Secure Mutual Testing Strategy for Cryptographic SoCs

Amitabh Das, Duško Karaklajić and Ingrid Verbauwhede

Abstract

This article presents a secure *mutual* testing strategy for System-on-Chips (SoCs) that implement cryptographic functionalities. Such approach eliminates the need for an additional *trusted* component that is used to test security sensitive cores in a SoC, like symmetric and public-key cryptographic modules. We combine two test approaches: Logic Built In Self Test (BIST) and secure scan-chain based testing and develop a strategy that preserves the test quality of the standard test methods, enhancing security of the testing scheme. In order to minimize the area overhead of the presented solution, we *re-use* the existing modules in different manners: a public-key cryptographic core to build the BIST infrastructure and a symmetric one to authenticate a device under test to a test server, thus preventing an unauthorized user from accessing the test interface. By doing so, we achieve both testability and security at the minimal cost.

Index Terms

Secure Testing, BIST, Scan-chains, Test Wrapper, Cryptographic SoC.

I. Introduction

Embedded System-on-Chips (SoC) used in smart cards, cell phones or travel documents are increasingly employed in applications that assume storing and transferring confidential and privacy sensitive data. In order to prevent a malicious entity to misuse a device, it must implement cryptographic services such as encryption, decryption or digital signing. Testing such a platform introduces an extra dimension to the process of digital testing – security. The testing strategy must be developed in such a way that it prevents an adversary in revealing secret data through the testing circuitry. On the other hand, testability and security are always at odds with each other: enhancing the security level of a design comes at the cost of testability and *vice versa*. Moreover, the complexity and the heterogeneous nature of a SoC, which is designed by embedding the pre-designed functional IP cores from different vendors, requires a modular approach- testing every core separately and the combination of different testing methods. Such a test scenario makes a security assurance even more challenging.

Many of the testing problems discussed above can be overcome by using a Built-In Self-Test (BIST). This approach eliminates or reduces the need for an external tester by integrating active test infrastructure onto the

Department of Electrical Engineering, KU Leuven, ESAT/COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium.
E-mail: firstname.lastname@esat.kuleuven.be

chip. Furthermore, it provides *field-test* capability, which is essential for cryptographic cores and since the internal registers of the device can not be accessed from the BIST I/O interface, it prevents an adversary in revealing the secret data. However, BIST has relatively high area overhead and provides a good test quality only for a limited types of logic. Moreover, it provides only a pass/fail signature which is not useful for diagnosis. On the other hand, scan-chains provide the highest testability employing a more general and flexible approach.

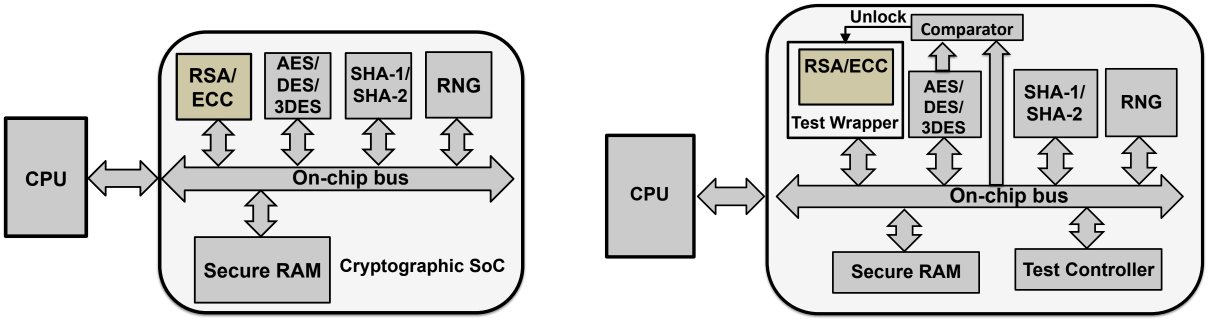
Additionally, the IEEE 1500 Test Wrapper provides a standard for wrapping multi-vendor IP cores in a SoC environment [1]. It is used to provide both test access and test isolation to the core and the external user-defined logic during testing. However, the fact that scan-chains provide observability of the internal nodes from the I/O pins of a device, can be exploited to mount several scan-based attacks to leak the secret keys in cryptographic hardware [2], [3]. Scan attacks on a hardware implementation of AES and related countermeasures in the presence of various commercial test schemes are presented in [4], [5], [6]. Differential scan attacks on hardware implementations of public-key ciphers, RSA and ECC, in the presence of industrial Design-for-Test (DfT) schemes are presented in [7], [8], [9], [10] respectively. In order to address this problem, there is ongoing research in developing a mechanism that ensures that only authenticated users can have access to the test interface [11], [12]. Such a mechanism is denoted as Secure Test Wrapper (STW) later in the paper.

In this article, we combine the two testing approaches – BIST and secure scan-chains and develop a modular and secure testing strategy for cryptographic SoCs, i.e SoCs that are capable of running cryptographic protocols. Symmetric cryptographic cores, such as symmetric ciphers and hash functions, expose very good random pattern testability properties and can be very efficiently tested using BIST [13]. In order to minimize the introduced area overhead, we do not implement standard BIST infrastructure, but *re-use* a modular multiplier from an existing public-key core (RSA, ECC) and build BIST circuitry based on it. We show that the area cost in our scenario is less compared to standard BIST. In order to test the public-key core itself and the other non-cryptographic modules in the SoC, e.g. a microcontroller, we use a secure scan-chain method employing the STW. Instead of adding a dedicated symmetric cipher to implement the authentication framework [12], we *re-use* the one that is already available in a cryptographic SoC. By doing so, we additionally reduce the area overhead, still preserving the test quality and security of a device.

The rest of the paper is organized as follows: Section II introduces the target platform for the presented testing scheme, which then is described in Section III. Sections IV and V analyze the cost and the security of the proposed testing method, respectively. Finally, we conclude in Section VI.

II. CRYPTOGRAPHIC SOC

Security protocols combine many different cryptographic algorithms to establish and maintain a sufficiently secure and privacy preserving connection between two communication parties. A hardware platform intended to run such a protocol (e.g. a smart card) must be able to implement public key cryptographic primitives such as RSA [14], ECC [15], [16] or El Gamal [17] and signature schemes (DSA [18], ECDSA [19]) in order to provide a key establishment or signature generation and verification. In addition, it must also support symmetric key primitives such as block



(a) A Cryptographic SoC.

(b) Testing of Public-key Crypto Cores.

Fig. 1: Cryptographic SoCs.

ciphers (AES [20], DES [21], 3DES [22]) for implementing encryption/decryption operations and hash functions (SHA-1 [23], SHA-2 [24]). In order to reduce the bottleneck they create, these computationally intensive and time consuming algorithms are often realized as hardware co-processors. A typical architecture of a cryptographic SoC which supports different security protocols is depicted in Figure 1a.

The test scenario that we present in this paper aims at testing the shown SoC in an efficient and a secure manner. The next section describes the proposed testing scheme in details.

III. Secure Testing Strategy

A. *Overall strategy*

The overall test strategy is depicted in the flowchart in Figure 2. The test process starts with the execution of the authentication protocol using the AES block cipher in order to check whether the testing server is *genuine*. For more details about the authentication protocol and its execution, we refer the reader to [12].

Only if the authentication succeeds, the test wrapper around the scan-chains of the public-key module is unlocked for testing. Since it can be the case that the authentication protocol fails due to faults in the AES block itself, we foresee the possibility to run it using the hash function block. Only if the authentication fails in that case as well, the testing process is aborted and the failure is reported to the server. Otherwise, the testing continues. Section III-E explains the fail-safe scenario in more details.

After proving that the test server authentic, testing of the public-key core is performed using the secure scan-chains. Then, the multiplier from the tested core is re-used to perform the BIST of the symmetric-key cryptographic cores, i.e. the AES and the hash module. Note that in the case that the initial authentication have failed because of the errors in the AES core, its testing would not be necessary since the outcome of the pass-fail BIST would be already known. However, at this testing stage it is not known weather the authentication was executed using the AES or the hash module and hence testing of both modules has to be performed.

When the testing of *security-critical* cores is finished, the other non-cryptographic modules are tested. Depending on their architecture, either scan-chains or BIST approach could be used. Finally, the remaining form the SoC in

Fig. 2: Secure and Efficient Testing Strategy Flow Diagram

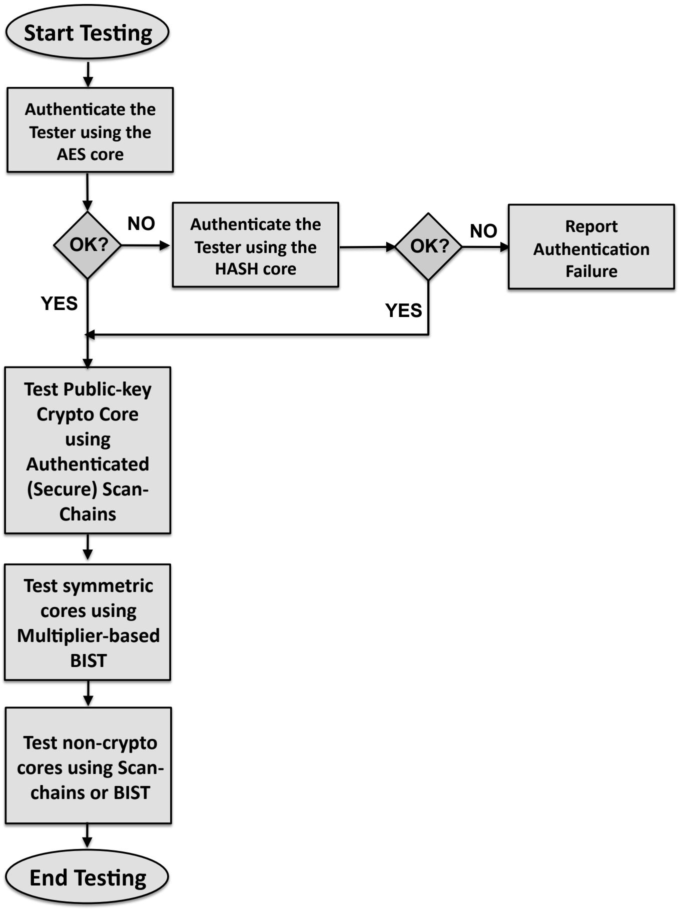

Start Testing
Authenticate the Tester using the AES core
OK?
NO
YES
Authenticate the Tester using the HASH core
OK?
NO
YES
Report Authentication Failure
Test Public-key Crypto Core using Authenticated (Secure) Scan-Chains
Test symmetric cores using Multiplier-based BIST
Test non-crypto cores using Scan-chains or BIST
End Testing

Fig. 1a, the RNG, needs to be tested. It is performed by examining the randomness of its output, using an online NIST or DIEHARD tests.

In the rest of this section we describe testing of different classes of modules in more details.

B. Testing of Public-key Modules

The public-key cryptographic processor can be thoroughly tested using scan-chain Design for Testability (DfT). An IEEE 1500 Test Wrapper, which has become the standard for SoC testing, is used in the form of a locking/unlocking mechanism to disable/enable access to the scan-chains.

A similar approach using a lightweight block cipher to test the AES modul using Test Wrappers was presented in [12]. A component under test (CUT) first sends a challenge to the test server. Using the *a priori* established key, the server encrypts the received challenge using the AES algorithm and sends the ciphertext back to the CUT. Using the same key stored in a non-volatile memory, the CUT calculates the same encryption using the existing AES core. When the received and calculated encryptions match, only then an Unlock Wrapper signal unlocks the wrapper to enable normal testing of the public-key core using scan chains, through the wrapper. This Unlock signal is connected to the AND gates between the input and output wrapper boundary registers (IWBR and OWBR) and the public-key core scan chains.

The AES block on the Crypto SoC has been used in this entity authentication mechanism in the form of a challenge-response protocol (CRP) to allow access to the internal scan chains of the public-key processor only to eligible testers. This requires an on-chip key storage and the AES running on the Test Sever. The data communication between the SoC and the Test Server happens over a serial interface (for instance RS232) or the IEEE 1149.1 compatible JTAG port.

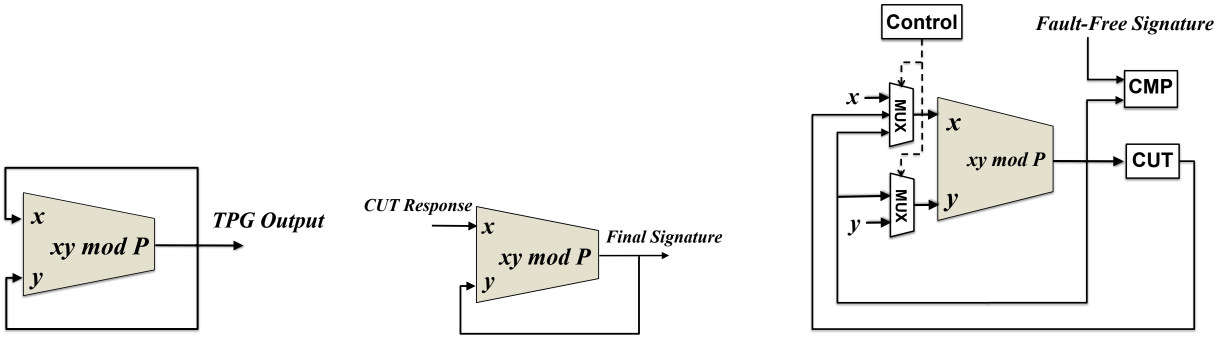
C. Testing of Symmetric-key Modules

The symmetric-key modules in a SoC from Fig. 1a, i.e., symmetric ciphers and hash functions, are tested using a pseudorandom Built-In Self-Test (BIST). Specific structures and operations used for the implementation of these modules provide high pseudorandom testability [13], which makes pseudo-random BIST a suitable test solution for this purpose. Rather than adding the *standard* BIST circuitry, we reuse an existing modular multiplier, which is an inherent part of a public-key module, to build the BIST infrastructure. It is shown that such a solution introduces less hardware overhead [25].

The standard BIST circuitry consists of a Test Pattern Generator (TPG), a Signature Analyzer (SA), and a BIST control unit (BCU). The TPG is used to produce test patterns which are fed into the Circuit Under Test (CUT) as a test stimulus. The SA compacts the CUT's responses into a single signature, which is then compared to the precomputed golden signature. The BCU is needed to activate the test and check the responses. If the two signatures match, the CUT is assumed to be fault-free, otherwise, an error is reported. Efficient test pattern generators and response compactors reusing the aithmetic functions shared with the main datapath have been proposed earlier in [26], [27], [28].

In what follows, we show how a modular multiplier can be configured to act as a TPG and a SA and explain how the presented testing strategy is applied to a cryptographic SoC. For more detailed description this testing strategy, we refer the reader to [25].

- **Multiplier as a TPG.** Figure 3a depicts a modular multiplier configured to act as a TPG. By choosing a proper modulus P, such a configuration implements the *Blum Blum Shub* (BBS) algorithm [29] and provides a sequence of pseudorandom numbers at its output.
- **Multiplier as an SA.** When configured as shown in Figure 3b, the multiplier acts as a signature analyzer. It collects the CUT's responses on its input and provides the final signature at the output. [25] evaluates and proves the good quality of a multiplier based signature analyzer.
- **Configurable Multiplier as a TPG/SA.** By adding two multiplexers at its inputs and a simple controller, a multiplier becomes configurable to act both as a TPG and SA (Figure 3c). The introduced hardware overhead of such an architecture is smaller than if a *regular* BIST infrastructure would be added to a design.
- **Testing Scenario.** To make understanding easier, we hereby explain a possible test scenario in a cryptographic SoC (Fig. 1b). A random pattern testable cryptographic core, such as a symmetric cipher or a hash function,



(a) Modular Multiplier as a TPG. (b) Modular Multipler as a signature analyzer. (c) Configurable Modular Multiplier as both TPG and SA .

Fig. 3: Multiplier based BIST strategy.

acts like a component under test (CUT). The modular multiplier from a public-key cryptographic core is re-used to produce the test patterns, collect the CUT's responses and create a *digest* of the final signature. The BCU controls the testing process and compares the obtained final signature with the precomputed golden one. The existing communication infrastructure in a SoC is used for data transfer between the modules involved in the testing process.

The prime number used as the modulus can be a NIST recommended Pseudo-Mersenne number having efficient modular reduction properties. It needs to be stored in an on-chip non-volatile memory and loaded at the beginning of the authenticated testing process. In case a change in the modulus (to switch between different sizes for RSA and ECC) is required, this non-volatile memory needs to be re-written with the new value.

D. Testing of other non-cryptographic modules

Depending on their testability properties, the non-cryptographic modules can be tested either using BIST or the IEEE 1500 Test Wrapper. In case there is no security requirement for these modules, no activation mechanism would be required. However, in case the non-cryptographic modules need to be protected (for instance, for IP protection), they can also be subjected to the same authentication process using the multiplier-based BIST or Secure Test Wrappers.

E. Test fail-safe scenario

In case the AES block that starts the testing process by initiating the authentication mechanism required for enabling the wrapper around the public-key block (which eventually tests the other blocks) is erroneous, the whole test process may fail. We have designed a fail-safe mechanism in case this happens.

If the AES block has some errors, the authentication mechanism fails. As a backup variant, the available hash module would serve to run the authentication protocol required for enabling the Test Wrapper around the public-key module, which would then test rest of the modules (Fig. 2). It is done by employing the available hash core in a *keyed-hash* mode, using the same shared key as in the case of the AES-based authentication. The only difference is that instead of the encryption of the generated challenge, both the chip under test and the test server calculate its keyed-hash value. A test flag is maintained storing the information whether the AES or the Hash block was used for the authentication procedure, and accordingly the other blocks are tested. This prevents redundant testing of blocks which were tested and found erroneous earlier.

Only if both symmetric-key primitives are erroneous, the testing process is aborted and the failure is reported by the server.

IV. Costs

A. Time Overhead

Security and efficiency of the presented test scheme come at the cost of increased test time. We hereby discuss the additional test time introduced by our secure testing methodology by splitting it into different test phases, as shown in Fig. 2.

- **Authentication phase.** The authentication is performed using a simple *challenge-response* protocol. The time overhead of this testing phase consists of the time needed to generate a challenge from the on-chip RNG, the time to encrypt it on both the test server and the chip and the time to transfer these values via the communication interface.
- **Testing public-key cores.** Once the SoC under test is authenticated and the test wrapper is *unlocked*, a regular scan-chain test process is used to test public key cores available in the SoC. Therefore, this testing phase does not introduce the additional time overhead compared to the ”regular” testing process.
- **Testing symmetric key cores.** This testing phase assumes using the multiplier based BIST methodology. Since the multiplier from the public-key core is reused for this purpose to act as both test pattern generator and the signature analyzer, the time overhead consists of the following elements:
  - *Test pattern generation.* The number of clock cycles needed to generate a test pattern, N_{PAT} depends on the multiplier architecture. In the case of the fully parallel implementation, $N_{PAT} = 1$ cycle per pattern.
  - *Signature analysis.* The cycle count of this phase, N_{SIG}, depends on the multiplier architecture as well. If the multiplication is performed in a single clock cycle, $N_{SIG} = 1$.
  - *Overall.* The number of clock cycles required to test a symmetric cryptographic primitive, N_{SYM} depends on its actual hardware implementation. For instance, the number of pseudo-random patterns needed to test AES presented in [30] is 2534. Overall, the number of clock cycles required to perform the multiplier based BIST of a symmetric core is $N_{SYM} \times (N_{PAT} + N_{SIG})$.

B. Area Overhead

Besides security, the test methodology we propose aims at achieving low area overhead. This section analyses the cost of different components in the presented testing scenario.

Table I specifies the area overhead introduced by different components used in our testing scenario. All the components are synthesized using Faraday 0.13 μm CMOS standard cell library.

TABLE I: Area Overhead of the Secure Testing Method.

Design	Area in # kGates
Security core	71.9
Security core with 10 scan chains	75.9
Security core with scan chains and Test Wrapper	96.4

of probing attacks on it. The attacker cannot observe the bit patterns of the on-chip encryption signal. Otherwise, he can just send a response matching with this sequence, to unlock the wrapper. This can also be achieved by a similar approach as in the previous case.

Some attack scenarios are considered.

- **Chosen plaintext attack** is not feasible as the plaintext is a Random Number nonce.
- **Replay attacks** are not useful for an attacker, due to the same reason. The attacker can only eavesdrop on the serial communication taking place between the secure chip and the test server. Since the random number nonce is generated on-chip by the AES block and sent serially to the server, while the ciphertext on this nonce is generated by the server, it makes no sense to replay it back, as each time, a new nonce is generated.
- **Man-in-the-middle attack** is feasible only if the attacker has physical access to the inside of the chip as well as the communication taking place between the secure test server and the chip.

A more detailed security analysis of the authentication protocol can be found in [12].

VI. Conclusion

We have presented a novel model for testing cryptographic SoCs by integrating authenticated scan-based testing through Test Wrappers and the pseudo-random Logic BIST. The security of the presented scheme is achieved by mutually testing public and symmetric-key cryptographic modules, which ensures the authenticity of the test server and the confidentiality of the key material. By re-using the public-key module to provide BIST infrastructure, we minimize the introduced area overhead. Furthermore, rather than adding new cryptographic primitives to check the authenticity of the testing server, the existing one is used. By doing so, we achieve the security of the testing scheme at the minimal cost.

Acknowledgment

This work was supported in part by KU Leuven-DBOF/08/047, by the Research Council KU Leuven: GOA TENSE (GOA/11/007), by the European Commission under contract number ICT-2007-216676 ECRYPT NoE phase II, by the Flemish Government FWO G.0550.12N and by the Hercules Foundation AKUL/11/19.

References

[1] IEEE, “Ieee standard testability method for embedded core-based integrated circuits,” in *IEEE Std 1500-2005*, 2005.

[2] B. Yang, K. Wu, and R. Karri, “Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard,” *International Test Conference*, pp. 339–344, 2004.

[3] M. Agrawal, S. Karmakar, D. Saha, and D. Mukhopadhyay, “Scan Based Side Channel Attacks on Stream Ciphers and Their Counter-Measures,” in *Progress in Cryptology - INDOCRYPT*, 2008, pp. 226–238.

[4] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede, “Security analysis of industrial test compression schemes,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 32, no. 12, pp. 1966–1977, 2013.

[5] J. DaRolt, G. D. Natale, M.-L. Flottes, and B. Rouzeyre, “A novel differential scan attack on advanced dft structures,” *ACM Trans. Design Autom. Electr. Syst.*, vol. 18, no. 4, p. 58, 2013.

[6] A. Das, B. Ege, S. Ghosh, L. Batina, and I. Verbauwhede, “Security analysis of industrial test compression schemes,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 32, no. 12, pp. 1966–1977, 2013.

[7] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, “Scan-based side-channel attack against rsa cryptosystems using scan signatures,” *IEICE Transactions*, vol. 93-A, no. 12, pp. 2481–2489, 2010.

[8] J. DaRolt, A. Das, G. D. Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “A new scan attack on rsa in presence of industrial countermeasures,” in *COSADE*, 2012, pp. 89–104.

[9] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki, “Scan-based attack against elliptic curve cryptosystems,” in *ASP-DAC*, 2010, pp. 407–412.

[10] J. DaRolt, A. Das, G. D. Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede, “A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures,” in *DFT*, 2012, pp. 43–48.

[11] A. Das, U. Kocabas, A.-R. Sadeghi, and I. Verbauwhede, “Puf-based secure test wrapper design for cryptographic soc testing,” in *Design Automation and Test in Europe (DATE)*, March 2012, pp. 866–869.

[12] A. Das, M. Knezevic, S. Seys, and I. Verbauwhede, “Challenge-response based secure test wrapper for testing cryptographic circuits,” in *IEEE European Test Symposium (ETS)*, May 2011.

[13] A. Schubert and W. Anheier, “On Random Pattern Testability of Cryptographic VLSI Cores,” in *Journal of Electronic testing: Theory and applications 16*. Kluwer Academic Publishers, 2000.

[14] R. L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[15] N. Koblitz, “Elliptic Curve Cryptosystem,” *Math. Comp.*, vol. 48, pp. 203–209, 1987.

[16] V. Miller, “Uses of Elliptic Curves in Cryptography,” in *Advances in Cryptology: Proceedings of CRYPTO’85*, ser. Lecture Notes in Computer Science, H. C. Williams, Ed., no. 218. Springer-Verlag, 1985, pp. 417–426.

[17] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.

[18] National Bureau of Standards, “Digital Signature Standard (DSS),” *Federal Information*, vol. Processing Standards Publication 186, May 1994.

[19] Cohen, Henri. and Frey, Gerhard. and Avanzi, Roberto., *Handbook of elliptic and hyperelliptic curve cryptography*. Chapman Hall/CRC, Boca Raton : , 2006 .

[20] J. Daemen and V. Rijmen, *The design of Rijndael*, ser. Information Security and Cryptography. Berlin: Springer-Verlag, 2002.

[21] National Bureau of Standards, “Data Encryption Standard,” *Federal Information* , vol. Processing Standards Publication 46, January 1977.

[22] National Institute of Standards and Technology, “Recommendation for the Triple Data Encryption Algorithm(TDEA) Block Cipher,” *Federal Information*, vol. Special Publication 800-67, May 2008.

[23] National Bureau of Standards, “Secure Hash Standard,” *Federal Information*, vol. Processing Standards Publication 180-1, April 1995.

[24] ——, “Secure Hash Standard,” *Federal Information*, vol. Processing Standards Publication 180-3, October 2008.

[25] D. Karaklajić, M. Knezević, and I. Verbauwhede, “Multiplier based Bulit-in Self-Test for Cryptographic Applications,” KU Leuven, Tech. Rep., 2010.

[26] F. Stroele, “Bit serial pattern generation and response compaction using arithmetic functions,” in *16th IEEE VLSI Test Symposium (VTS)*, 1998, pp. 78–84.

[27] J. Rajski and J. Tyszer, “Multiplicative window generators of pseudorandom test vectors,” in *European Design and Test Conference*, 1996, pp. 42–48.

[28] ——, “Accumulator-based compaction of test responses,” *IEEE Transactions on Computers*, vol. 42, no. 6, pp. 643–650, June 1993.

[29] L. Blum, M. Blum, and M. Shub, “A Simple Unpredictable Pseudo-Random Number Generator,” *SIAM Journal on Computing*, vol. 15, no. 2, pp. 364–383, 1986. [Online]. Available: http://link.aip.org/link/?SMJ/15/364/1

[30] M. Doulcier, M. L. Flottes, and B. Rouzeyre, “AES-Based BIST: Self-Test, Test Pattern Generation and Signature Analysis,” in *4th IEEE International Symposium on Electronic Design, Test and Applications*. IEEE, 2008.