J Electron Test
DOI 10.1007/s10836-014-5471-7

Testing Methods for PUF-Based Secure Key Storage Circuits

Mafalda Cortez · Gijs Roelofs · Said Hamdioui · Giorgio Di Natale

Received: 30 April 2014 / Accepted: 21 August 2014

Abstract Design for test is an integral part of any VLSI chip. However, for secure systems extra precautions have to be taken to prevent that the test circuitry could reveal secret information. This paper addresses secure test for Physical Unclonable Function based systems. It investigates two secure Built-In Self-Test (BIST) solutions for Fuzzy Extractor (FE) which is the main component of PUF-based systems. The schemes target high stuck-at-fault (SAF) coverage by performing scan-chain free functional testing, to prevent scan-chain abuse for attacks. The first scheme reuses existing FE blocks (for pattern generation and compression) to minimize the area overhead, while the second scheme tests all the FE blocks simultaneously to minimize the test time. The schemes are integrated in FE design and simulated; the results show that for the first test scheme, a SAF fault coverage of 95 % can be realized with no more than 47.1k clock cycles at the cost of a negligible area overhead of only 2.2 %; while for the second test scheme a SAF fault coverage of 95 % can be realized with 3.5k clock cycles at the cost of 18.6 % area overhead. Higher fault coverages are possible to realize at extra cost (i.e., either by extending the test time, or by adding extra hardware, or a combination of both).

Responsible Editor: M. Tehranipoor

M. Cortez (✉) · G. Roelofs · S. Hamdioui
Faculty of EE, Mathematics and CS, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: mafalda.m.cortez@gmail.com

G. Di Natale
Giorgio Di Natale LIRMM, Université Montpellier II, 161 Rue Ada, 34392 Montpellier Cedex 5, France
e-mail: DiNatale@lirmm.fr

Keywords PUF-based systems · Fuzzy Extractor · Secure testing · Scan-chain free testing

1 Introduction

Physical Unclonable Functions (PUFs) based systems are becoming popular solutions for secure key storage against physical attacks [12, 13, 25]; they use the unique, random, uncontrollable and intrinsic physical properties of *Integrated Circuits* (ICs) to derive a cryptographic key. The robustness of a such system is evaluated by means of its *reproducibility* (i.e., ability of the system to recover the cryptographic key from the same IC) and its *uniqueness* (i.e., the ability of the system to generate a unique cryptographic key for each IC) [12, 25]. A *Fuzzy Extractor* (FE) is one of the main components of a PUF-based system; its responsibility is to assure the system's reproducibility and uniqueness [10, 21]. Hence, FE flawless operation is critical for the robustness of PUF-based systems. Testing a PUF-based system, and FE in particular, is a challenge. Testability demands excellent accessibility and observability, while security demands poor/no accessibility and observability to the chip, especially during the operation mode where an attacker could easily retrieve partial or complete cryptographic key. The trade-off between testability and security is a major challenge.

Design-for-Test and testability of secure devices have recently gained a lot of attention [8, 9, 11, 15, 19, 29]. Overall, the published schemes can be classified into two classes: enhanced scan-chains [6, 8, 15, 19, 29] and functional based *Built-In Self Test* (BIST) [9, 11].

Enhanced scan-chains target the protection of chains from being misused by attackers. In [29], B. Yang et al. developed a test solution for crypto cores based on a type of register that cannot be scanned out during test mode until being reset. In [8], A. Das et al. developed a test wrapper for secure test that authenticates legitimate testers. In [15], D. Hely et al. introduced spy flip-flops in the scan-chain that detect malicious shifts. In [19], J. Lee et al. applied a technique that makes the scan-chain operate unpredictably for untrusted users. In [6], J. Da Rolt et al. designed a smart test controller that automatically discerns scan shift operations and blocks any scan-out leakage. However, industry strongly believes that enhanced scan-chains cannot provide 100 % secure IC, as many researches have showed that scan-based DFTs can be hacked [2, 4, 7, 23]. Therefore, industry is reluctant to include them in designs targeting secure applications.

On the other hand, functional test based BIST targets the enhancement of security, although reaching a very high fault coverage with these schemes is a major challenge. In [11], M. Doulcier et al. presented a technique to reuse an *Advanced Encryption Standard* (AES) for self-testing; the work shows that AES cores have enough randomness to be used as test pattern generators and built on this property to seft-test the AES core in a loop fashion. In [9], Di Natale et al. proposed a generic self-test scheme for crypto cores; the work is an extension of the work presented in [11]. It performs the same analysis but for *Data Encryption Standard* (DES). However, both [11] and [9] are not suited for testing PUF-based systems for two main reasons. First, AES/DES crypto cores are not available in all PUF-based systems and second, PUF-based systems comprise, on top of the crypto cores, error correction blocks which make it more challenging to test functionally.

Although the research in hardware security including test is getting more attention due to the importance of the field, there is almost nothing published on testing PUF-based systems. This topic is addressed in this paper. In particular, this work targets testing of FEs, which are the main blocks of such systems; FEs are challenging to test as they comprise not only a crypto core, but also error correction blocks, being typically hard to test functionally.

This paper is an extension of our previous work presented in [5]; it proposes two efficient *scan-chains free* secure test schemes that realize high test quality based on pattern generation for stuck-at-faults using functional testing. The first proposed solution reuses FE existing blocks (for pattern generation and compression) to minimize the area overhead [5], while the second solution tests all comprising FE blocks simultaneously to minimize the test time. In addition, optimization techniques to even further reduce the test time and increase fault coverage are proposed. In addition to the main contribution of [5], i.e.,

- a low area overhead secure test method with its inherent concept, methodology, results and discussion,

this paper has the following contributions

- fast and secure test method with its inherent concept, methodology, results and discussion;
- in depth discussion of the results, including comparison between secure test methods, comparison with state-of-the-art, security analysis and list of recommendations on how to securely test FE;
- and classification of methods to improve test quality and implementation of one of these methods.

The rest of the paper is organized as follows. Section 2 briefly reviews the background on PUF based-systems and analyzes FE in detail. Section 3 defines the test requirements, proposes the two secure test methods and gives means to further improve the quality of proposed methods. Section 4 defines the experiments and presents the results. Section 5 discusses them, compares both test methods, analysis the methods security and provides a generic list of step-by-step instructions to securely test FE. Finally, Section 6 concludes the paper.

2 PUF-Based Secure Systems

We first briefly show how PUFs are deployed in a key storage system. Thereafter, we describe the Fuzzy Extractor in detail; it is the main block of such a system and the primary focus of this work.

2.1 Key-Storage based on PUFs

Figure 1 shows the flow of a PUF-based key-storage system [12, 25] implemented with a *Fuzzy Extractor* (FE) [10, 21], which typically consists of two phases:

(a) **Enrollment** : a cryptographic key is generated from a PUF. First, a PUF measurement is taken and used as *PUF Reference Response* (PRR). Next, PRR and *Random Seed* (provided externally) are processed by the FE into a cryptographically strong *Cryptographic Key*, and helper data is generated as an FE byproduct.

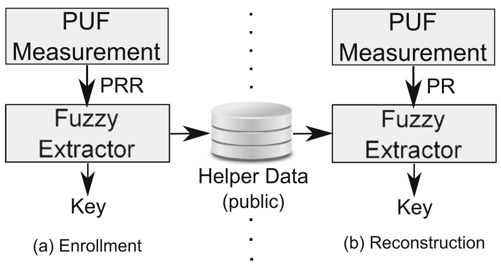


Fig. 1 PUF based Key Storage System

Finally, the helper data is stored in an external *Non-Volatile Memory* (NVM); hence, it becomes public information.

(b) **Reconstruction** : the earlier enrolled *Cryptographic Key* is reliably recovered. First, a PUF measurement is taken and used as *PUF Response* (PR). Typically, some bits of PR are different from the original PRR; hence, PR is a noisy version of PRR. Next, PR is processed by the FE in combination with the helper data which is retrieved from the external NVM. If the noisy PR is close enough to the PRR measured during enrollment (i.e., the PUF response is reproducible up to a limited amount of noise), then the FE succeeds to reliably reconstruct the enrolled *Cryptographic Key*.

2.2 Fuzzy Extractor

A Fuzzy Extractor (FE) is the fundamental component of a PUF-based key storage system; it has two main functions. (a) Error correction: it uses the helper data combined with error correction to correct errors in the measured PUF response; and (b) Privacy amplification: considering that the helper data contains information on the PRR, privacy amplification is needed to make sure that the helper data does not reveal any information on the derived cryptographic key; the FE compresses the resulting data into a cryptographic key with maximum entropy making it hard for the attacker to retrieve the key [10, 21]. It also removes any biasing (unequal distribution of zeros and ones) in the error-corrected PUF response. Privacy amplification is realized with Hash Function.

Figure 2 shows the six main blocks of a Fuzzy Extractor; this implementation is based on the one used for the UNIQUE project [27]. The Peripheral Circuitry has two main functions: (a) it selects between both functional modes (enrollment versus reconstruction), and (b) it performs XOR function either between the PUF response and the output of the Repetition Encoder (*RE_O*) to generate the Helper Data during the enrollment or, between the stored Helper Data and PUF response to generate the input of the Repetition Decoder (*RD_I*) during the reconstruction. The other five blocks are mostly computation intensive and are responsible for the enrollment and the reconstruction. Each of the five blocks is explained next.

1) *Golay Encoder:* first block of the enrollment phase. Its responsibility is to prepare the data for the error correction. This block maps the input *Random Seed* (12 bits per iteration $\times$ 86 iterations) to *GE_O* (24 bits per iteration) by appending twelve parity bits used for error correction. This makes it feasible for the Golay Decoder in Reconstruction phase to correct up to three bits [16, 24]. The main core of this block comprises a loop that generates the Golay space (space of perfect code words). Our implementation of the Golay Encoder has a latency of two clock cycles and it comprises 6.5 % of the total number of FE gates.

Repetition Encoder second block of the enrollment phase. It adds extra robustness to the error capabilities of the Golay Encoder. The block replicates each of the *GE_O* 24 bits 11 times resulting in 264 bits serial output *RE_O*; this enables the error correction up to five bits for the Repetition Decoder in the reconstruction phase. The enrollment phase completes after 86 iterations, i.e., the computations described above are performed 86 times. At each time, the 12 bits fraction of the 1032 bit *Random Seed* are processed with a 264 bits fraction of the PUF response to generate Helper Data. The total size of both PUF and Helper Data are each 2.8kB (264×86). The main part of the block comprises two counters. The first counter loops over the 24 bits of *GE_O*, while the second counter replicates each bit of *GE_O* 11 times. Our implementation of the Repetition Encoder has a latency of 267 clock cycles and it comprises 7.3 % of FE gates.

Repetition Decoder first block of the reconstruction phase and also the first stage of error correction. The

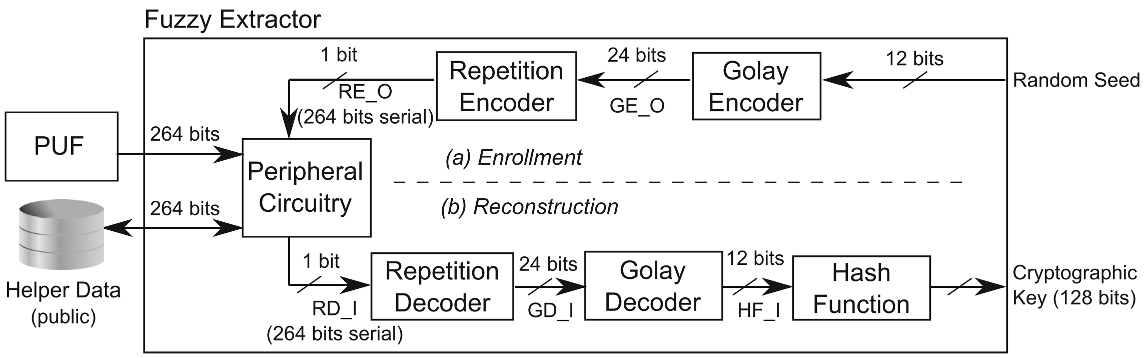


Fig. 2 Example of a Fuzzy Extractor

reconstruction phase starts with performing a new measurement of the PUF and XORing it with the Helper Data. The result of this operation is the serial input of the Repetition Decoder block. The block performs majority voting on each of the 24 groups (each of 11 bits) scanned serially via *RD_I*, and produce 24 bits at the output *GD_I*. This block performs the inverse operation of the Repetition Encoder and its main core comprises three counters: *one counter*, *repetition counter* and *destination counter*. The one counter counts the number of ones in a chunk of input *RD_I* (see Fig. 2) and its value is reset after the repetition counter processed n=11 input bits. Next, a single output bit is written on the index provided by the destination counter which is subsequently incremented. The written output bit presents the majority voting result of the processed input chunk derived from the one counter. Our implementation of the Repetition Decoder has a latency of 290 clock cycles and comprises 6.5 % of FE gates.

Golay Decoder second block of the reconstruction phase responsible for error correction. The block recovers *Random Seed*, i.e., *HF_I* (12 bits), as long as the provided input *GD_I* is within the error capabilities of the error correction system. Also during the reconstruction phase, the Repetition Decoder and the Golay Decoder repeat their operations 86 times; each time, they serially process 264 bits generated based on PUF and Helper Data. The results of each iteration is a 12 bits buffered inside the Hash Function block. The Golay Decoder is the most complex block of the FE; it contains a *Finite State-Machine* (FSM) with nine states for vector decoding. As stated previously, a Golay Decoder can correct up to three errors. Its input *GD_I* comprises 24 bits (12 message bits combined with 12 parity bits). Figure 3 shows the states dedicated to error correction; these are selected depending on the location and number of errors in *GD_I*. Error wise, five different cases are possible, denoted in Fig. 3 as case (i) till (v).

(i) *GD_I* is error-free; thus, the four states where the error correction takes place are skipped.
(ii) there are three or less errors in the message bits of *GD_I* and none in the parity bits.
(iii) there are one or two errors in the message bits of *GD_I* and exactly one in the parity bits.
(iv) there is exactly one error in the message bits of *GD_I* and two or less in the parity bits.
(v) there are no errors in the message bits of *GD_I* and three or less in the parity bits.

The Golay Decoder has a variable latency depending on its input, with a maximum of 10 clock cycles and it comprises 61.5 % of FE gates.

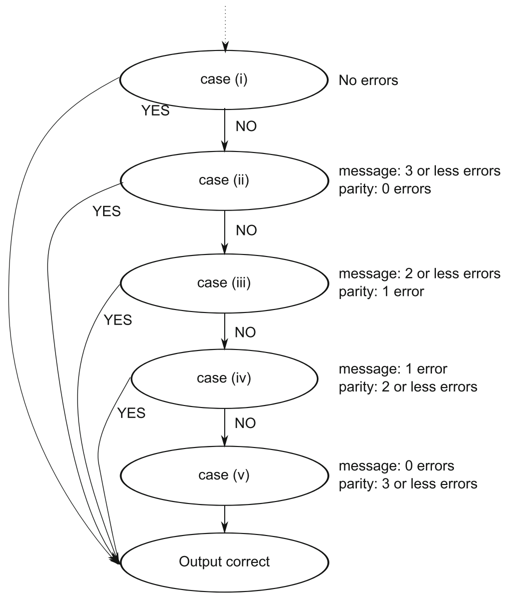


Fig. 3 Golay Decoder state-machine

2.2.1 Hash Function

last block of the reconstruction phase. It performs privacy amplification. This block concatenates the 1032 bits (12×86 iterations) received from the Golay Decoder and applies the hash function on it to calculate the 128 bit Cryptographic Key. Our Hash Function implementation comprises three main components: an input buffer, a *Linear Feedback Shift Register* (LFSR) and an accumulator register. First, the input *HF_I* is copied to the input buffer, which is then analyzed bit per bit. If the bit is one, the current LFSR output (which updates itself each cycle based on its polynomial function) is added (XORed) with the accumulator. However, if the bit is zero, the accumulator keeps its value. When all input bits are analyzed the value of the accumulator register is propagated to the output. The Hash Function has a latency of 32 clock cycles and it comprises 18.2 % of FE gates.

It is worth noting that the Fuzzy Extractor presented here is a *generic* construction of industrial implementations [27]; therefore, any test method developed for this circuit can be applied also to any other implementation.

3 Test Methods

First we define test and security requirements considered for the development of our test solutions. Then, we present our test methods and thereafter give means that can be used to further improve the quality of the proposed methods.

3.1 Test versus Security Requirements

Efficient test solutions for FE must prevent compromising the system security. The following requirements and assumptions apply:

(a) The signals of *PUF* measurement, *Random Seed* and *HF_I* (see Fig. 2) shall not be revealed at any time, partially nor fully. An attacker learning this information might derive the *Cryptographic Key*, breaking the systems security.
(b) Helper data is assumed to be public knowledge and does not have to be secured.
(c) Reverse engineering the Fuzzy Extractor is not an issue. The Fuzzy Extractor uses algorithms that are standard and publicly known.
(d) The PUF circuitry has its own internal test method, therefore it is outside the scope of this work.
(e) Minimum fault coverage of 95 %. Extended test times combined with methods to increase fault coverage are supposed to compensate for the remaining 5 %.

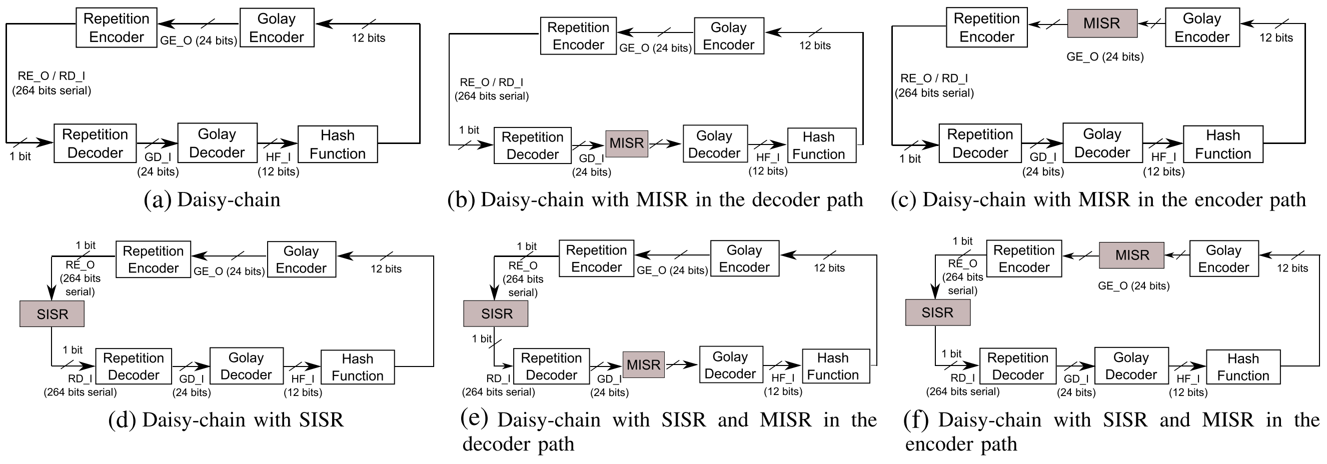
3.2 Secure Test Methods for FE

Next, we propose two secure test methods for FE: daisy-chain based and parallel test based methods. Both of them are scan-chain free, which is a security requirement. The two proposed methods will enable a good profiling of the maximum and minimum test time and area overhead. In an industrial application, a hybrid solution between these methods might be preferential.

3.2.1 Daisy-Chain Secure Test Method

We propose to (a) reuse the *Linear Feedback Shift Register* (LFSR) of the Hash Function block to create a random generator and, (b) test the FE in a loop-chain fashion, i.e., the outputs of each block are directly provided as inputs to next (connected) block as depicted in Fig. 4a. This approach results in a negligible area overhead. However, a high fault coverage for the Golay Decoder cannot be guaranteed. This is because the Golay Decoder receives error free input messages as provided by the Golay Encoder, which prevents the correct checking of all the decoder' states (see Fig. 3); e.g., in case the input vector of the Golay Decoder is error free as in case (i) of Fig. 3, the remaining four cases will be skipped. Hence, reusing LFSR of hash function with daisy-chain approach *alone* will not provide the required test quality for Golay Decoder. To solve this problem, the randomness of the patterns provided at the Golay Decoder inputs have to be improved in order to trigger all states of the Golay Decoder FSM. This can be done by inserting a *Multiple-Input-Shift-Register* (MISR) at the input of the Golay Decoder as shown in Fig. 4b. However, as the blocks are connected in a loop, the desired effect of randomness improvement can also be achieved by placing a MISR in any location between the Golay Encoder output and the Golay Decoder input (such as at the output of Golay Encoder in Fig. 4c), or a *Single-Input-Shift-Register* (SISR) if the location is just a serial line as it is the case in Fig. 4d). Moreover, a combination of MISR and SISR could be also used as shown in Fig. 4e and Fig. 4f. Comparing the area overhead and randomness of the several constructions presented in Fig. 4 reveals that:

1 Construction (d) results in the smallest area overhead.
2 Constructions (e) and (f) could lead to higher fault coverage, as the combination of using SISR and MISR could improve the randomness.
3 Constructions (b) and (c) as well as (e) and (f) are equivalent, reducing the number of constructions to four.



(a) Daisy-chain
(b) Daisy-chain with MISR in the decoder path
(c) Daisy-chain with MISR in the encoder path
(d) Daisy-chain with SISR
(e) Daisy-chain with SISR and MISR in the decoder path
(f) Daisy-chain with SISR and MISR in the encoder path

Fig. 4 Daisy-chain for a Fuzzy Extractor - different constructions

3.2.2 Parallel Secure Test Method

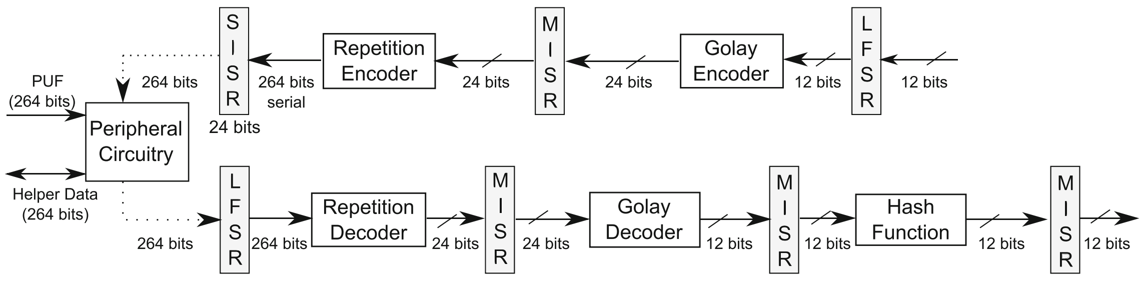
One way that can be used to reduce the overall test time and potentially increase the fault coverage of each FE block is to add a *Random Test Pattern Generator* (RTPG) at the input of each FE block. Increasing the randomness of the input patterns of each block will *potentially* increase the fault coverage as well. We will explore this approach while testing all FE blocks simultaneously (parallel test); the total test time is then the test time required by the largest block (in this case the Golay Decoder). In addition, a SISR/MISR is inserted at the output of each block for test data compression; see Fig. 5a. Starting with the first block, Golay Encoder, an LFSR is added at its input for random pattern generation and a MISR is added at its output for output test compression. The second block, Repetition Encoder, can reuse the MISR for pattern generation, reducing area overhead; a SISR is used for output compression as the output of the Repetition Encoder is a serial line. The same idea is applied for the remaining blocks. An LFSR is used as RTPG for the Repetition Decoder and a MISR is used to compress its test response. This MISR is reused as pattern generator for the Golay Decoder; a new MISR is added at the output of the Golay Decoder for output compression and reused for test pattern generation for the Hash Function. Finally, a last MISR is added at the output of the Hash Function for output compression.

Figure 5b shows the scheme implementation details for the Golay Encoder block; the remaining blocks have similar implementations. First, a MUX selects between the functional mode and the test mode by forwarding either the functional input (i.e., Random Seed) or the output of the RTPG (i.e., LFSR) to the input of block under test (i.e., Golay Encoder). In the functional mode, the output of the Golay Encoder is forwarded to the Repetition Encoder, while in the test mode the output of the Golay Encoder (test response) is compressed (MISR) and at the same time being sent as input test stimuli for the next block (Repetition Encoder). Once the test is concluded, the results are compared against a hardwired golden reference by means of XOR gates. The result of this comparison is a pass/fail signal. This approach results in a small test time, as the maximum test time is the test time of the most time consuming block to be tested. However, when compared with the daisy-chain approach, it has a larger area overhead.

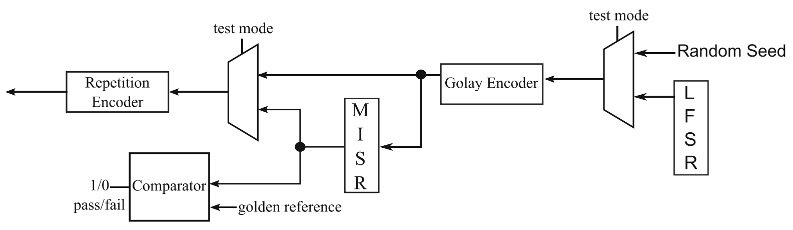
Considering a golden reference with defects, one of two cases may happen: either (a) a faulty device is not detected or (b) a good device is rejected. While both cases are costly, case (a) is more damaging but also very improbable. For a faulty device to pass the test, the faulty circuit would need to generate a MISR signature such that would match perfectly the also faulty golden reference. Moreover, the faulty MISR signature would serve as input to other blocks, which would cause the fault to be detected. However, if we want to increase the robustness of the golden references, some options are; e.g., comparing the test signatures against not one but two golden references (costly in area) or including a parity bit comparison of the test signature (cheaper as only 1 bit per golden reference is required).

3.3 Test Quality Improvement Using RTPG

Random-pattern-resistant faults might increase the test time; due to the specificity of the input test vector that detects such



(a) Block diagram of test mode



(b) Implementation details for Golay Encoder

Fig. 5 Parallel secure test method for a Fuzzy Extractor

faults, random test pattern generators might take a large number of clock cycles to generate the required test vector. In literature many methods targeting the reduction of the test time by means of reconfiguration of the random test pattern generator parameters can be found. We identify three main classes of reconfigurable parameters that affect random test pattern generation: output, state and structure. Moreover, these methods can be combined. They are explained next.

- **Output reconfigurability:** are methods where one or more original generated random test patterns being not able to detect any fault are mapped into other test vectors that detect one or more faults. Known examples of this method are *bit-flipping/fixing* [28][26] and *reconfigurable network* [22].
- **State reconfigurability:** are methods where one or more states (seed) of the random test pattern generator LFSR are changed to skip a sequence of random patterns that do not detect additional faults. Known examples of this method are *full reseeding*, *partial reseeding* [18] and *encoded reseeding* [3].
- **Structure reconfigurability:** are methods where one or more random test pattern generator feedback networks are dynamically reconfigured; hence, the generated random test pattern sequence is also changed. A known example of this method is *multi-polynomial* [14].
- **Combined reconfigurability:** to even optimize further the results, some of the previous methods can be combined together; e.g., in [14] the authors combined state (reseeding) and structural (multi-polynomial) reconfigurabilities.

Note that optimizing the Daisy-chain secure test method using the above scheme is not allowed as the circuitry of the random test pattern generator (i.e., the circuitry of the hash function) cannot be manipulated; this is because otherwise an attacker could use this feature to gain access to the cryptographic key during operation mode.

4 Experiments Results

We first define the experiments. Thereafter, we present and discuss the results. Finally, we investigate the impact of reconfigurable RTPG on test quality improvement.

4.1 Experiments Performed

We synthesized the Fuzzy Extractor, described in VHDL, using 0.35μm technology node and Synopsys Design Compiler. The design compiler outputs a verilog netlist that is used to extract a fault list with Synopsys *Automated Test Pattern Generation* (ATPG) tool TetraMAX. We used LIFTING fault simulator optimized for functional BIST to analyze the fault coverage [17]. The results are analyzed thereafter with MATLAB. The experiments performed to evaluate each of the proposed schemes are described next.

Daisy-chain secure test method: To evaluate the quality of the proposed solutions in Fig. 4 in terms of fault coverage (FC), test time and area overhead, we performed the following six experiments:

1) **Default:** we simulated the circuit as in Fig. 4a for 15×10^4 clock cycles and analyzed the FC. This number of clock cycles is assumed to be our test time budget for all remaining experiments.
2) **MISR:** we simulated the circuit as in Fig. 4b (equivalent to Fig. 4c).
3) **SISR:** we simulated the circuit as in Fig. 4d.
4) **SISR + MISR:** we simulated the circuit combining SISR and MISR as in Fig. 4e (equivalent to Fig. 4f).
5) **Default + SISR:** we simulated the circuit in two stages. First, as in Fig. 4a, we simulated the FE using the default loop-chain for 25 % of the test time budget. Second, as in Fig. 4d, we included the SISR in the chain flow (between the Repetition Encoder and Repetition Decoder blocks) and analyzed the FC over the remaining 75 % of the test time. The goal of this experiment is to analyze the impact of *combining* the default scenario as in Fig. 4a with that of SISR in Fig. 4d.
6) **Default + MISR:** in this experiment, we repeated the procedure (5), but replacing the SISR with a MISR.

Parallel secure test method: To determine the impact that the state (seed) and the structure (polynomial) reconfigurability have on the FC and test time, we performed nine experiments per FE block; each FE is tested using three polynomials, each combined with three seeds. Each of the three used seeds and polynomials are described next. Note that conceptually, output reconfigurability and state reconfigurability are very similar; hence, the influence of the output reconfigurability can be easily derived from the results of the experiments carried out for the state reconfigurability.

(1) **State reconfigurability:** we tested each FE block with a random test pattern generator (primitive polynomial) using three different initial seeds; these are:
   (i) **Seed 0:** all bits are zero except the last bit, which is a one (e.g., '0...00001').
   (ii) **Seed 1:** a randomly chosen starting state (e.g., '10...0110').
   (iii) **Seed 2:** a string of alternating zeros and ones (e.g., '01...10101').

(2) **Structure reconfigurability:** we tested each FE block with a random test pattern generator using three different polynomials; one primitive ('Poly 0') and two non-primitive polynomials ('Poly 1' and 'Poly 2'). The size of the polynomials depends of the number of input bits of the block being tested. The polynomials used for Golay Encoder, Repetition Decoder and Hash Function are:

- (i) **Poly 0:** $x^{16} + x^{14} + x^{13} + x^{11} + 1$.
- (ii) **Poly 1:** $x^{16} + x^{14} + x^{12} + x^{11} + x^{10} + x^7 + x^6 + x^5 + 1$.
- (iii) **Poly 2:** $x^{16} + x^{13} + x^{12} + x^8 + x^4 + x^3 + 1$.

while those used for Repetition Encoder and Golay Decoder are:

- (i) **Poly 3:** $x^{24} + x^{23} + x^{22} + x^{17} + 1$.
- (ii) **Poly 4:** $x^{24} + x^{22} + x^{17} + x^{16} + x^{15} + x^{14} + x^{10} + x^6 + x^5 + x^1 + 1$.
- (iii) **Poly 5:** $x^{24} + x^{22} + x^{21} + x^{19} + x^{17} + x^{14} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^4 + x^3 + x^2 + 1$.

All polynomials were used with 'Seed 0'.

4.2 Results for Daisy-Chain Secure Test Method

Figure 6 shows the FC (y-axis) versus the number of clock cycles (x-axis) of the experiments; part (a) gives the results for the first four experiments, and part (b) for the remaining two experiments. From Fig. 6a we can observe the following.

1) Default experiment realizes a FC of only 35.26 %. This FC is quickly realized in the first 2.3k clock cycles (ccs). The figure clearly shows that the FC remains constant for the remaining clock cycles.
2) For the remaining three schemes, the targeted FC of 95.00 % is achieved after 4.71×10^4 ccs for SISR, after 4.56×10^4 ccs for MISR and after 4.33×10^4 ccs for SISR+MISR. The remaining clock cycles until the end of the experiment lead to an additional FC increment of 1.29 % for SISR, 1.34 % for MISR and 1.43 % for SISR+MISR.
3) Using SISR+MISR is relatively the best method in terms of FC and test time; however, it has the largest area overhead.

From Fig. 6b we can observe the following.

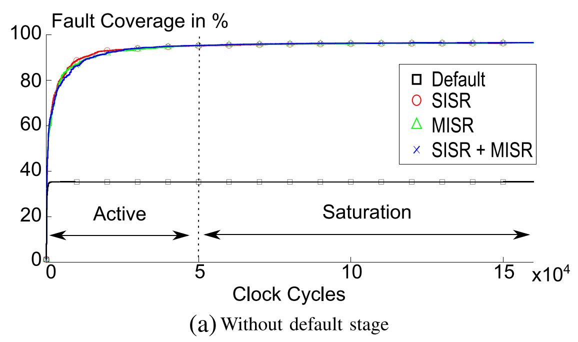
1) During the first stage of Experiment 5 ('Default + SISR') and Experiment 6 ('Default + MISR'), a FC of 35.07 % respectively 34.16 % is realized. This is a little less than the FC achieved with Experiment 1 ('Default') due to the insertion of the extra hardware (SISR/MISR).
2) In the second stage, the FC is significantly increased; Experiment 5 realizes the targeted 95 % FC after 8.57×10^4 clock cycles, while Experiment 6 does this after 9.80×10^4 clock cycles (first stage included).
3) Making use of the entire test time budget of 15×10^4 results in a FC of 96.24 % for 'Default + SISR' and 95.77 % for 'Default + MISR'.

Obviously the inserted blocks required additional area overhead; this is 2.2 % w.r.t. the FE for SISR, 6.80 % w.r.t. the FE for MISR and 9.0 % w.r.t. the FE for SISR combined with MISR.

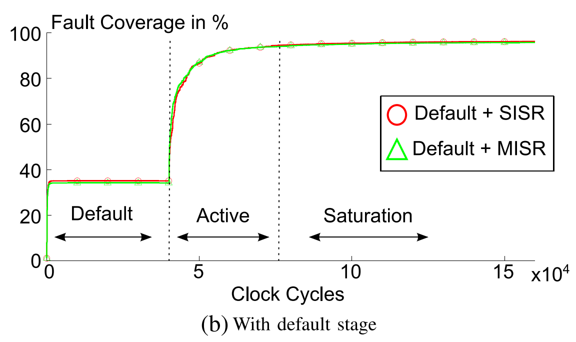
Inspecting the obtained simulation results clearly reveals that testing FE in a loop chain fashion (Fig. 6a) will never realize the required product quality; the FC realized in our case study does not exceed 35 %. Additional DFT to increase the randomness of the test patterns is essential. E.g., introducing a SISR in the loop can increase the FC up to 96.29 % at the cost of 2.2 % area overhead for the predefined test budget.

4.3 Results for Parallel Secure Test Method

The target of this method is to optimize the test time while realizing the targeted FC. The test time will be then



(a) Without default stage



(b) With default stage

Fig. 6 Fault coverage of Fuzzy Extractor Daisy-chain secure test method versus clock cycles

defined by the Golay Decoder which comprises 61.5 % of the FE. The simulation results show that the test time is 3.5k ccs. Moreover, the realized FC by can be obtained by combining the weighted FC of each block. In our case, the FC can be calculated as follows. $FC_{total} = 4.5\ \% \times FC_{GolayEncoder} + 5.2\ \% \times FC_{RepetitionEncoder} + 5.2\ \% \times FC_{RepetitionDecoder} + 53.1\ \% \times FC_{GolayDecoder} + 32.0\ \% \times FC_{HashFunction}$.

4.3.1 Golay Encoder

Figure 7 shows the FC versus the number of clock cycles for the nine experiments. Part (a) of the figure shows the results of 'Poly 0' for the three different seeds and part (b) and (c) present similar results but then for 'Poly 1' and 'Poly 2', respectively. The figure clearly reveals that all experiments realize the targeted FC ($FC_{GolayEncoder}$). Nevertheless, the impact of varying the polynomial and/or the seed on both test time and FC cannot be ignored; e.g., after only six ccs 'Poly 0' with 'Seed 2' reaches FC of 82.30 %, while with 'Seed 0' or with 'Seed 1' this does not exceed 57.67 % and 53.27 %, respectively. Note that 'Poly 1' with 'Seed 1' is the most efficient combination in realizing the targeted FC.

4.3.2 Repetition Encoder

Figure 8 shows that all the nine experiments result in a constant FC of maximum 90.7 % after circa 540 ccs. The impact of the polynomials and the seeds is significant. 'Poly 4' with 'Seed 0' is the best combination realizing FC of 90.7% after 542 ccs.

To reach the targeted FC_{total} with no increase in the test time there are two options. First, investigate and apply the specific test vectors that detect the remaining faults. Second, increase the FC of the other blocks, such that it compensates for the lower FC of Repetition Encoder. The first option is very expensive, as it requires storing the extra vectors on the die. However, the second option is cost-free; simply by extending the test time of one or more of the other blocks, as long as still below the test time budget.

In our case, we chose to extend the FC of the Golay Encoder to 99.8 %, which is realized in 27 ccs when using 'Poly 3' combined with 'Seed 1'.

4.3.3 Repetition Decoder

Figure 9 shows the simulation results. The figure reveals that all experiments realize the targeted FC in no more than 600 ccs. Also here varying the polynomial and/or the seed has a clear impact. 'Poly 2' combined with 'Seed 0' is the most efficient pair realizing the targeted FC.

4.3.4 Golay Decoder

Figure 10 shows the simulation results. All the nine experiments realize the targeted FC. 'Poly 4' combined with 'Seed 2' is the most efficient combination.

4.3.5 Hash Function

Figure 11 shows the simulation results for all the nine experiments. The impact of varying the polynomial and or seed is marginal. After circa 700 ccs, the FC ($FC_{HashFunction}$) for all cases is 95 %.

The results clearly reveal that an appropriate selection of polynomial and or seed significantly increases the FC and/or decreases the test time per FE block, except for the Hash Function. This is due to the specificity of the remaining test patterns required to detect the last remaining faults. Additionally, the results reveal that the required test time per FE block varies significantly (from 14 ccs up to 3.5k ccs) as well as the realized FC. Finally, the results also show that the remaining test time budget is a useful resource to further increase the targeted FC. The additional area overhead of this method is 18.6 % w.r.t. the FE.

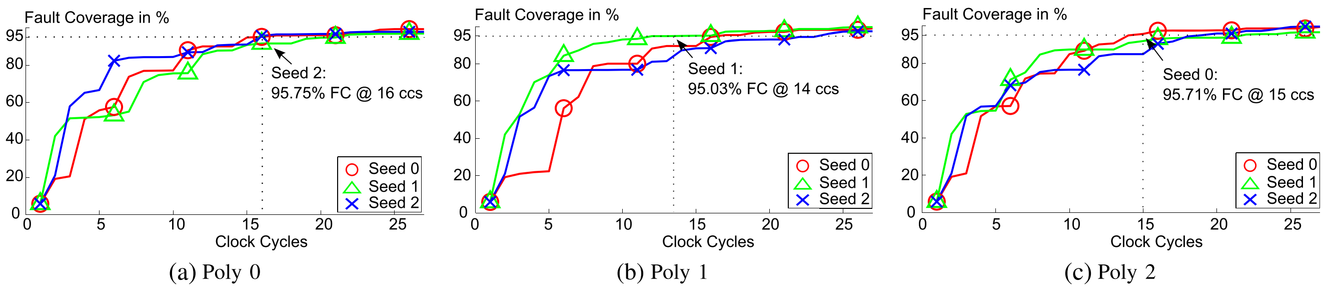


Fig. 7 FC of Golay Encoder with different polynomials and different seeds

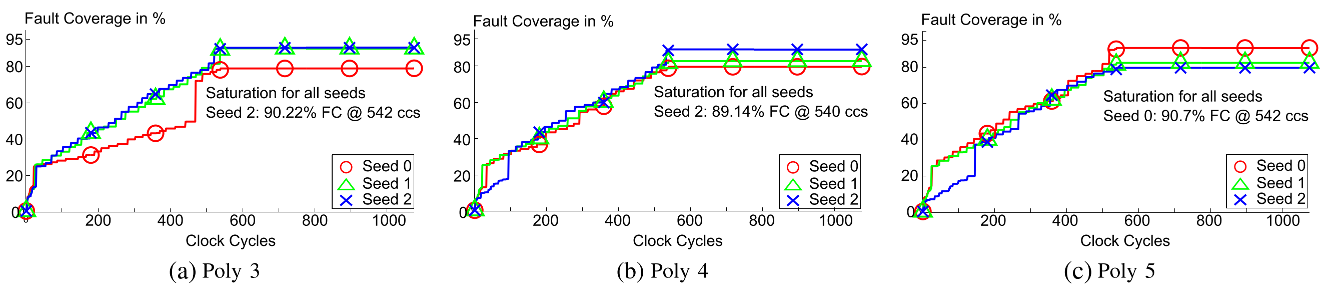


Fig. 8 Fault coverage of Repetition Encoder with different polynomials and different seeds

4.4 Impact of Reconfigurable LFSR/RTPG Parameters

To demonstrate the potential of reconfiguring RTPG parameters in improving FC and reducing the test time, we use the Golay Encoder as a case study. The idea is to skip test patterns without additional FC. For example, Figure 7a shows that during clock cycles 9 and 10 no faults are detected. We skip all the clock cycles that do not contribute to FC but that do consume test time by reseeding, i.e., by changing the seed of the registers of the RTPG. We stop the test when the 95 % FC target is realized. Figure 12a shows the number of faults that each test vector detects (obtained using LIFTING tool [17]), while Fig. 12b shows the impact of skipping the test vectors that do not contribute to FC such that the overall test time is minimized. Figure 12b shows that a speedup of 1.5× can be realized by reseeding three times (speedup from 16 ccs to 11 ccs).

On the downside, the area overhead required for the implementation of the method is very large when compared to the test without reseeding. Each seed requires 12 bits, hence, a total of 36 bits (12 bits×3 seeds) has to be stored on the die. Due to the small size of the Golay Encoder, the extra area overhead that would be needed makes the optimization methods prohibitively expensive for our case study circuit. Moreover, depending on the implementation, extra test time to load the seeds might be required; however, in [18] the authors present a reseeding solution that does not increase the total test time. However, the goal is to find trends to apply in larger circuits. When considering applying one of the test quality improvement methods, the test designer must take into consideration the following parameters: (i) the number of test vectors to anticipate, (ii) the number of bits that need to be flipped from the original RTPG to generate test vectors that detect faults and (iii) the number of times that this operation needs to be performed.

5 Discussion

First we compare both the efficiency of both secure test methods. Second we compare our results with the prior work. Thereafter we make a security analysis of the proposed secure test methods. Finally we provide a list of recommendations to secure test an FE.

5.1 Comparison Between the Secure Test Methods

Table 1 summarizes the main features of the two secure test methods previously proposed. Testing the FE using the parallel secure test method with dedicated RTPGs per block for test pattern generation and for result compression has an area overhead of 18.6 % w.r.t the FE while the area overhead of the daisy-chain secure test method is of only 2.2 % w.r.t. the FE, i.e., the parallel secure test method has 8.5× larger area overhead. Note that the additional area overhead represents a small value, as a typical

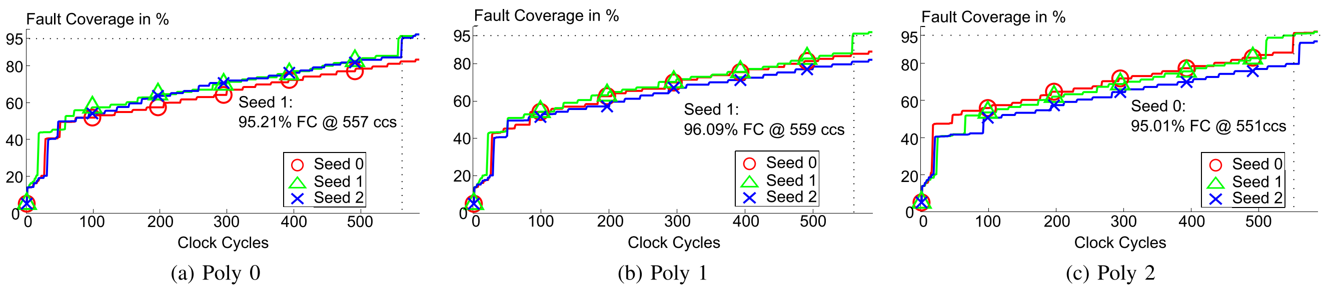


Fig. 9 Fault coverage of Repetition Decoder with different polynomials and different seeds

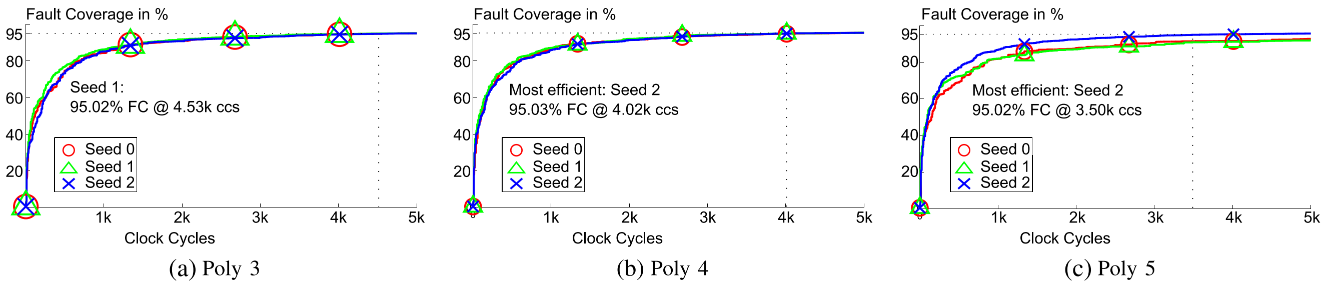


Fig. 10 Fault coverage of Golay Decoder with different polynomials and different seeds

FE has a overall area of 1.4kGE, where GE (Gate Equivalent) is a measure of area in any given technology; 1 GE is the area of a NAND gate [20]. However, it realizes the same fault coverage in 3.5k clock cycles, i.e., 13.46× faster when compared with the 47.1k clock cycles of the daisy-chain secure test method. In addition, having a dedicated RTPG per block enables optimizing methods to achieve higher FC in shorter test time, but again at the expense of area overhead. Optimizing the daisy-chain secure test solution is not feasible because the circuitry of the random test pattern generator (i.e., the circuitry of the hash function) cannot be manipulated as an attacker could use this feature to gain access to the cryptographic key during operation mode.

5.2 Comparison with Prior Work

No prior work focus on the same problem addressed by this paper; hence, a quantitative comparison with prior work is not possible. However, we compare the work qualitatively of the Daisy-chain secure test method to [11] and [9]. The fault coverage of the method is in line with the FC reported in other self-test methods [11] and [9]. The area overhead of the proposed method is negligible, which is intrinsic to methods that reuse hardware. The parallel secure test method can be seen as a test time optimization of the daisy-chain method; realizing the same fault coverage but in 14.3× less time at the cost of 8.5× more area, which is a common trade-off.

5.3 Security Analysis

Ideally, the design of a secure device begins with identifying which class(es) of attacks it should prevent. Several countermeasure methods must be combined to deliver the required level of security. In other words, no method alone is secure. In this work we aim at the prevention of side-channel attacks by means of the test infrastructure. Our BIST methods do not allow data to be scanned-in nor do they leak information on the test results (only pass/fail).

We make a brief analysis of the security of the methods proposed considering the following vulnerabilities. We consider that an attacker a) might use Hardware Trojan to gain access to the switch between functional and test modes to get knowledge on the full or partial value of either PUF or key and b) might try to attack the stored seeds to, e.g., decrease the fault coverage.

With respect to attack a), during test mode, regardless if activated by a legitimate source or by a Hardware Trojan, all registers are reset. Therefore, any traces sensitive information are destroyed. However, considering that the inserted Hardware Trojan can circumvent this protection measure, we would need to combine/enhance our methods with one or several countermeasures proposed in the literature; e.g., [1].

Considering an attack on the stored seeds. Attacks aiming the stored seeds would need to be invasive attacks. Typically, invasive attacks require the depackaging of the device

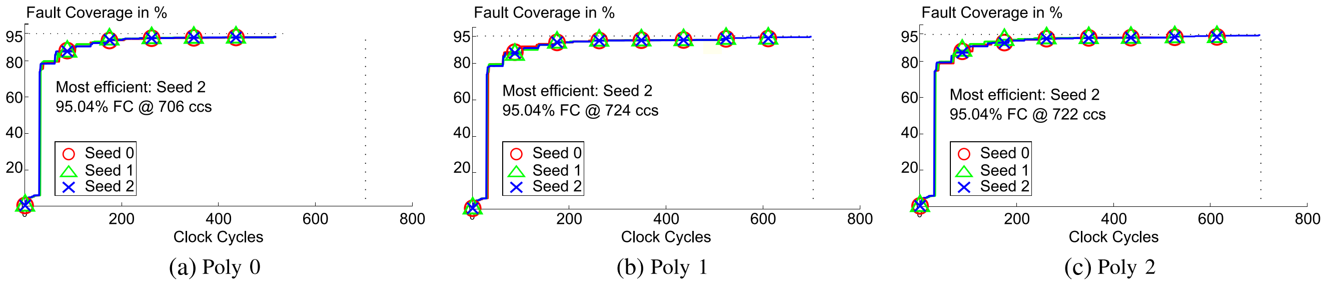
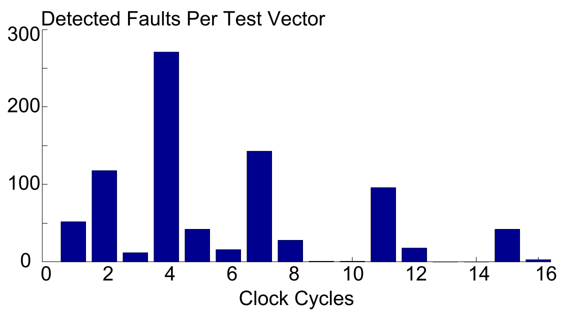
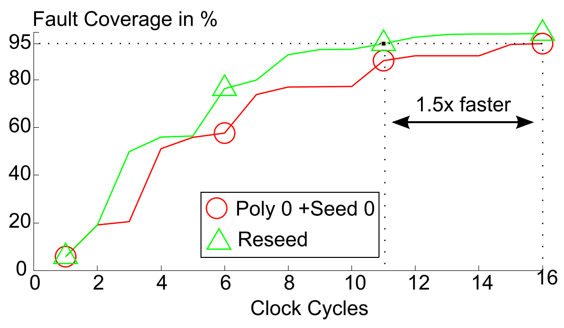


Fig. 11 Fault coverage of Hash with different polynomials and different seeds



(a) Detected faults per test vector



(b) FC before and after reseeding

Fig. 12 Reseeding results for Golay Encoder

under attack and other destructive measures. Destroying a device to change its test seeds to lower its fault coverage it is not profitable. Therefore, stored seeds do not seem vulnerable to this type of attack, as no information could be gained from it.

5.4 Recommendations

We provide a generic step-by-step procedure for secure testing of FE based on our findings. These steps are:

1) Identify each block and deeply understand its functionality (which is critical for successful functional testing).
2) Assess the characteristics of each block (e.g., its area overhead, if the block comprises a state-machine or not and state-machine complexity).
3) Identify possible challenges by testing a certain block with a random input source.
4) Implement parallel secure test method for the shortest test time and implement Daisy-chain secure test method for lowest area overhead.
5) Identify if extra components are needed in order to increase the fault coverage, such as a SISR. If so, analyze the trade-off of such components in terms of their possible locations and of its impact on security, fault coverage, test time and area overhead.
6) Explore further the two secure test methods; optimize parallel secure test method by choosing an optimal polynomial and seed or optimize Daisy-chain secure test method by using a SISR to activate uncovered paths in the state-machine.
7) Analyze the FC, test time and area overhead of all test methods separately and when combined with complementary schemes.
8) Determine and select the best test method and complementary schemes to meet the design requirements.

Table 1 Test methods' results

Test Method	Area overhead	Test time	FC
Daisy-chain (+ SISR)	2.2%	47.1k clock cycles	95 %
Parallel (+ DFT)	18.6 %	3.5k clock cycles	95 %

6 Conclusion

We demonstrated two secure test methods for a Fuzzy Extractor, both are scan-chain free. The first secure test method is based on daisy-chains; it reuses Fuzzy Extractor blocks for test pattern generation and output compression. The second method tests all the Fuzzy Extractor blocks simultaneously by adding dedicated test pattern generation blocks. The results show that the first method has an inherent low area overhead 2.2 %, while it realizes a fault coverage of 95 % using only 47.1k clock cycles. The second method realizes a similar fault coverage with 8.5× more area overhead but 13.46× faster, when compared with the first method. In addition, we identified and analyzed techniques to optimize the previous methods, and provided a generic step-by-step procedure to test any given Fuzzy Extractor based on our findings.

This case study considers a small FE. Nonetheless, the proposed approaches are still valid for any FE construction. Moreover, the two proposed methods provide upper and lower bounds for both test time and area overhead. A real system would benefit from a hybrid solution between the two proposed methods.

Acknowledgments The authors would like to thank Geert-Jan Schrijen and Peter Simons from Intrinsic-ID B.V. for the useful discussions on the architecture of the Fuzzy Extractor used in this publication. It is worth noting that the work presented in this publication was partially sponsored by COST action TRUDEVICE IC1204.

References

1. Agrawal D, Baktir S, Karakoyunlu D, Rohatgi P, Sunar B (2007) Trojan Detection using IC Fingerprinting, IEEE Symposium on Security and Privacy (SP) pp 296–310
2. Ali SS, Said SM, Sinanoglu O, Karri R (2013) Scan Attack in Presence of Mode-Reset Countermeasures. IEEE International online testing symposium (IOLTS) 230:231
3. Al-Yamani AA, McCluskey EJ (2003) Built-in reseeding for serial BIST. VLSI Test Symp:63–68
4. Bo Y, Kaijie W, Karri R (2004) Scan-based Side-Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard, Proceedings of International Test Conference, pp 339–344
5. Cortez M, Roelofs G, Hamdioui S, Di Natale G (2014) Testing PUF-Based Secure Key Storage Circuits Design, Automation and Test in Europe Conference and Exhibition (DATE), pp 1–6
6. Da Rolt J, Di Natale G, Flottes ML, Rouzeyre B (2013) A Smart test controller for scan-chains in secure circuits. IEEE International On-line Testing Symposium (IOLTS):228–229
7. Da Rolt J, Di Natale G, Flottes ML, Rouzeyre B (2012) New security threats against chips containing scan chain structure. IEEE International Symposium on Hardware-Oriented Security and Trust (HOST) p 110
8. Das A, Kocabaş Ú, Sadeghi AR, Verbauwhede I, Sadeghi AR, Verbauwhede I (2012) PUF-based Secure Test Wrapper Design for Cryptographic SoC Testing Design, Automation and Test in Europe Conference and Exhibition pp 866–869
9. Di Natale G, Doulcier M, Flottes ML, Rouzeyre B (2010) Self-test techniques for crypto-devices. IEEE Trans VLSI Syst 18:2
10. Dodis Y, Reyzin L, Smith A (2004) Fuzzy Extractors: How to Generate Strong Keys from Biometrics and other Noisy Data, Advances in Cryptology-EUROCRYPT vol. 3027, LNCS, Springer Berlin Heidelberg, pp 523–540
11. Doulcier M, Flottes ML, Rouzeyre B (2008) AES-based BIST: self-test, test pattern generation and signature analysis. IEEE International symposium on electronic design, Test & Applications, pp 314–321
12. Guajardo J, Kumar SS, Schrijen GJ, Tuyls P (2007) FPGA Intrinsic PUFs and their Use for IP Protection. Workshop on Cryptographic Hardware and Embedded Systems (CHES): 63–80
13. Hamdioui S, Di Natale G, van Battum G, Danger JL, Smailbegovic F, Tehranipoor M (2014) Hacking and Protecting IC Hardware Design, Automation and Test in Europe Conference and Exhibition, pp 1–7
14. Hellebrand S, Rajski J, Tarnick S, Venkataraman S, Courtois B (1995) Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers. IEEE Trans Comput 44(2):223–233
15. Hely D, Bancel F, Flottes ML, Rouzeyre B (2006) A Secure Scan Design Methodology Design, Automation and Test in Europe Conference and Exhibition, pp. 1–2
16. http://mathworld.wolfram.com/GolayCode.html
17. http://www.lirmm.fr/
18. Krishna CV, Jas A, Touba NA (2001) Test vector encoding using partial LFSR reseeding. Int Test Conf:885–893
19. Lee J, Tehranipoor M, Patel C, Plusquellic J Tehranipoor M, Patel C, Plusquellic J (2005) Securing scan design using lock and key technique. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT), pp 51–62
20. Leest Vvd, Preneel B, Sluis Evd (2012) Soft Decision Error Correction for Compact Memory-Based PUFs using a Single Enrollment, Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp 268–282
21. Linnartz JP, Tuyls P (2003) New shielding functions to enhance privacy and prevent misuse of biometric templates. In: Proceedings of Audio- and Video-Based Biometric Person Authentication (AVBPA) 03, pp 393–402, Springer Berlin / Heidelberg
22. Lei L, Chakrabarty K (2004) Test set embedding for deterministic BIST using a reconfigurable interconnection network. IEEE Trans Comput Aided Des Integr Circ Syst 23(9):1289–1305
23. Liu Y, Wu K, Karri R (2011) Scan-based attacks on linear feedback shift register based stream ciphers. ACM Trans Design Autom Electr Syst 16(2):20
24. Pless V (1986) Decoding the golay codes. IEEE Trans Inf Theory 32(4):561–567
25. Skoric B, Tuyls P, Ophey W (2005) Robust key extraction from physical unclonable functions, Applied cryptography and network security, vol. 3531 of LNCS, pp 99135, Springer Berlin / Heidelberg
26. Touba NA, McCluskey EJ (2001) Bit-fixing in pseudorandom sequences for scan BIST. IEEE Trans Comput Aided Des Integr Circ Syst 20(4):545–555
27. www.unique-project.eu
28. Wunderlich HJ, Kiefer G (1996) Bit-flipping BIST. IEEE/ACM Int Conf Comput Aided Des:337–343
29. Yang B, Wu K, Karri R (2006) Secure Scan: A Design-for-Test Architecture for Crypto Chips. IEEE Trans Comput Aided Des Integr Circ Syst 25(10):2287–2293

Mafalda Cortez received her M.Sc. degree in Electrical and Computers Engineering - Telecommunications, Electronics and Computers from Faculdade de Engenharia da Universidade do Porto (FEUP), Portugal. She is currently pursuing her PhD at the Computer Engineering Lab from the Delft University of Technology in collaboration with Intrinsic-ID B.V., a company on Hardware Intrinsic Security. Her research interests include circuit design and modelling, hardware security and secure IC test.

Gijs Roelofs received his M.Sc. degree in Embedded Systems from the Delft University of Technology in 2013. Currently he is a Security Evaluator at Brighsight B.V., an IT security lab, which tests Integrated Circuits, Smart Card Software, including crypto libraries and payment terminals. He specialized in Semi Invasive Attacks, namely Laser Attacks and Electro Magnetic Fault Injecting. Over the years he has gained expertise in testing state-of the-art secure systems. His current research is focused on attacks of products with Near Field Communication and Single Wire Protocol.

Said Hamdioui received the MSEE and PhD degrees (both with honors) from Delft University of Technology (TUDelft), Delft, The Netherlands. He is currently co-leading dependable-nano computing research activities within the Computer Engineering Laboratory of TUDelft. Prior to joining TUDelft, Hamdioui worked for Intel Corporation (in Santa Clara and Folsom, Califorina), for Philips Semiconductors R&D (Crolles, France) and for Philips/ NXP Semiconductors (Nijmegen, The Netherlands). His research interests include Nano-Computing, Dependability, Reliability, Hardware Security, Memristor Technology, Test Technology & Design-for-Test, 3D stacked IC, etc. Professor Hamdioui published one book and co-authored over 130 conference and journal papers. He has consulted for many semiconductor companies (such as Intel, ST Microelectronics, Altera, Atmel, Renesas, DS2, ST Microelectronics, etc). He is strongly involved in the international test technology community as a member of organizing committees or as a member of technical program committees of the leading conferences, as a reviewer for major journals, etc. He delivered dozens of keynote speeches, distinguished lectures, and invited presentations and tutorials at major international forums/conferences and leading semiconductor companies. Hamdioui is a Senior member of the IEEE; he serves on the editorial board of the Journal of Electronic Testing: Theory and Applications (JETTA), on that of Design and Test. He is also a member of AENEAS/ENIAC Scientific Committee Council (AENEAS =Association for European NanoElectronics Activities).

Giorgio Di Natale received the PhD in Computer Engineering from the Politecnico di Torino (Italy) in 2003 and the HDR (Habilitation Diriger les Recherches) in 2014 from the University of Montpellier II (France). He is currently a researcher for the National Research Center of France at the LIRMM laboratory in Montpellier. He has published publications spanning diverse disciplines, including VLSI Testing, Memory Testing, Fault Tolerance, Reliability, Hardware Security and Trust. He is the Action Chair of the COST Action IC1204 (TRUDEVICE) on Trustworthy Manufacturing and Utilization of Secure Devices. He is the chair of the European group of the TTTC, Golden Core member of the Computer Society and Senior member of the IEEE.