Low-Cost Self-Test of Crypto Devices

G. Di Natale, M. Doulcier, M. L. Flottes, B. Rouzeyre
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université Montpellier II / CNRS UMR 5506
161 rue Ada, 34392 Montpellier Cedex 5, France
{dinatale,doulcier,flottes,rouzeyre}@lirmm.fr

Abstract

Testability is a major issue, particularly for secure chips. Design-for-Testability techniques based on scan chains proved to be a highway for potential attacks. BIST approaches appear as good alternatives since they do not rely on visible scan chains. In this paper we propose a generic BIST solution for block-cipher devices. Taking advantage of the iterative process involved in such encryption algorithms which results in structural implementation consisting of (quasi) identical round transformations executed by the same piece of hardware, self-test procedures are easily set-up. Compared to classical BIST solutions based on pseudo-random test pattern generation and output responses compactors, its main advantages are a negligible area overhead and a very short test time, while guaranteeing 100% of fault coverage.

1. Introduction

Public, industry, and state agencies rely on cryptography for the protection of information and communications in various domains of application such as pay TV, e-commerce, critical infrastructures, etc. At the core of the device offering cryptographic services is the cryptographic module. Crypto-cores execute cryptographic algorithms for providing services such as privacy, confidentiality, integrity, and authentication. Although cryptography is used to provide security, weaknesses such as weak crypto-algorithms, poor design or physical failure of the hardware platform that implements the crypto-algorithm can render the product insecure and place highly sensitive information at risk. Consequently, appropriate validation and testing of the crypto-algorithm and corresponding crypto-core are essential to provide security assurance.

United States National Institute for Standards and Technology (NIST) organized contest for selecting encryption standards. The Data Encryption Standard (DES) [1] was adopted as national standard in 1976, and the Advanced Encryption Standard (AES) [2] has been selected in October 2000. Since the hardware implementation of DES is not expensive, it is still used in many applications in the form of Triple DES for security improvement [3].

Validation of such algorithms for efficient encryption is not discussed here. This paper aims at providing efficient test solution for the physical platform that implements the crypto-algorithm, i.e. the dedicated piece of hardware that executes the encryption.

Independently of the intended function, defects created during the manufacturing process of integrated circuit (IC) are unavoidable and some number of ICs is expected to be faulty. Post-manufacturing testing is thus required to guarantee fault free products. It's all the more important for applications requiring digital security because a faulty chip could fail to protect the secret data.

IC testing consists of applying a set of test stimuli to the inputs of the device under test (a crypto core for instance) while analysing the output responses. Circuits that produce the expected responses pass the test and are considered to be fault-free.

Due to the extremely large number of possible defect types and defect locations, fault models are used for computational efficiency during fault simulation and test stimuli generation. A combination of different fault models is generally used in the evaluation of testing approaches; among them the stuck-at fault model remains inescapable.

Detection of such faults generally requires test-oriented design methodology that aim to facilitate generation of proper test stimuli. Scan design is the most widely used structured Design-for-Testability (DfT) methodology. While it greatly facilitates the test of the IC and minimizes the probability to deliver faulty chips, it compromises the security of the system

since it provides facilities for controlling or observing sensitive data. Scan based attacks have been demonstrated in [4] (DES) and [5] (AES). Countermeasure such as secure scan design methodologies detailed in [5][6][7][8][9] prevent abusive usage of the scan facility but requires extra area and design efforts.

Conversely, the Built-In Self-Test (BIST) approach does not require visible scan chains. The test patterns are classically generated on chip by an additional Test Pattern Generator (TPG) and test responses are compacted into a signature before comparison with the pre-computed golden one with the help of a Signature Analyzer (SA). The result of the comparison is the only test output. This test strategy is a good alternative if it provides low area overhead and acceptable fault coverage.

Note that apart from its recurrent cost, extra silicon area for BIST may in turn be subject to faults. As usual, additional hardware for BIST implementation must be kept as low as possible.

Conversely to scan design relying on deterministic test sequences, BIST classically relies on pseudo-random sequences due to the impossibility to store or generate deterministic sequences at low cost with built-in hardware.

However, pseudo-random testing is an efficient technique for crypto-cores [10]. High fault coverage can be achieved with short pseudo-random test sequences because traditional cryptographic operations (XOR, substitution, modulo, …) are easily tested with random data. Moreover, the inherent properties of these operations allow the propagation of random data through the circuit.

In order to save TPG and SA related area overheads, we propose a BIST methodology specifically designed for block-cipher circuits. The proposed BIST technique incurs almost no area overhead.

The paper is organized as follows: Section 2 discusses inherent properties of cryptographic algorithms and it introduces the BIST approach. Section 3 describes the DES and the AES, while Section 4 presents experimental results. Eventually, Section 5 concludes this paper.

2. Cryptography and testability

The security provided by block cipher algorithms such as Data Encryption Standard (DES) and its successor the Advanced Encryption Standard (AES) relies on two main properties named "diffusion and confusion". *Confusion* refers to making the relationship between the key and the ciphertext as complex and involved as possible; *diffusion* refers to the property that redundancy in the statistics of the plaintext is "dissipated" in the statistics of the ciphertext. Those properties are supported by using a Feistel [11][1] network in the first case and by a substitution-permutation network for the later one (see next sections).

These two algorithms also have some common characteristics:

- They are iterative algorithms. DES is composed of 16 rounds while AES is made of 10 rounds. All rounds are identical i.e. the result of a round is used as the input of the next round. Since the rounds are identical, their hardware implementations typically consist of a single round and a feedback loop.
- Ciphering and deciphering algorithms are almost identical.
- Since block ciphering is a bijective operation (one-to-one mapping), each round is a bijective operation too (on a set of 2^{64} elements for DES, on a set of and 2^{128} elements for AES).

The diffusion property is a very interesting feature with regard to the test of their hardware implementation:

- It implies that every input bit of a round influences many output bits, i.e. every input line of a round is in the logic cone of many output bits. In other words, an error caused by a fault in the body of the round is very likely to propagate to the output. Thus, the circuit is very observable.
- Moreover, since rounds are bijective, the input logic cone of every output contains many inputs. In other words, each fault is highly controllable.

It can be concluded that the circuits are highly testable by nature.

Thus, for this kind of circuit, we propose the following self-test procedure:

1. Encrypt an initial message M_0 into $M_1=E(M_0)$
2. Repeat n times : $M_{i+1} = E(M_i)$
3. Compare the final cipher M_n with the expected one E(E(E(…..E(M)…)). If they differ, the circuit is faulty otherwise it is correct.

In other words, the result of an encryption is used as the next test vector. It should be noticed than for *n* encryptions, the main part of the circuit under test (i.e. the hardware implementation of the round) receives *R* test vectors, being *R* the number of iterations of the concerned algorithm (*R*=10 for the AES, *R*=16 for the DES).

We investigate now whether this procedure leads to the application of $n \times R$ distinct test vectors.

To our knowledge, there is no published general result about the length of the cycles on the output state

graph for either algorithm. Nevertheless, we conjecture that the length k of such a cycle (i.e. $M_{i+k} = M_i$) is quite large. Since the support set is very large (2^{64} or 2^{128}), and the encryption is bijective, these two algorithms can be considered as random permutations. As a consequence, the probability distribution function is flat. It can be computed that the expected cycle length is $2^{64}/2$ for DES and $2^{128}/2$ for AES. Thus the probability that the output states fail in a loop is very small for moderate values of n. In practice (see section 5) we never observed such cycles.

To resume, whatever the initial message M_0 and the secret key (except weak keys of DES), the actual length of the test sequence will be $n \times R$.

Finally, concerning test response comparison, due to the diffusion property and to the size of the support sets, the fault masking phenomenon is very unlikely to happen. That is, there are very few chances that in the presence of a fault the final signature equals to the correct one (and again we never noticed this phenomenon).

3. Symmetric encryption algorithms

In this section we describe the characteristics of two considered algorithms Data Encryption Standard (DES) and Advanced Encryption Standard (AES).

3.1. Data Encryption Standard (DES)

The Data Encryption Standard (DES) has been selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976. DES is a block cipher, with a block size of 64 bits and a key of 64 bits. However, only 56 bits of the key are actually used by the algorithm, while the other 8 bits are used for checking parity, and are thereafter discarded.

The algorithm's overall structure is shown in Figure 1.a: There are 16 identical stages of processing, called *rounds*. There are also an initial and a final permutation, called IP and FP, which are inverses (i.e., $IP(x) = FP^{-1}(x)$).

In each round, the block is divided into two 32-bit halves and processed alternately. This crossing scheme is known as the Feistel scheme [11]. The F-function scrambles half a block together with some of the key. The output from the F-function is then combined with the other half of the block, and the halves are swapped before the next round. After the final round, the halves are not swapped; this is a feature of the Feistel structure which makes encryption and decryption similar processes.

The F-function, depicted in Figure 1.b, operates on half a block (32 bits) at a time and consists of four stages:

- Expansion: the 32-bit half-block is expanded to 48 bits using the expansion permutation, denoted E in the diagram, by duplicating some of the bits;
- Key mixing: the result is combined with a subkey using an XOR operation. Sixteen 48-bit subkeys (one for each round) are derived from the main key using the key schedule (described below);
- Substitution: after mixing in the subkey, the block is divided into eight 6-bit pieces before processing by the S-boxes, or substitution boxes. Each of the eight S-boxes replaces its six input bits with four output bits according to a non-linear transformation, provided in the form of a lookup table;
- Permutation: finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation, the P-box.

In order to generate the round keys, the 56 bits of the original key are divided into two 28-bit halves; each half is thereafter treated separately. In successive rounds, both halves are rotated left by one or two bits (specified for each round), and then 48 subkey bits are selected: 24 bits from the left half, and 24 from the right.

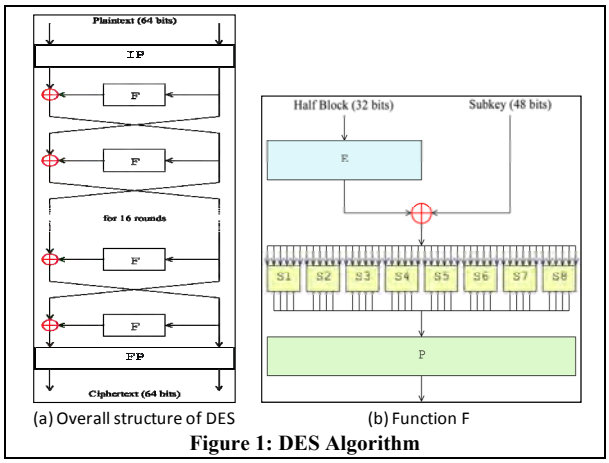

Figure 1: DES Algorithm

3.2. Advanced Encryption Standard (AES)

AES [2] is a block cipher adopted as an encryption standard by the U.S. government. AES began immediately to replace the Data Encryption Standard (DES, used since 1976) for the reason that it outperforms in long-term security thanks to, among other things, larger key sizes (128, 192, or 256 bits).

Another major advantage of AES is its efficient implementation on various platforms. It is suitable for small 8-bit microprocessor platforms, common 32-bit processors, and dedicated hardware implementations that can reach throughput rates in the gigabit range.

The AES algorithm's internal operations are performed on a two dimensional array of bytes called State. For sake of simplicity, we focus on key length equal to 128 bits. The State consists of 4 rows of bytes and each row has Nb=4 bytes. Each byte is denoted by $S_{i,j}$ ($0 \le i < 4$, $0 \le j < Nb$) . The four bytes in each column of the State array form a 32-bit word, with the row number as the index for the four bytes in each word. The 128-bit block can be expressed as 16 bytes: in_0, in_1, in_2… in_{15}. Encryption and decryption processes are performed on the State, at the end of which the final value is mapped to the output bytes array out_0, out_1, out_2, … out_{15}.

The AES algorithm is an iterative algorithm composed of 10 rounds. At the start of encryption, input is copied to the State array. After the initial secret key addition (roundkey(0)), the first 9 rounds are identical, with small difference in the final round. As illustrated in Figure 2, each of the first 9 rounds consists of 4 transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey. The final round excludes the MixColumns transformation.

The encryption scheme in Figure 1 can be inverted to get a straightforward structure for decryption.

SubBytes Transformation

The SubBytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-Box). This S-Box is constructed by composing two transformations:

1. Take the multiplicative inverse in the finite field $GF(2^8)$; the element $(00000000)_2$ is mapped to itself;
2. Apply the following affine transformation (over GF(2)):

$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

for $0 \le i < 8$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c whose value is fixed and is equal to {01100011}.

This transformation can be pre-calculated for each possible input value since it works on a single byte, therefore there are only 256 values. S-Boxes can be implemented either as a ROM or as combinational logic.

ShiftRows Transformation

In this transformation, the bytes in the first row of the State do not change. The second, third, and fourth rows shift cyclically to the left one byte, two bytes, and three bytes, respectively.

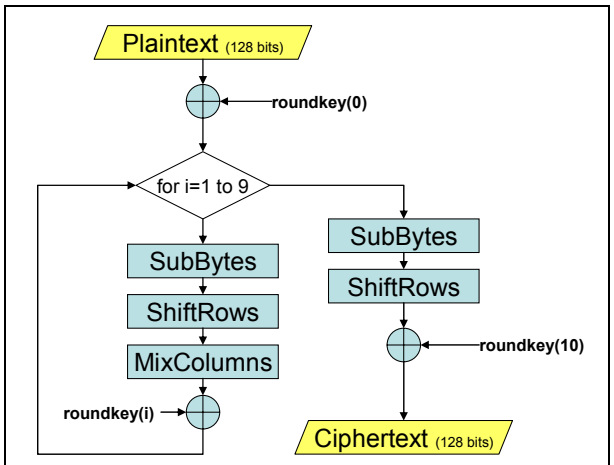

Figure 2: AES Algorithm (encryption)

MixColumns Transformation

The MixColumns transformation is performed on the State array column-by-column. Each column is considered as a four-term polynomial over $GF(2^8)$ and multiplied by a(x) modulo $x^4 + 1$, where:

$$a(x) = (00000011)_2\, x^3 + (00000001)_2\, x^2 + (00000001)_2\, x + (00000010)_2$$

AddRoundKey Transformation

In AddRoundKey transformation, a roundkey is added to the State array by bitwise XOR operation. Each roundkey consists of 16 words generated from Key Expansion described below.

Key Expansion

The key expansion routine, as part of the overall AES algorithm, takes the input secret key of 128 bits. The output is an expanded key of 11*128 bits, i.e., the expanded key is composed of the secret key and 10 roundkeys, one for each round. Details of the algorithm that allows determining the value of each roundkey are given in [2].

4. Testability Analysis

In the following two sub-sections we provide some results related to the area overhead and the fault coverage for the self-test scheme, applied to the DES and the AES. The two architectures have been described in VHDL and synthesized using Synopsys Design Compiler [14] using a 350nm CMOS library provided by AMS [15].

In both cases we have studied a theoretical approach to pre-calculate the number of encryptions needed to reach 100% of fault coverage. We considered the following aspects:

1. the stream generated by a crypto algorithm when the input is fed by its output can be considered as

random. Strong randomness is an inherent feature of crypto algorithms. This property has been confirmed using the NIST statistical tests [12]. In both cases the bit streams passed the 15 randomness test [13];

2. substitution boxes represent the biggest part of the device, and their inputs are independently fed by a sub-part of the input. We can therefore assume that they are fed by a random stream;
3. each substitution box needs N deterministic patterns to be fully tested and it receives one pattern every clock cycle;
4. we assume that test patterns able to test faults in the Sbox are also able to test faults in the remaining parts of the circuit (see 5.1 and 5.2 for details on the particular architecture).

All these points allow us to estimate the number of clock cycles (and therefore the number of encryptions) required to fully test the circuit using the formula that gives the minimal-length random sequence that would include N patterns, having probability p to appear, with a given confidence level [13]:

$$P[X \le n] = 1 - \sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} (1-ip)^n \qquad (1)$$

4.1 DES Hardware Implementation

The architecture of the Self-Test approach for DES is depicted in Figure 3. The area overhead of the proposed approach is equal to 3,58%, corresponding to the initial 64-bits multiplexer and some additional logic for the control unit.

In order to determine the number of clock cycles required to fully test the circuit we applied the equation (1). In particular, we considered a confidence level of 99% for a sequence of k=64 patterns (i.e., the exhaustive set of input pattern for the Sbox), where each pattern has the same probability to appear (p=1/64). From this equation it comes that the length of a random sequence that contains each necessary vector with a confidence level of 99% is n=540 patterns.

According to the implementation of the Sboxes, and in particular based on the actual number of deterministic patterns k (≤64) required to fully test the Sbox, the length of the test procedure can vary from 440 clock cycles (28 encryptions, being each encryption composed of 16 clock cycles) to 540 cycles (34 encryptions).

Concerning the other parts of the DES, they are mainly xor operations and they should be very easily tested using the same random patterns issued from the Sboxes.

Experimental results confirmed this result. We fault simulated the DES with several keys and initial input messages. It comes that after 21 encryptions (i.e. 336 clock cycles) the circuit is fully tested.

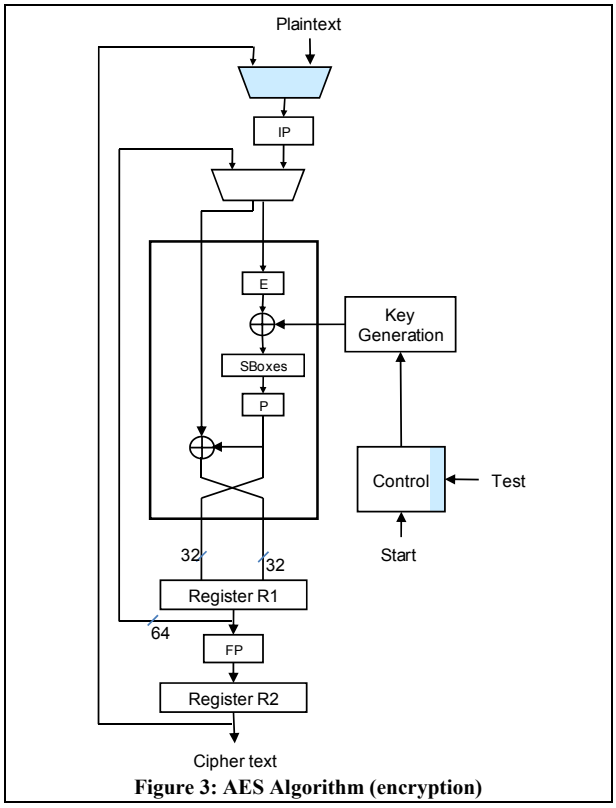

Figure 3: AES Algorithm (encryption)

4.2 AES Hardware Implementation

The architecture of the Self-Test approach for AES is depicted in Figure 4. The area overhead of the proposed approach is equal to 2,13%.

Regarding the testability and the number of encryptions required to reach 100% of fault coverage, we computed the minimal-length random sequence (with a confidence level of 99%) that would include k=256 patterns (i.e., the exhaustive set of input pattern for the Sbox), where p is equal in this case to $1/2^8$ (since the Sbox operates on 8 bits). From this equation it comes that the minimal random sequence length is n=2593 patterns.

The same experiment have been performed for various implementations of the Sboxes and thus for different minimal deterministic test sets. In any case, the theoretical minimal length of the random sequence for including the targeted deterministic patterns ranges from 2400 to 2593 patterns.

Concerning the other operations of the AES, ShitRow function requires only wires for its implementation and is tested when every bit of this interconnection structure has been set to both “0” and “1” (under the assumption of stuck-at fault model).

This should be easily achieved with the patterns issued from the Sboxes (bijective operations fed with 2600 random patterns). MixColumn and AddRoundKey operations are mainly xor trees and should be very easily tested too using random patterns issued from the Sboxes.

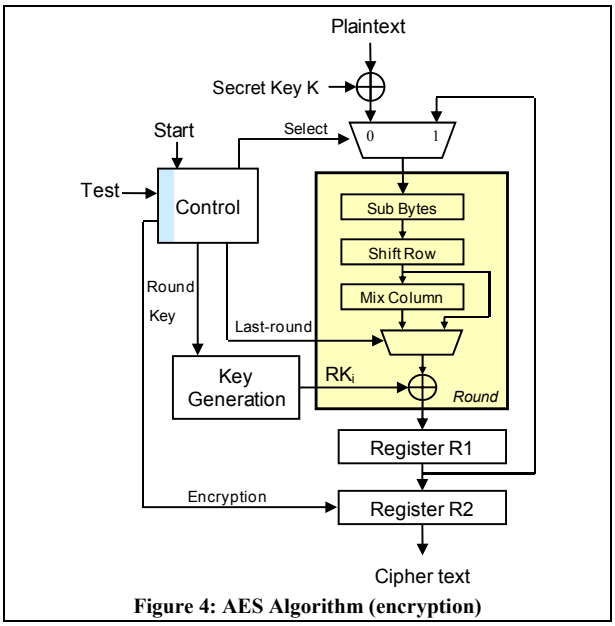

Figure 4: AES Algorithm (encryption)

As for the DES, in order to confirm this hypothesis we have performed a fault simulation on the proposed AES core sets in self-test mode. This experiment has shown that all the faults have been tested after 210 encryptions (i.e. 2100 round cycles). This experiment has been repeated with different plaintexts and secret keys as starting points. We obtained test sequences ranging from 2100 to 2500 patterns for 100% fault coverage, as expected from the equation (1).

From a practical point of view, 2600 round cycles in self-test mode should be sufficient to test the whole structure with a confidence level of 99%.

5. Conclusions

In the context of secure circuits, BIST approaches appear as good alternatives since they do not rely on visible scan chains. In this paper, a generic BIST solution for cryptographic devices is presented. The basic principle is to feed the device with its own output and let the device run for a certain number of encryptions, and then to compare the output of the final encryption with a pre-computed signature.

We showed that the area overhead entailed by this technique is negligible and the required test time is very short, while guaranteeing 100% of fault coverage.

6. References

[1] Data Encryption Standard, Federal Information Processing Standard (FIPS), Publication 46, National Bureau of Standards, U.S. Departement of Commerce, Washigton D.C., January 1977.

[2] Joan Daemen, Vincent Rijmen, "The Design of Rinjael, AES - *The Advanced Encryption Standard*-", Springer, ISBN 3-540-42580-2

[3] NIST, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", Special Publication 800-67.

[4] B. Yang, K. Wu, R. Karri, "Scan-based Side-Channel Attack on Dedicated Hardware Implementations on Data Encryption Standard", Proc. International Test Conference (ITC 2004), pp 339-344.

[5] B.Yang, K.Wu, R.Karri, "Secure Scan: A Design-for-Test Architecture for Crypto Chips", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems TCAD 06, Oct. 2006, Vol 25, Issue: 10, pp 2287-2293.

[6] J. Lee, M. Tehranipoor, C. Patel, J. Plusquellic, "Securing Scan Design Using Lock and Key Technique", Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2005), pp 51-62.

[7] D. Hely; F. Bancel; M.L. Flottes, B. Rouzeyre, "Secure Scan Techniques: a Comparison", IOLTS'06 12th International On-Line Testing Symposium , 2006, pp. 119-124

[8] D. Hely, F.Bancel, M-L.Flottes, B. Rouzeyre, "Securing Scan Control in Crypto Chips", IEEE Journal of Electronic Testing: Theory and Applications 23, 5, oct. 2007 pp 457-464.

[9] D. Mukhopadhyay S. Banerjee D. RoyChowdhury B. B. Bhattacharya, "CryptoScan: A Secured Scan Chain Architecture", 14th Asian Test Symposium (ATS 2005), pp 348 – 353.

[10] A. Schubert, W. Anheier, "On random pattern testability of Cryptographic VLSI cores", Journal of Electronic Testing: Theory and Applications, Volume 16 , pp 185 – 192

[11] H. Feistel, "Cryptography and computer privacy" Scientific American magazine, May 1973, pg:15-23

[12] NIST Special Publication 800-22, "A statistical test suite for random and pseudorandom number generators for cryptographic applications", (with revisions dated May 15, 2001).

[13] Doulcier M., Flottes M.-L., Rouzeyre B., "AES-based BIST: Self-test, Test Pattern Generation and Signature Analysis", DELTA'08: 4th IEEE International Symposium on Electronic Design, Test & Applications, (2008), pp. 314-321.

[14] http://www.synopsys.com

[15] www.austriamicrosystems.com

[16] S. Shioda, "Some upper and lower bounds on the coupon collector problem", Journal of Computational and Applied Mathematics, March 2007, Volume 200, Issue 1, pp 154-167