithms into three Syndrome coding methods: Code-Offset; Index-Based Syndrome; Pattern Vector. We analyze and compare these methods with a focus on security and reliability properties, including a comparison of relevant security assumptions as well as a comparison of relevant ASIC PUF reliability data.*

Keywords: Physical Unclonable Function (PUF); Key Generation; Syndrome Coding; Security Properties

Introduction

Over the past decade, the concept of net-enabled operations has become a cornerstone for our national-defense posture. The underlying assumption of this vision is the availability of robust, reliable, secure information and communications infrastructures. Silicon-based Physical Unclonable Functions (PUFs) serve as a critical design primitive to secure these microelectronics systems [3]. The idea is to use unclonable manufacturing variations that not even a device manufacturer can control or reproduce (within tolerances of lithography and fabrication equipment) to authenticate a microelectronic device based on its "birth characteristics," similar to how biometrics can be used to identify a human being. This silicon manufacturing variation, similar to human biometric (e.g., fingerprint, iris scan), has readings that are *noisy* when digitized and represented as 1s and 0s. No two repeated readings are guaranteed to be the same bit-for-bit, but they are similar enough such that identity authentication can be achieved to a sufficient degree of confidence. These identifying silicon signatures take advantage of atomic and sub-atomic level manufacturing variations that, however, vary with a change in environmental condition (e.g., voltage, temperature, aging) between a *provisioning* condition and a *regeneration* condition. To use manufacturing variation material to derive cryptographic keys, a *stability algorithm* is required. These stability algorithms have a form of error correction or error processing capability, and require *Syndrome* (aka Helper Data) to return a regenerated response in a bit-exact fashion to a response snapshot taken during provisioning. To date, to our best knowledge, there have been three primary methods of Syndrome coding: Code-Offset [2]; Index-Based Syndrome [14]; and Pattern Vector [8]. For the first time in open literature, we analyze, compare and contrast these methods, and include an analysis of security properties and as well as summarizing reliability data from PUF ASIC implementations.

The remainder of the paper is organized as follows. We first give an overview of Physical Unclonable Functions (PUFs) and PUF Key Generation. The subsequent sections focus on the stability algorithm and discuss the three primary methods as identified above for Syndrome coding. We then discuss security properties. Finally, we conclude with a survey of PUF Key Generation reliability data from PUF ASIC implementations.

Introduction to Physical Unclonable Functions

The use of Physical Unclonable Functions (PUFs) as a silicon-unique root of trust was first published by researchers at MIT [3], enabling authentication based on chip-unique responses as well as generation of chip-unique cryptographic keys. Multiple silicon-based PUF circuits have since been realized. An *arbiter-based PUF* was prototyped in an ASIC [5]. A *ring-oscillator PUF* was built and tested in [12]. Use of initial SRAM state as a PUF was explored and tested in [13]. These are the main first representatives of the three major forms of PUFs that are commonly discussed today in open literature.

Introduction to PUF Key Generation

PUF Key Generation comprises of two steps: Provisioning and Regeneration. During the Provisioning process (Figure 1), a response is generated from a PUF, and a Syndrome is generated then stored for later use. The Syndrome can be stored in a public fashion but is generally assumed to be integrity protected. During a Regeneration process (Figure 2), the PUF is queried and a noisy response is regenerated; it is noisy due to environmental differences between the Regeneration and Provisioning conditions, e.g., as variations in temperature, voltage, and/or aging affect the way the response is derived. The Syndrome is read in. A Stability Algorithm (e.g., an error correction decoder) reads

in both the Syndrome information as well as the noisy response, to re-derive in a bit-exact fashion the stable PUF bits. The stable PUF bits are further processed or used by downstream cryptographic functions such as key derivation functions, hash functions, block ciphers such as AES, public-key based algorithms such as RSA, etc.

While there may be other forms of Syndrome "Helper Data" information generated during Provisioning and used during Regeneration, such as a random value used for universal hash functions, we focus our attention on the portion that is strictly used to reconcile the PUF noise between the two queries in a manner such that the same Stable PUF Bits (B) can be generated over and over again. We will provide in the following sections an analysis and comparison of security properties as well as reliability properties. The simplified diagrams in Figures 1 and 2 reflect this focus.

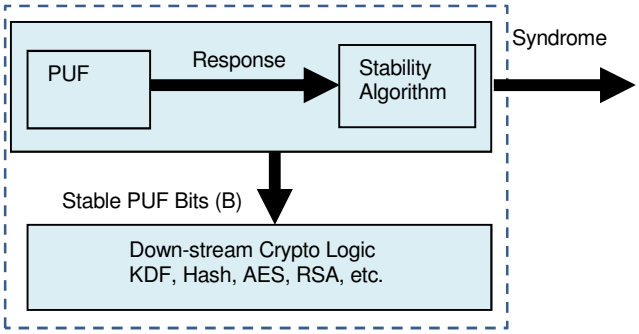

Figure 1: Provisioning

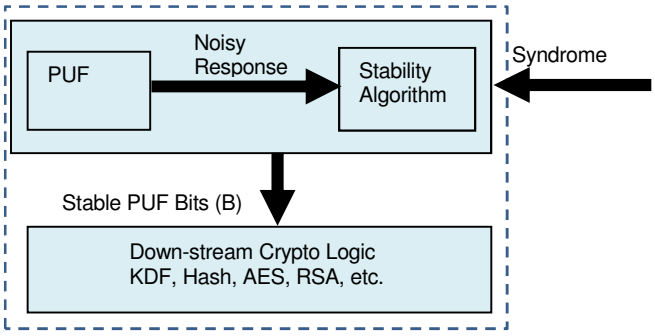

Figure 2: Regeneration

Classes of Stability Algorithms

Now we discuss classes of stability algorithms, i.e., Syndrome coding techniques, to account for PUF noise, so as to allow a PUF to regenerate the same, bit-exact stable bits despite environmental changes. For the purpose of comparisons, we categorize them into three primary ones: Code-Offset [2]; Index-Based Syndrome [14]; and Pattern Vector [8]. The publications [2], [14], [8], respectively, are the first representatives of each of the three main methods, while noting that there are derivative methods under this categorization framework, e.g., layering Soft Decision information on top of Code-Offset [6] or oscillator ordering encoding on top of Code-Offset [7], or using Index-Based Syndrome in a redundant and "complementary" manner to improve soft decision decoding properties [4].

We note that in a conventional modem connection, for every data byte transmitted, a parity bit may be attached for error detection. The parity bit reduces amount of uncertainty in the data byte to no more than 7 bits. While in a conventional communication system, such information leakage is generally not a concern, this is a central concern for PUF Key Generation. Thus, instead of applying conventional error correction directly to account for PUF noise, a Syndrome coding scheme is applied. An effective algorithm needs to simultaneously address stability as well as security concerns. In next section, we describe security properties of the three Syndrome coding schemes, and in section following reliability properties.

Security Properties

Code-Offset. In Code-Offset, the Syndrome bits generated correspond to the XOR mask for a sequence of PUF output bits required to form a valid error correction codeword. The entropy loss due to the Syndrome bits corresponds to the maximum number of Syndrome bits λ exposed [2]. As such, the number of Syndrome bits should be kept small; no entropy remains under this framework if the number of Syndrome bits exceeds the min-entropy of the PUF output bits used. Let us assume a linear code example, with an $\{n, k, t\}$ error correction code of block size n, data bits of length k, and error correction capability of t bits. The PUF generates n bits, and after syndrome encoding, $n - k$ bits of Syndrome is outputted. Here, entropy loss $\lambda = n - k$. Let n' designate the minimum entropy (min-entropy) of the n-bit PUF output. With a λ bit loss, the residual entropy (secrecy remaining from the starting k bits) is $n' - \lambda$. Under this security framework that is information-theoretic, there is no secrecy left if $n' <= \lambda$. Therefore, the amount of secrecy left depends on the min-entropy of the PUF once the error correction coding scheme and associated parameters are chosen. We note that even under the assumption that PUF outputs are independent and identically distributed (i.i.d.), if PUF output bits are biased, PUF min-entropy can be reduced to the point where there is no secrecy remaining in the system if certain error correction codes are used. As an extreme example, if a PUF has a 0.5152 bias (out of every 10000 bits there are 5152 1s on the average and the rest 0s), if a 33x repetition code is used, every single bit is leaked out, statistically speaking [14]. Alternatively, in a less extreme case, if a 9x repetition code is used, 1 bit is leaked out of every 5.6 bits encoded [14]. This is because if a single random bit is generated (from an RNG, or a PUF), and that bit is repeated 33 times (or 9 times) and bit-wise XORed with an equal number of PUF output bits, the PUF bias leak gets amplified in the resulting XOR Syndrome mask. For a repetition of r, this corresponds to a $\{n=r, k = 1, t\}$ code. The residual entropy is $r' - (r - 1)$, which can go negative (no secrecy left) for $r' <= r - 1$, and is highly sensitive to PUF bias. We note that the Code-Offset

implementation in [7] properly accounts for the entropy loss based on the framework in [2] while most prior works such as [1], which focused on efficient implementation as opposed to security, did not explicitly account for the entropy loss.

Index-Based Syndrome. In the Index-Based Syndrome scheme, the Syndrome output bits are divided into Syndrome Words; each is an index lookup into a sequence of PUF output bits [14]. In the canonical example, a "1" bit or a "0" bit randomly generated (not necessarily from a PUF) can be encoded as a "max" or a "min" value in the distribution of a stream of PUF output bits that contain "soft-decision" or confidence information. The scheme is not subject to the requirement that there is no secrecy remaining if $n' <= \lambda$ under the assumption that PUF output bits can be regarded as independent and identically distributed (i.i.d.), which is a common assumption. Under such an assumption, even if the PUF responses are heavily biased, the Syndrome does not impose additional min-entropy leakage on the secret bits, which is not the case for Code-Offset. Additionally, there is an inherent coding gain in the scheme, reducing the downstream error correction requirements, and reducing / eliminating repetition coding which can be, as demonstrated previously, problematic from a security standpoint. Under a PUF i.i.d. assumption, Index-Based Syndrome can be proven to be information-theoretically secure in that the Syndrome bits do not induce additional average min-entropy leakage. Since i.i.d. is difficult to verify in practice, [15] attempted to equate independence with unlearnability, e.g., making sure there are more unknowns in the manufacturing variation parameters than equations leaked via syndrome. Index-Based Syndrome has been analyzed for Oscillator PUFs [16] as well as analyzed with Memory PUFs [4]. Its main advantages lie in its high coding gain (reduced error correction complexity and reduced need for leaky repetition coding), as well as a decoupling PUF output bias characteristics (for an i.i.d. PUF) from security considerations for a more modular design methodology.

Pattern Vector. In the Pattern Vector approach, a sequence of randomly generated "1" or "0" bits is recovered using a PUF response Pattern Vector corresponding to a challenge offset that is not readily known to an adversary. Assuming a PUF with a challenge seed that is used to derive 1024 subsequent challenges, generating 1024+512 bits, let us encode a single 10-bit secret value. We use a response size of 512 bits; there are 2^{10} challenge offsets which can be used to encode the 10-bit secret. The security of this scheme relies on machine learning assumptions. From [9], one can infer that an 8-XOR 128-stage PUF is out of reach for present day machine learning attacks from a CPU time standpoint and requires >> 0.5 Million Challenge and Response Pairs (CRPs). To generate a single key, only a limited number of response bits (<<0.5M) are used, and the attacker does not know the challenges corresponding to these response bits. The advantage of this scheme is that no complex error correction is required. Instead of relying on a PUF i.i.d. assumption (as in Index-Based Syndrome), a machine learning assumption based on a *limited response set* and *uncertain challenge offset* is used. The uncertain challenge offset is used to derive the key; the adversary does not have precise knowledge of the challenge that maps to each response bit, with up to 10 bits uncertainty per string of 512 bits in the previous example.

Comparisons. In the table below, we list different assumptions we make about a PUF to derive security for each scheme. Code-Offset is secure if there is sufficient min-entropy in the PUF response bits; we want security parameter >= $n' - \lambda$ (e.g., security parameter = 128 for a 128-bit key). Index-Based Syndrome is secure if PUF output bits can be regarded as independent, for example, using disjoint PUF components. In Index-Based Syndrome, having first order bias alone does not affect security (unlike Code-Offset even in an i.i.d. setting). Syndrome size does not matter from a security proof standpoint, but requires response independence. For Pattern Vector, the security framework is based on machine learning assumptions given limited amount of information available for the adversary, e.g., limited number of response bits and uncertainty in challenge offset; as an example, we want bits to learn the system >> syndrome size, the latter is generally linear and >> security parameter.

TABLE I. PUF SECURITY ASSUMPTIONS

	Assumption	***Example***
Code Offset	Sufficient PUF min-entropy	security parameter >= n' – syndrome size
Index-Based Syndrome	i.i.d. PUF response bits	security parameter >> independent manu. var. unknowns (regardless of syndrome size)
Pattern Vector	PUF resistant to machine learning attacks (given limited response bits, challenge offset uncertainty)	bits needed to learn sys >> syndrome size (syndrome size >> security parameter)

Reliability Properties

Table II contains a summary of published results on PUF reliability associated with PUF Key Generation. The Code-Offset representative available in open literature that tested the most extreme environmental conditions corresponds to [10]. Under the assumption that an error correction scheme can correct up to a quarter of the PUF bits being noisy (flipped) as reported in [10], there is a stability safety margin of 24% in that 24% of the error correction capability remain unused under the Temperature and the Voltage ranges shown in the first row of Table II. The test comprised of 68 PUF devices (4 PUFs on each of 17 distinct devices). The Index-Based Syndrome representative is [16], which contains V/T corner testings, and has a stability safety margin in excess of 50%,

attributed to the high coding gains inherent in Index-Based Syndrome, prior to any conventional error correction techniques being applied. The test set comprised of 133 PUF devices (7 PUFs on each of 19 distinct devices). Since aging results were not presented in this particular publication, accelerated aging results were obtained from [15] which also used Index-Based Syndrome. For the Pattern Vector representative [8], only temperature testing was performed and with better than parts per billion (ppb) reliability. The scheme works reliably so long as normal PUF authentication, based on a Hamming threshold value, works reliably. Test set comprised of 10 PUF devices (1 PUF on each of 10 distinct devices).

TABLE II. PUF KEY GENERATION ENVIRONMENTAL TESTING AND RELIABILITY

	Temperature	***Volt-age***	***V/T Corners***	***Aging***	***Reliability***
Code Offset	-40ºC to 80ºC	±10%	n/a	4.7 equivalent years @ 40ºC	24% Stability Margin
Index-Based Synd.	-55ºC to 125ºC	±20%	4-corners	20 equivalent years @ 55ºC	50%+ Stability Margin
Pattern Vector	-25ºC to 85ºC	n/a	n/a	n/a	<< 1 ppb

Conclusions

Since the first PUF Key Generation architecture was proposed in the context of a secure processor [11], several PUF Key Generation implementations have been realized and several Syndrome coding schemes developed to address the dual mandate of security and reliability. We have categorized the approaches and summarized the security assumptions. The reliability results published to date show that each of the approaches is capable of achieving fairly high level of reliability under a relatively wide range of conditions. Future work includes further elaboration of security assumptions and threat models, and derivation of possibly improved Syndrome coding schemes.

References

1. C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, P. Tuyls, "Efficient Helper Data Key Extractor on FPGAs," *Workshop on Cryptographic Hardware and Embedded Systems (CHES),* 2008*,* LNCS vol. 5154, pp. 181-197.
2. Y. Dodis, L. Reyzin, A. Smith, ''Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data,'' *Eurocrypt*, 2004.
3. B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Silicon Physcal Random Functions," *ACM Computer and Communication Security (CCS) Conference*, 2002.
4. M. Hiller, D. Merli, F. Stumpf, "Complementary IBS: Application Specific Error Correction for PUFs," *IEEE Int'l Symposium on Hardware-Oriented Security and Trust (HOST)*, 2012.
5. D. Lim, ''Extracting Secret Keys from Integrated Circuits,'' Master's thesis, EECS, MIT, 2004.
6. R. Maes, P. Tuyls, I. Verbauwhede, "A Soft Decision Helper Data Algorithm for SRAM PUFs," *IEEE Int'l Symposium on Information Theory (ISIT)*, 2009.
7. R Maes, A. Herrewege, I. Verbauwhede, "PUFKY: A Fully Functional PUF-based Cryptographic Key Generator," *Workshop on Cryptographic Hardware and Embedded Systems (CHES),* 2012*,* LNCS vol. 7428, pp. 302-319.
8. Z. Paral, S. Devadas, "Reliable and Efficient PUF-based Key Generation Using Pattern Matching," *IEEE Int'l Symposium on Hardware-Oriented Security and Trust (HOST),* 2011.
9. U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber, "Modeling Attacks on Physical Unclonable Functions," *ACM Conference on Computer and Communications Security (CCS)*, 2010.
10. G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G-J. Schrijen, M. van Hulst, P. Tuyls, "Evaluation of 90nm 6T-SRAM as Physical Unclonable Function for Secure Key Generation in Wireless Sensor Nodes," *IEEE Int'l Symposium on Circuits and Systems (ISCAS)* 2011.
11. G. Suh, "AEGIS: A Single-Chip Secure Processor," Ph.D. dissertation, MIT, 2005.
12. G. Suh, S. Devadas, "Physical Unclonable Functions for Device Euthentication and Secret Key Generation," *Design Automation Conference (DAC),* 2007.
13. Y. Su, J. Holleman, B. Otis, "A 1.6pJ/bit 96% Stable Chip ID Generating Circuit Using Process Variations," *IEEE Int'l Solid-State Circuits Conf. (ISSCC)*, 2007.
14. M. Yu, S. Devadas, "Secure and Robust Error Correction for Physical Unclonable Functions," *IEEE Design and Test of Computers*, Special Issue on Verifying Physical Trustworthiness of ICs and Systems, vol. 27, no. 1, pp. 48-65, Jan./Feb. 2010.
15. M. Yu, D. M'Raïhi, R. Sowell, S. Devadas, "Lightweight and Secure PUF Key Storage Using Limits of Machine Learning," *Workshop on Cryptographic Hardware and Embedded Systems (CHES),* 2011*,* LNCS vol. 6917, pp. 358-373.
16. M. Yu, R. Sowell, A. Singh, D. M'Raïhi, S. Devadas, "Performance Metrics and Empirical Results of a PUF Cryptographic Key Generation ASIC," *IEEE Int'l Symposium on Hardware-Oriented Security and Trust (HOST),* 2012.