Stream vs Block ciphers for scan encryption

Emanuele Valea, Mathieu Da Silva, Marie-Lise Flottes, Giorgio Di Natale, Bruno Rouzeyre
LIRMM (Université Montpellier - CNRS), Montpellier, France
{mdasilva,valea,flottes,dinatale,rouzeyre}@lirmm.fr

***Abstract*—Security in the Integrated Circuits (IC) domain is an important challenge, especially with regard to the side channel offered by test infrastructures. Test interfaces provide access to the internal states of the IC by means of the scan chains for testing and debugging purposes. In terms of security, however, scan chains are a potential source of leakage that can be exploited by attackers. A countermeasure against such attacks is to encrypt the data flowing through the scan chains. Two types of ciphers can be employed: stream ciphers or block ciphers. Both have pros and cons in terms of performance (footprint, impact on test activity) and security. In this paper, we present two solutions: one exploiting stream ciphers fulfilling security requirements, and another exploiting block ciphers. We draw a comparison between these two scan encryption countermeasures taking into account design cost functions and security properties.**

Keywords—Test and Security; Stream Cipher; Block Cipher; Scan Attacks Countermeasure; Scan Encryption

I. INTRODUCTION

The most common Design-for-Testability (DfT) technique is the use of scan chains. It consists in replacing original flip-flops (FFs) by so-called scan FFs organized in shift registers during the test phase. The internal state of the circuit can thus be controlled at the serial input of the scan chain, called Scan-In (SI), and it can be observed at the serial output of the scan chain, called Scan-Out (SO). In addition to the scan chains, several standards at system level have been proposed to facilitate the testing of chips. The standard for board testing is the IEEE 1149.1 [1], also known as JTAG, the IEEE 1500 [2] standard has then been proposed to deal with System-on-Chips (SoC), more recently the standard IEEE 1687 [3], also known as IJTAG, facilitates the access to test embedded instruments by defining a Reconfigurable Scan Network (RSN). The testing infrastructure of a chip forms a scan network, based on these DfT solutions.

The scan network provides full observation and control on the internal states of a device, ensuring its testability. On the other hand, the security is compromised by these control and observation facilities. An attacker can indeed exploit the scan network to steal secret information. For instance, scanning out the scan chain content of a crypto-processor after one-round execution in functional mode permits to retrieve part of the used secret key. Several attacks ([4], [5], [6] and [7]) on the Advanced Encryption Standard (AES) [8], called differential scan attacks, have been proposed in the literature.

Another threat coming from the test standards (JTAG, IEEE 1500 and IJTAG) concerns the Intellectual Property (IP) theft. The scan network has a direct access to the internal registers of an IP, giving information on the logic structure of the circuit. An attacker can exploit this information to reveal the IP design.

A threat within the device is also presented in [9], which consists in studying the case of a malicious device inserted in the JTAG daisy-chain. If a malicious device is implemented within the system, this one can capture sensitive data of other devices connected to the same daisy-chain, such as secret keys, configuration files or firmware sent through the JTAG interface.

Several countermeasures have been proposed to cope with these attacks. An industrial practice involves blowing fuses to disconnect the scan chains after manufacturing test. Another countermeasure is based on the Built-In Self-Test (BIST) [10]. This DfT approach limits the external control and observation on the scan chains, preventing scan attacks. The application of this technique is interesting on crypto-processors because they are generally easily testable with pseudo-random data [11][12]. However, these solutions compromise diagnostics, debugging or maintenance in the field.

Other countermeasures are based on test locking mechanisms. The so-called *secure test wrappers* guarantee that only authorized users can unlock the test access with a password [13] or through a secure protocol [14][15]. In addition to the area and test time overhead introduced with these solutions, a key management is required to share the test session keys with authorized users. These countermeasures rely on secure test interface to prevent any attack through IC ports but do not target malicious devices in the system.

Other architectures have been explored to hide the scan chain architecture by means of combinational or sequential functions. In [16], the scan chain is dynamically obfuscated with a Linear Feedback Shift Register (LFSR) and XOR gates inserted within the scan chains. The tester has to know the specific test procedure in order to scan-in the desired test data and observe readable scanned-out data. This solution impacts the design flow since the scan chains have to be replaced by the obfuscated ones at core level. In addition, this countermeasure is based on obfuscation, thus it relies on the assumption that the attacker has no way to get information on the scan chain implementation. Obfuscation is not considered as strong as encryption according to the Kerckhoff's doctrine.

A last family of protections relies on test communication encryption, which ensures the confidentiality of the exchanged messages between the circuit and the tester. The encryption provides protection against unauthorized users who want to communicate with the protected circuit, and attackers trying to intercept the communication. The encryption can be performed either with block ciphers or with stream ciphers. The choice must be driven by performance and security trade-offs. The preferred mechanism in the literature ([9], [17]–[19]) for encrypting the scan network is stream ciphering, because block ciphers present a larger area and have to be adapted to cope with the serial nature of the exchanged data with the ATE (Automatic Test Equipment). Nevertheless, as shown in Section IV, stream ciphers may introduce vulnerabilities. For this reason, the implementation of dedicated countermeasure introduce a larger costs.

In this paper, the main contribution is the evaluation of the two solutions: Scan Encryption based on Stream Cipher (SESC) not exposed to the state-of-the-art vulnerability, and Scan Encryption based on Block Cipher (SEBC). Security, implementation costs and impact on testability are presented and discussed.

The remainder of this paper is organized as follows. Section II presents the threat model that the countermeasures are intended to prevent. Section III gives a brief presentation of block ciphers and stream ciphers. Section IV summarizes the state-of-the-art protection based on encryption. Section V presents the vulnerabilities of both ciphers, and shows how to exploit them on the scan attack countermeasures. Section VI presents the two secure countermeasures: one based on stream ciphers, the other based on block ciphers. Section VII evaluates and compares both solutions in terms of security, implementation costs, test procedures and integration in SoC designs. Finally, Section VIII concludes the paper.

II. THREAT MODELS

This section describes the various testing structures that can be exploited to carry out attacks. The threat model is not limited to attacks targeting internal scan chains. The focus of this paper covers general scan structures where data can be written and read through a scan network, along the IC supply chain.

A. *Testing structures*

Scan chains are inserted within circuits in order to control and observe the internal states via the Scan-In (SI) and Scan-Out (SO) pins. Manufacturing test relies on probe-based testing, using dedicated test pads on the silicon die to access the scan chains. Increasing the number of internal scan chains enables parallel processing of test data. Hence, it produces significant test time savings.

Chip packaging moves away the external test pins accessible during manufacturing. The IEEE Std. 1149.1 [1], called JTAG, allows the tester to access chips on a board and directly communicate with them in a serial way, in order to run debug functions or to program reconfigurable hardware. Instead of probing, the board-level JTAG allows an external tester to access the scan chains via a four-signal Test Access Port (TAP).

In SoC design, cores are manufactured at the same time on a unique piece of silicon. Therefore, all cores must be scan-tested after the integration of the whole system. In order to save test time, parallel scan chains are preferred. The IEEE Std. 1500 [2] provides a very similar architecture to JTAG, with the same TAP. One of the main differences with respect to JTAG is that the test infrastructure may include optional parallel test pins to access multiple scan chains of the internal cores. As a consequence, test time is drastically reduced.

Fig. 1 shows a SoC with two cores, equipped with IEEE 1500 test wrappers, namely *Core 1* and *Core 2*. A 5-bit test bus enables parallel testing of these two cores with parallel access to their test I/Os (three-pin interface in *Core 1* and two-pin interface in *Core 2*). Note that *Core 2* is further equipped with decompressor/compressor devices enabling the delivery and the collection of test data to/from its three internal scan chains through only two pins.

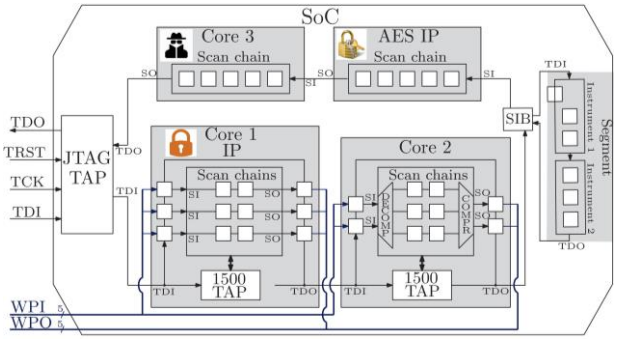

Fig. 1 Test infrastructure in a SoC: Core 1-2 with IEEE 1500 wrappers, an IJTAG SIB for including/excluding instruments 1-2, TAP controllers for system and core management.

In addition to JTAG and IEEE 1500, a recent standard has been set up to deal with the important number of embedded instruments. These instruments are used to control different pieces of hardware within a chip, such as temperature monitors (used to set a temperature sensor), or a memory BIST engine (used to control the self-testing of a memory). The IEEE Std. 1687 [3], called IJTAG, provides access to these instruments through a Reconfigurable Scan Network (RSN). It defines the Segment Insertion Bit (SIB) that dynamically configures the IJTAG RSN. Selecting one SIB activates an RSN segment. As a consequence, the instrument(s) on that specific segment of the scan path is activated. Conversely, de-selecting the SIB deactivates that portion of the RSN and renders the instruments on that segment inaccessible. In Fig. 1, one SIB is used to activate/deactivate a segment composed of two instruments (*Instrument 1* and *Instrument 2*).

We use the generic test infrastructure depicted in Fig. 1 to further define the threat model. This test structure combines on-core scan chains, a JTAG interface, IEEE 1500 wrappers and an IJTAG SIB. Everything is integrated inside the same SoC. An external tester can access the test structures of the cores via the JTAG interface. All the cores and instruments are connected together through a test daisy-chain. The scan chains of *Core 1* and *Core 2* are accessible through the IEEE 1500 test wrappers. The two instruments are accessible through the IJTAG RSN. The scan chains of the AES IP and of *Core 3* are directly accessible via the test daisy-chain.

B. *Possible attacks through the scan network*

The controllability and the observability offered by the scan network can be exploited to steal critical information or to disturb the system operations. For instance, an external attacker can discover the secret key of a crypto-processor (e.g. the AES IP in Fig. 1), or steal information on an IP design (e.g. *Core 1* in Fig. 1). The only requirement to perform such attacks is the free access to the test interface. Another threat can come from internal components. For instance, cores provided by untrusted third-parties, inserted in the system and connected to the test daisy-chain (e.g. *Core 3* in Fig. 1) can represent a threat.

1) *Scan attacks*

The first scan attack on an AES co-processor has been described in [4]. Its objective is to retrieve the encryption secret key. AES executes several rounds of operations (e.g. substitutions and permutations) resulting on confusion and diffusion of the plaintext data on the ciphertext. The scan attack targets the result of the first round, when data stored

into the round register are partially encrypted. The attack procedure consists at first in switching the circuit to test mode after the first round. After that, the partially encrypted result, stored in the round register, is scanned out. The attacker carries out the differential scan attack by applying plaintext pairs on the AES inputs, and then calculating the Hamming distance between the two results. According to the Hamming distance value, the attacker can identify with certainty one key byte. The attack strategy is thus to try different plaintext pairs until the difference between two intermediate results permits the attacker to determine a key byte. The attacker repeats these steps for every key byte, in order to retrieve the whole key. In average, 512 plaintexts allow the attacker to retrieve the 128-bit AES key.

The described scan attack deals with a single scan chain. However, the use of advanced DfT structures, such as multiple scan chains, on-chip test data decompression, scan response compaction, X-masking and partial scan, has been widely adopted. When advanced DfT approaches are adopted, the entire round register of the crypto-processor is not necessarily directly observable at scan output. This makes the execution of the scan attack, as presented in [4], very difficult. Improved scan attacks have thus been proposed in [5] and [6] that deal with this issue.

In [7], a *test-mode-only* attack has been proposed in order to cope with the *chosen plaintext* strategy described in [4]. In this attack, plaintexts are applied by scanned-in bit streams. The major challenge in this attack is to figure out which scan flip-flop corresponds to which input bit of the AES. Once the attacker has this knowledge, he or she proceeds, as in [5] and [6], without switching from test mode to normal mode. This permits to circumvent simple countermeasures, such as resetting the round register when the circuit switches from normal mode to test mode. However, this attack assumes that the key used for encryption in test mode is the same as in normal mode.

2) Design analysis using scan side channel

The scan network can also be used to obtain some information on the sequential and combinational functions of a design. In the studied threat model, let us consider that *Core 1* in Fig. 1 is an IP core that holds secret information about its design. The direct access to the internal registers of the IP core allows an attacker to analyse the logic structures of the design. He or she can shift desired data in the scan network in order to analyse the resulting outputs. Therefore, the exploitation of the scan network provides information on the design, which helps to reverse engineer the IP cores.

3) Malicious IP inserted in the test daisy-chain

In the two previously presented threats, the attacker is external to the chip and gets benefit from the controllability and the observability offered by the test infrastructure. Another potential threat is the introduction of a malicious element in the integrated system or in the board. The malicious IP core may be able to spy and modify the test communication. This threat is further described in [9], where the depicted scenario involves several devices connected to the same JTAG network. This threat can be directly extended to scan chains and other test standard networks.

Let us consider a malicious IP core inserted in a test daisy-chain, for instance *Core 3* in Fig. 1. This core may steal confidential data sent by the user through the test interface. For example, when the device is configured, the malicious IP core can intercept the critical configuration data (e.g. the firmware of a microprocessor). The malicious core may also modify the data exchanged between a target core and the tester, in order to force the device into an illegal behavior, or to send fake test responses to the tester. This can prevent the tester from detecting possible malfunctions in the circuit, leading to an unexpected system failure.

C. Threats throughout the IC supply chain

Mentioned threats can be present at every stage of the IC supply chain. Fig. 2 presents the actors of the semiconductor supply chain and the threats they represent, based on the testing procedures that are performed at each stage.

The first actor is the IC designer who is in charge of the DfT (i.e. scan chains definition, test standard interfaces, test pattern generation). At this level, the circuit is not yet fabricated, therefore no threat is considered. During the generation of the IC layout, several third-parties IP cores can be inserted into the design, representing a potential target for attackers.

After the definition of the circuit layout, the design is sent to a foundry in order to produce several IC samples. Before wafer slicing, all dies are independently tested, in series or in parallel, depending on the available ATE. No threat is considered at this stage. The foundry is regarded as trustworthy and the potential malicious cores are not able to carry out attacks from this stage. Scan attacks on crypto-cores would be useless at this stage, because the attacker could only steal the key used during manufacturing test. The key used in mission mode is configured at a later stage.

In the assembly stage the circuit is packaged. When the SoC is mounted on the board, the considered threats concern

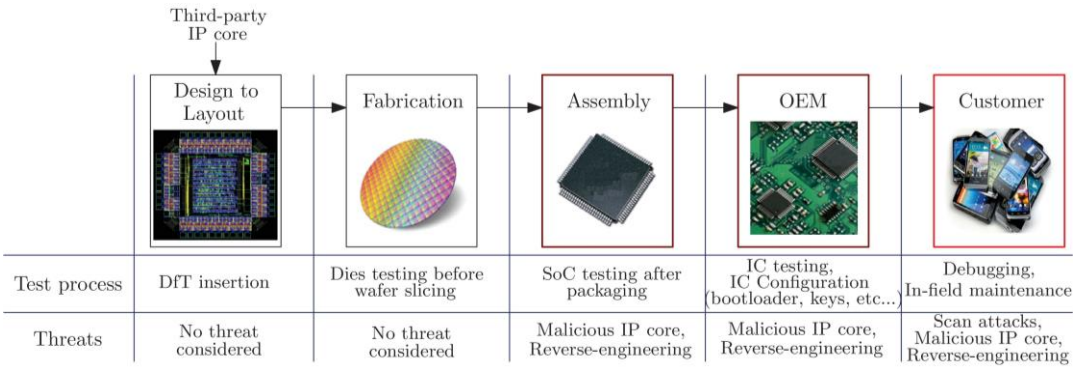

Fig. 2 IC supply chain with threats of the different actors using the test facilities

the IP cores (e.g. *Core 1* in Fig.1, susceptible target of reverse engineering), and internal malicious cores (e.g. *Core 3* in Fig.1) inserted in the scan network at system level. Concerning scan attacks, the key of the crypto-core is still the test key, which does not represent a meaningful secret to steal.

The next stage consists in mounting the IC on the board by the Original Equipment Manufacturer (OEM). The OEM performs the test of the IC and sets its configuration. The configuration goes through loading the bootloader and the firmware in the memory, configuring the crypto-processor with the key used in mission mode and configuring the microprocessors. The threats at this stage are the same as during the assembly stage. The secret key of the crypto-core is configured by the OEM, therefore scan attacks do not represent a threat at this stage.

The final stage sees the device owned by the final user. The test interface of the device can be used to debug the system or to perform in-field maintenance. All the studied attacks represent a threat at this stage. A malicious user can exploit the scan chains to steal the secret key configured by the OEM.

III. Cryptographic Primitives for Encryption

For the sake of completeness, we provide in this section a brief reminder about the cryptographic primitives, called *ciphers*, which underlie the encryption techniques discussed in the paper. We focus on *symmetric* ciphers since their operations (same key used for decryption/encryption) fit with the encryption of the test communication. Moreover, symmetric ciphers propose a lower cost in terms of area and computation time than asymmetric ciphers.

First, we recap the rationale of symmetric data encryption. We then provide a brief introduction on block and stream ciphers in order to set the terminology and highlight the key features that are needed to appreciate the vulnerabilities explained in Section V.

In general, a cipher allows the *sender* to transform an input message *m* (plaintext) in a ciphered version *c* using a secret key *k*. The *receiver* needs to be able to rebuild *m* from *c* upon knowledge of the same *k* (or derived from *k*).

A cipher is composed of two functions: **E**, called *encryption* function, and **D**, called *decryption* function, such that:

- The encryption algorithm takes as input the message *m* and the secret key *k*, and outputs a ciphertext *c*, so that $E(k, m) = c$.
- The decryption algorithm takes as input the ciphertext *c* and the secret key *k*, and outputs the plaintext *m*, so that $D(k, c) = m$.

The encryption of a message followed by the decryption of the correspondent ciphertext must result in the initial message, i.e. $D\big(k, E(k, m)\big) = m$. Ciphers that are used for providing confidentiality in the test infrastructures are *stream ciphers* and *block ciphers*.

The main difference between stream and block ciphers relies on the size of the data that are processed in each encryption. Stream ciphers encrypt one bit at a time from a bitstream; this results in the encrypted message having a bit-to-bit correspondence with the plaintext message. Differently, block ciphers take as input an *n*-bit block of the plaintext, which is encrypted in an *n*-bit block ciphertext; in this case, the properties of the plaintext are dispersed on the whole *n* bits of the ciphertext.

A. Stream ciphers

Stream ciphers are based on a theoretical cipher, called *One Time Pad* (OTP). In the OTP, the secret key must be as long as the message *m*. The encryption function is defined as $E(k, m) = m \oplus k$, and the decryption function as $D(k, c) = c \oplus k$. If *k* is perfectly random (i.e. according to the uniform distribution), the OTP has *perfect secrecy*. This means that the produced ciphertext is indistinguishable from a random sequence (this is due to the properties of the XOR operator). In this case, it is impossible for an attacker that intercepts the ciphertext to derive any information neither on the message nor on the key. However, from a practical point of view, the OTP is not implementable because of the key length.

Stream ciphers are an implementation of the OTP. Instead of XORing a random key k as long as the plaintext, a *Pseudo-Random Generator* (PRG) generates a pseudo-random sequence of bits called *keystream*. The PRG takes as input a value k, called *seed* of the stream cipher, and outputs the keystream $S(k)$. The encryption and decryption functions are thus defined as $E(k, m) = m \oplus S(k)$ and $D(k, c) = c \oplus S(k)$.

As far as the PRG produces a keystream that is unpredictable, the resulting stream cipher is considered to be secure. As shown in Section V, it is also important that a given keystream is not used twice.

Because of its low cost implementation, the TRIVIUM [20] stream cipher is widely used in the context of scan chain protection. It is based on a Non-Linear Feedback Shift Register (NLFSR) used as PRG. The seed of the TRIVIUM PRG is made by an 80-bit secret key *K*, and an 80-bit Initialization Value (*IV*), which is publicly known. The generated keystream is denoted as $S(K, IV)$.

B. Block ciphers

Block ciphers are based on mathematical objects called *Pseudo Random Permutations* (PRP). They are invertible functions that take as input an *n*-bit value *m* and a secret key *k*, and output an *n*-bit value *c*. A PRP is considered secure if, fixed with a key *k*, the resulting function is indistinguishable from a random bijective function on *n*-bit values.

Block ciphers implement a secure PRP. They are made of an encryption function that is able to encrypt a plaintext block into a ciphertext block using a secret key; and a decryption function that performs the inverse operation and retrieve the plaintext block from the ciphertext.

The most used block cipher is the AES. Other algorithms have been proposed to be more lightweight, i.e. with a lower cost in terms of area and power consumption, such as PRESENT [21] or SKINNY [22].

IV. State-of-the-art countermeasures based on test communication encryption

Many solutions have been proposed in order to guarantee the confidentiality of communications within test infrastructures. Solutions proposed so far rely on a modified interface of the test infrastructure that combines both test data transmission and encryption. Fig. 3 presents the core scheme of countermeasures based on test communication encryption.

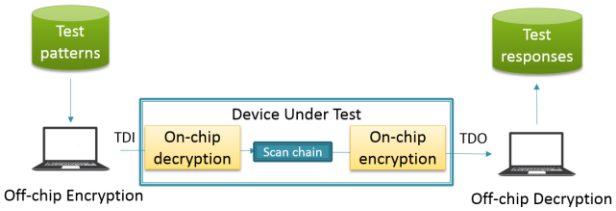

Fig. 3 Basic scheme of the test communication encryption

Test data is first encrypted off-chip and stored into the test equipment. At test time, encrypted test vectors are sent to the target device, which decrypts them on-chip using the encryption key, and performs test operations. Before scanning out a test response, the data is encrypted on-chip by the device under test. The tester collects encrypted responses and decrypts them off-chip using the encryption key for further comparison with the expected data. Test data, vectors and responses, are thus kept confidential during the testing process. Without knowing the key, there is neither a possibility to control the device to a specific state nor the opportunities to observe the device state. Encrypted test data can thus flow safely through the entire system containing the device under test without risking to be read or written by an unauthorized third party.

Since data in test infrastructures is transmitted serially, most of the proposed test data encryption schemes are based on stream ciphers. In [9], the authors propose the use of the TRIVIUM stream cipher in order to encrypt the content of the JTAG communication. The TRIVIUM stream cipher is seeded with an *IV* and a secret key. The *IV* is hardwired in the device with fuses that are programmed at manufacturing time. The secret key is derived from a challenge sent by the user; the challenge is hashed inside the device exploiting the initialization of the TRIVIUM cipher itself; the response of this hashing procedure is the secret key used for the encryption. An authorized user knows the response to any challenge while an unauthorized user, without the knowledge of the challenge/response pairs, cannot have the secret key used for test data encryption.

To improve the integrity, the solution presented in [9] also proposes to use a HMAC signature appended to the test messages. While the TRIVIUM-based encryption prevents unauthorized user to write a chosen plaintext in the scan chain, the HMAC signature prevents any *write* operation missing the correct signature.

The SESC technique proposed in [17] addresses the threat posed by untrustworthy cores in SoCs. It eliminates the risk of a malicious core sniffing test data. The countermeasure consists in encrypting test vectors using the TRIVIUM stream cipher. For that purpose, the tester generates a random key for the TRIVIUM cipher and shifts it to the core under test via a dedicated scan chain, non-visible from the other cores. The *IV* setup methodology is not described by the authors.

Stream ciphers are also used on IJTAG reconfigurable scan networks (RSN) in [18]. A TRIVIUM cipher encrypts and decrypts the data shifted in and out of the RSN. The goal is to protect items under test against malicious embedded instruments sniffing the communication, and against external attackers who want to illegally use the embedded instruments. The authors deal with key and *IV* management by proposing a unique set of keys and *IV* for each device. The proposed implementation is either with fuses or Physical Unclonable Functions (PUFs). The authors do not further discuss the usage of key or *IV* values between two encryption sessions.

All the SESC techniques that have been presented share a vulnerability that is based on the incorrect management of the stream cipher seed (the *IV* and the secret key). In the next section, we expose the foundation of this vulnerability, and we explain how to exploit it to attack the aforementioned solutions.

V. CIPHERS LIMITATIONS AND VULNERABILITIES

A. *Stream cipher limitations*

Stream cipher security relies on the implementation of the PRG. Stream ciphers are secure as far as the PRG produces a keystream that is unpredictable. However, they have intrinsic weaknesses that facilitate attacks when they are used incorrectly.

An important requirement for the security of the stream cipher is that the seed k must be used only once. In the opposite case, a simple attack can be performed, which is called *two-time pad*. When the same seed k is used to encrypt two different messages m_1 and m_2, the two bitstreams are equal. Thus: $c_1 = E(k, m_1) = S(k) \oplus m_1$ and $c_2 = E(k, m_2) = S(k) \oplus m_2$. This leads to:

$$c_1 \oplus c_2 = (S(k) \oplus m_1) \oplus (S(k) \oplus m_2) = m_1 \oplus m_2$$

The attacker can exploit the XOR of two messages for a differential attack that consists in obtaining confidential information from the difference between messages.

The *two-time pad* vulnerability can be exploited in the protections based on stream cipher presented in the literature so far. Let us consider a protection on an AES IP, such as the one implemented in the SoC in Fig. 1. The protection is based on the encryption of the test communication with a stream cipher whose secret key is unknown from the attacker. The responses of the AES core are thus encrypted with a stream cipher. We carry out the differential scan attack on the protected AES IP.

The structure of the encryption with stream cipher is shown in Fig. 4. The stream cipher produces the keystream $S(K, IV)$ from a key K and an initial value IV, and encrypts the data XORing them with the keystream. Let us perform a differential scan attack and let R_1 and R_2 be the two responses. For response R_1, the stream cipher generates a keystream $S(K, IV)$ from the key K and the initial value IV. Then, after a reset of the circuit, the second response R_2 is encrypted with the same keystream $S(K, IV)$. By applying the differential scan attack between the two encrypted responses, we obtain $[R_1 \oplus S(K, IV)] \oplus [R_2 \oplus S(K, IV)] = R_1 \oplus R_2$, removing the stream cipher encryption. Therefore, an attacker can carry out the differential scan attack, even if the test responses are encrypted.

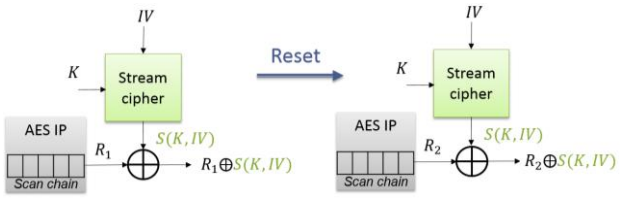

Fig. 4 *Two-time pad* exploited for the differential scan attack

The countermeasures presented in [9], [17] and [18] are all exposed to this vulnerability because the *IV* and the key are not managed properly (i.e. they do not change between two encryptions). In [9], the *IV* is hard-coded with fuses, consequently being the same for each generated keystream. The key is the response to a challenge sent by the user. An attacker needs to send the same challenge in order to generate the same keystream to encrypt different test data. Concerning the solution proposed in [17], the key is set by the user. Therefore, an attacker can send the same key to the circuit in order to generate the same keystream for different encryptions. In [18], the authors evoke the use of a unique set of keys and *IVs* for each protected instrument in the IJTAG network. However, the re-use of the same sets of keys and *IV*s to encrypt test data shifted through a protected instrument leads to the presented vulnerability.

The same seed must not be used more than once to avoid the generation of the same keystream for several encryptions. In order to protect against differential scan attacks, the stream cipher encryption needs to have different *IV*s and/or keys for each generated keystream.

B. *Block cipher limitations*

Block ciphers are stronger primitives than stream ciphers. Differently from stream ciphers, they allow to perform encryptions of different plaintexts with the same secret key, without lacking of security. Depending on the implementation mode, an attacker may have the ability to identify that identical plaintexts have been processed, or to provide correct ciphertexts for a given (potentially unknown) plaintext, without knowledge of the key.

Several modes of implementation exist for the block ciphers in order to prevent replay attacks. The simplest one is the *electronic codebook* (ECB) mode, in which data are divided into blocks, which are then encrypted separately with the same key. Another mode is the *cipher block chaining* (CBC), in which the encryption of a plaintext block depends on both the key and all the ciphertext blocks that have been processed up to that point.

The ECB implementation presents the limitation related to the identification by an attacker of identical ciphered blocks corresponding to the same plaintext block. Conversely, CBC mode does not allow an attacker to identify repeated data blocks in the encrypted communication since each encrypted block of data depends on the previous encryptions.

Applied to the encryption of the test communication, differential scan attacks are ineffective no matter which block cipher encryption mode is employed. However, ECB mode gives the possibility for an attacker to identify bits of interest in the encrypted test data. Let assume that data scanned out from the device includes round-register bits of a crypto-processor under test. Using two different plaintexts to exercise the crypto-processor will result in two different round-register values while other data stored in the scan chain will remain the same. After the encryption of the scanned out data in ECB mode, the data that did not change were encrypted in the same way (i.e. resulting in the same ciphertext). On the other hand, round register data, which differ due to the application of two different plaintexts, result in two different encrypted blocks, revealing the bits of interest in the scan chain.

Consequently, the application on a crypto-processor of two different plaintexts could allow an attacker to identify the encrypted segments in the scan chain containing at least one bit of the round register of a crypto-processor under attack. In the case where such information could allow the attacker to carry out a new attack, CBC mode is a more secure implementation.

VI. SCAN ENCRYPTION COUNTERMEASURES

In this section, we first present a SESC [19] implementation that does not present the vulnerability of the state-of-the-art solutions discussed so far (see Section V.A). After that, we describe a SEBC [23] implementation using lightweight block ciphers.

Fig. 5(a) details the basic scheme of both countermeasures. Two encryption schemes are implemented at the scan pins. The input scan ciphering process is in charge of processing the on-chip decryption of the test patterns, the output scan cipher is in charge of processing the on-chip encryption of the test responses before scan-out.

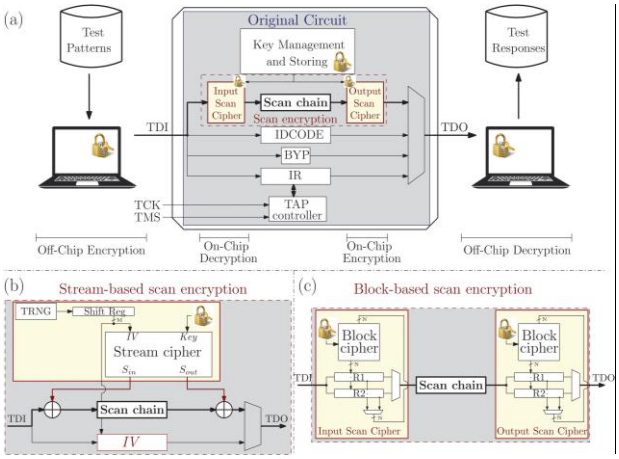

Fig. 5 (a) Global architecture of the scan encryption countermeasures. (b) SESC implementation. (c) SEBC implementation.

Since the scan attack countermeasure relies on test data encryption, the process must also provide a scheme for test-key management, i.e. a way to securely share the secret key between the test equipment and the device under test. Therefore, it requires the integration of a Secret Key Management Unit (SKMU) mechanism within the chip, in addition to the ciphers.

Assuming that a device must be protected from scan attacks because of embedded crypto-processors, we propose to re-use the secure storage containing all the secret keys and a SKMU, in order not to introduce issues about key management. The key is securely stored in the secure storage of the crypto-core, and managed with the SKMU in order to share the secret only with authorized users. The SKMU also performs the operations of key generation, activation and revocation during the life cycle of the scan encryption key.

A. *Stream-based scan encryption*

The present SESC countermeasure has been proposed in [19]. Compared to previous solutions, the *IV* value is randomly generated by a True Random Number Generator (TRNG) at every circuit reset, thus preventing *two-time pad* attacks. The use of a TRNG guarantees to not re-use the same *IV* for the keystream generation. Moreover, the TRNG guarantees the generation of a different *IV* across circuit

resets. This requirement precludes the possibility to employ other memory-based techniques (e.g. counters), which would impose to keep track of the already-used *IV*s.

The on-chip generated random value *IV* has to be known by the external tester/debugger, in order to allow a correct communication between the tester and the device. To make the random *IV* value accessible, an extra test data register is used to store its value. A custom instruction, *GETIV*, is added to the JTAG instruction set in order to read the content of this register. The on-chip generated *IV* is then accessible to the external world. It is important to note that the security of the stream cipher is not jeopardized by making the *IV* public, since its key is still kept secret.

The implementation is illustrated in Fig. 5(b). The stream cipher generates two keystreams: (i) S_{in} for decrypting the test data shifted in the scan-in port; (ii) S_{out} for encrypting the test data shifted out through the scan-out port. A single stream cipher does not generate the same keystream for both input decryption and output encryption in order to avoid any temporal correlation in the generated output bitstream. The generation of independent keystreams can be done in an efficient way for some standard stream ciphers. For instance, the TRIVIUM requires very few additional logic gates (i.e., 3 AND gates and 11 XOR gates) to generate 2 streams at the same time compared to its standard implementation with a unique stream.

The SESC operates in two phases: the initialization phase, followed by the encryption phase. In the initialization phase, the TRNG is firstly initialized in order to reach a sufficient entropy for generating a random value. Once the TRNG is initialized, the random bitstream is generated and stored in a shift register. The shift register content is then used as *IV* for the stream cipher. Finally, the stream cipher has a setup phase before becoming operational. During the whole initialization process, the scan chain is not accessible for an external user, since the TAP controller remains set on bypass mode. The initialization process is completed when the stream cipher setup ends. The stream cipher then generates the keystreams while the TAP controller is in the shift state.

Beforehand, the external user has to execute the custom instruction *GETIV*. The execution of this instruction involves copying the content of the shift register into the specific test data register *IV*, and connecting the test data register to the TDI/TDO ports. This way, the user can shift out the *IV* value that has been produced by the TRNG. Once the user knows both the secret key and the *IV*, it is possible for him or her to encrypt off-chip the test patterns sent to the circuit, and to decrypt off-chip the test responses obtained from the circuit.

B. Block-based scan encryption

The present SEBC technique has been proposed in [23]. It relies on the test communication encryption with lightweight block ciphers. As shown in Fig. 5(c), two block ciphers are implemented, one for the decryption performed at scan input, another for the encryption performed at scan output. Each of these two ciphers have two round registers (R1 and R2) with two operating modes, *parallel load* and *serial shift*. The *parallel load* is used to perform the encryption/decryption operations, while the *serial shift* is used to acquire the data shifted through the scan chain. The two operating modes are performed in parallel in order not to waste test time.

In this paper, differently from [23], we have implemented the block ciphers in CBC mode in order to propose a more secure implementation, as explained in Section V.B, regarding the possible identification of repeated test blocks in the test sequence.

Test data are acquired serially by the scan chain. The use of block ciphers implies thus padding test data into the block size. The scan chain is filled/flushed segment by segment, each segment consisting of the block size on *N* bits. When the scan chain length is not a multiple of the block size, the solution still works, but each test pattern is padded with extra data. The tester has thus to complete the shift operations employing additional clock cycles, resulting in a test time overhead on each test pattern.

Formally, by considering a circuit having $F = SN + R$ flip-flops, where F is the total number of flip-flops in the original circuit, S is the number of N-bit segments, and $R = F \bmod N$, the test time overhead $T_{overhead}$ introduced by the SEBC on K test patterns is equal to:

$$T_{overhead} = \begin{cases} 2 \cdot 2N & if\ R = 0 \\ 2 \cdot 2N + (N - R)(K + 1) & if\ R > 0 \end{cases} \quad (1)$$

This test time overhead is evaluated in Section VII considering several circuits. The authors in [23] propose a solution not to waste the additional clock cycles used to synchronize the scan chain encryption scheme with constant segment length N. These $N - R$ extra clock cycles are actually exploited for testability improvement without requiring any additional test time. Indeed, by adding $N - R$ dummy scan flip-flops to the original scan chain, these extra flip-flops can be used as observation points to the circuit logic. The observation points permit to improve the propagation of test responses to the scan flip-flops, without any impact on the test time. The goal of the observation point insertion is to reduce the test sequence length, i.e., reducing the number K of test patterns for the same fault coverage. We present the results of the optimisation in Section VII showing that the additional flip-flops, used as observation points, can compensate the additional shift operations required by the SEBC.

VII. Evaluation and comparison

In this section, we evaluate the scan encryption countermeasures in terms of security, implementation costs and integration in a SoC design. Moreover, we compare both SESC and SEBC implementations.

A. Security analysis

The scan encryption countermeasures are evaluated regarding the threat model described in Section II. Both solutions, SESC and SEBC, are considered.

An attacker cannot carry out scan attacks, reverse engineer an IP using scan side channel, or exploit a malicious core inside the device, since the scan network is encrypted. The decryption performed on scanning-in test data prevents the setup of desired values in the scan chain without knowing the secret key. The encryption performed on the scanning-out test responses prevents the observation of the internal states of the circuit under test without firstly performing the decryption. The controller of the scan encryption enables the stream cipher as soon as a scan operation is required to prevent any clear bitstream from being inserted or observed.

The SESC technique proposed in [19] does not present the same vulnerability as the previous countermeasures in [9], [17] and [18] concerning the differential scan attacks. The seed used to initialize the stream cipher is randomly generated, consequently encrypting with a different keystream at each reset of the circuit. Therefore, differential scan attacks are not possible. Concerning SEBC, it does not present the vulnerabilities of state-of-the-art SESC techniques, since the countermeasure is based on block ciphers, consequently being a more secure solution.

Both SEBC and SESC prevent the exploitation of the scan chain that would help to reverse engineer an IP. In FPGA, the protection of the critical information on IP design contained in the bitstream is achieved through data encryption [24]. In the same manner as the proposed countermeasure, the decryption at scan input does not allow to analyse the IP behaviour after the scan-in of desired data. The encryption at scan output prevents scanning out critical data related to the IP design.

In terms of security, SESC and SEBC both protect against the considered threat model.

B. Implementation costs

First, we compare the solutions evaluating the protection of some benchmarks, namely a triple-DES, a Pipelined 128-bit AES, a Pipelined 256-bit AES, a 1024-bit RSA and a LEON3 processor. In these first experiments, all the circuits are equipped with a single scan chain.

These benchmarks have been synthesized using a 65nm technology library, as well as the ciphers. We have implemented the SESC with the TRIVIUM stream cipher, and two block ciphers in *counter mode* (CTR): PRESENT [21] and AES [8]. Block ciphers in CTR mode operate like stream ciphers. For SEBC, we have used the PRESENT and the SKINNY [22] block ciphers in CBC mode.

Concerning the SESC, we do not consider in the experimental results the cost of the TRNG. In the case where a TRNG is already implemented for the functional mode of the original circuit, the proposed countermeasure can exploit this TRNG during the test mode, implying no overhead for the random number generation. If a TRNG has to be implemented, the related area cost is evaluated to be 15,000 GE from the Synopsys DesignWare IP library [25]. This values is equivalent to 31,200µm² on the 65nm technology library adopted for the experiments.

1) Area and test time overheads

Area overheads are reported in Tab. 1 for both SEBC and SESC solutions. It must be noted that the SEBC requires the implementation of two ciphers, one for decrypting input test patterns, the other one for encrypting test responses. The SESC requires the implementation of only one cipher to deliver two keystreams. The overhead is to be compared with the original circuit under test. As it can be seen, both solutions are expensive in terms of area, making these solutions suitable to large designs only. Furthermore, in our experiments, we have not considered the key management, using the one adopted in the benchmark under test. Furthermore, this solution is particularly adequate when at least one crypto-core is implemented in the circuit under test.

Concerning test time, we consider for each IP core the test sequence for detecting stuck-at faults, obtained with an ATPG (Automatic Test Pattern Generator). Tab. 1 reports the test time overhead for the scan encryption countermeasures. The test time overhead for SESC is due to the initialization phase: the TRNG initialization, the shift of the random IV, and the stream cipher setup. Without considering the TRNG initialization, we have respectively 138, 95 and 1232 clock cycles for AES-128 CTR, PRESENT-128 CTR, and TRIVIUM. In any case, they represent a marginal cost.

For SEBC, every pattern has to be padded in such a way that its total length is a multiple of the block size of the cipher, i.e. 64 bits in the case of PRESENT and SKINNY. This induces a test time overhead for each pattern. This overhead can be reduced thanks to the optimization presented in Section VI.B. The DfT tool Tetramax is used for the selection of observation points in the circuit logic and for test pattern generation. We have constrained the tool to use only the $N - R$ extra flip-flops for testability improvement i.e. for the scan chain length to be a multiple of 64. After selection, observation points drive extra XOR trees ending on the

Original circuit		Triple-DES		Pipelined AES-128		Pipelined AES-256		RSA 1024		LEON3	
		Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)	Area (µm²)	Test time* (clock cycles)
		187,494	687,101	367,926	1,944,877	669,193	4,559,845	468,415	39,405,239	1,902,095	11,612,051
Scan Encryption	Area (µm²)	Overhead (%)		Overhead (%)		Overhead (%)		Overhead (%)		Overhead (%)	
SESC ***Error! Reference source not found.*** *(without TRNG implementation)*											
AES-128 CTR	48,118.20	+25.66	+0.020	+13.08	+0.007	+7.19	+0.003	+10.27	+0.0004	+2.53	+0.001
PRESENT-128 CTR	6,833.84	+3.64	+0.014	+1.86	+0.005	+1.02	+0.002	+1.46	+0.0002	+0.36	+0.0008
TRIVIUM	5,408.52	+2.88	+0.18	+1.47	+0.06	+0.81	+0.03	+1.15	+0.003	+0.28	+0.01
SEBC ***Error! Reference source not found.***											
PRESENT-128 CBC	10,915.84	+5.82	+0.31	+2.97	+0.81	+1.63	+0.006	+2.33	+0.33	+0.57	+0.004
Insertion of observation points on original circuit		+5.95	+0.038**	+3.48	+0.013**			+2.44	+0.001**	+0.57	+0.002**
SKINNY-64-128 CBC	10,172.24	+5.43	+0.31	+2.76	+0.81	+1.52	+0.006	+2.17	+0.33	+0.53	+0.004
Insertion of observation points on original circuit		+5.55	+0.038**	+3.27	+0.013**			+2.28	+0.001**	+0.54	+0.002**

Tab. 1 Area and test time cost of scan encryption with stream cipher and block cipher

*: test time considered for a fault coverage of 100% on the original circuit, except for the LEON3 processor where the fault coverage reaches 70%

**: test time overhead for the block-based solutions compared to the test time of the optimized circuit (obtained with the insertion of observation points)

proposed extra flip-flops, thus allowing their observation at test time. For instance, the Triple-DES core has 8808=137×64+40 flip-flops, implying an extra test time (64-40=24) for the scan encryption technique to pad on 64-bit segments. By adding 24 scan flip-flops in the scan chain connected to observation points, the test time cost has an overhead of 0.038%, compared to 0.31% for the solution without the optimization. Tab. 1 presents the results of the optimization performed on the other circuits, except for the Pipelined AES-256, whose scan chain is a multiple of 64 (12736=199×64 flip-flops).

It is clear that SESC outperforms SEBC due to the data volume-dependant overhead of the last one. Except when the optimization of SEBC is applied on the original circuit, the test time overhead is reduced, representing a marginal cost.

2) Testability evaluation

The scan encryption permits to test the original circuit without reducing the test coverage, since the original test patterns are applied as they are. However, the architecture of the scan encryption solution must also be tested, without the help of scan chains that would expose the cipher to scan attacks. We propose to functionally test the cipher using test patterns dedicated to the circuit under test.

The test of block ciphers, such as PRESENT, SKINNY and AES, is facilitated by the diffusion properties of the crypto-algorithms, as described in [11] and [12]. Stream ciphers based on shift registers, such as TRIVIUM, are also easily testable since all the states of the stream cipher are shifted out the circuit as keystream. Therefore, both ciphers easily propagate the possible errors to the circuit outputs when an encryption is performed. To validate the assumption, we have evaluated the test coverage on every studied cipher (TRIVIUM, PRESENT, SKINNY and AES) by applying the test sequence of the original circuits. At scan-input, test patterns are processed by the input scan cipher, and the test responses are processed by the output scan cipher. We have performed experiments with the test sequence of the Pipelined AES-256, Triple-DES, Pipelined AES-128, RSA 1024 and LEON3 processor cores. In all cases, the fault coverage for stuck-at faults in the scan encryption is 100%. In other words, the ciphers are tested for free, with no additional test patterns compared to those used for testing the original circuit.

Concerning the test procedure, as described in Section IV, the tester has to encrypt off-chip the test patterns and to decrypt off-chip the test responses. The off-chip decryption is performed in order to compare with the expected plaintext responses, but this computation is not always necessary depending on the choice between SEBC and SESC. In fact, expected test responses computed in simulation by the ATE can have unknown values (X-values), describing an unknown binary state in the test responses obtained on the real tested circuit. These unknown bits are then ignored during the comparison with the obtained test responses.

Due to the confusion and diffusion properties of block cipher, the SEBC spreads the unknown bits in the entire encrypted test response. For this reason, the off-chip decryption for SEBC is mandatory in order to perform the comparison with the plaintext responses.

In case of SESC, the unknown bits are at the same position in the plaintext responses as in the encrypted test responses, since the encryption operates bitwise. Therefore, the unknown bits can directly be ignored on the encrypted test responses, avoiding the need for the tester to decrypt every obtained test response from the circuit. The comparison is thus performed between the encrypted obtained responses and the encrypted expected ones.

C. Integration in a SoC design

The integration on the scan encryption countermeasures just consists in adding ciphers at the input and the output of the scan chain. In the case of SESC, it also implies some modifications on the JTAG test wrapper, but not on the core itself. Therefore, the solutions can be applied without modification on the protected core.

Test wrappers of the cores composing a SoC are often connected in a test daisy-chain, as shown in Fig. 1. The serial input/output of the test interface provides access to the scan chains for all cores implemented in the SoC. When the scan encryption is implemented on one core, all the cores included in the test daisy-chain receive the encrypted data, providing protection against malicious cores. The stream cipher encryption operates bitwise on data, implying no issue on the integration of the SESC in a test daisy-chain. Contrarily, the block cipher encrypts blocks of data, implying to pad the test data to a multiple of the block size. The extra data used for padding is thus sent to the other cores in the test daisy-chain, resulting in possible issues during the test operations. Therefore, the designer has to be aware of the potential problems when the SEBC is implemented. A way to avoid the padding issue is to have a scan chain, whose length is multiple of the block size, resorting to observation points (see Section VI.B). However, the insertion of observation points implies the modification of the original circuit. It is thus not always possible to apply the optimization in the case of a non-modifiable core.

D. Extension to multiple scan chains

Both SESC and SEBC solutions can be extended to protect multiple scan chain designs. The encryption of multiple scan chains has a cost in terms of area and power.

The SESC can be directly adapted to multiple scan chains. The only limitation is the number of keystreams that the stream cipher is able to generate. Considering a circuit where multiple scan chains are accessible through L scan-inputs and L scan-outputs, as shown in Fig. 6, the stream cipher has to produce two L-bit keystreams.

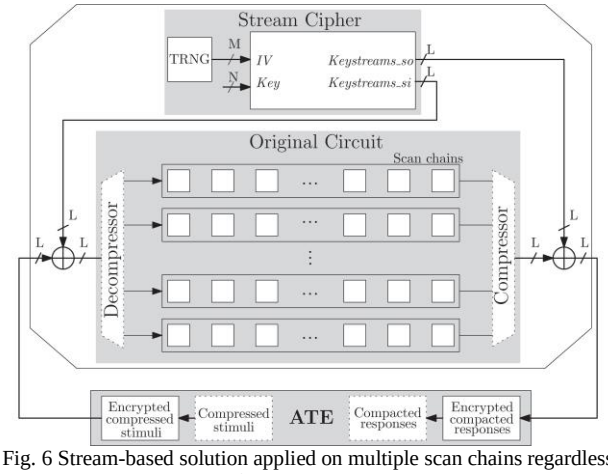

Fig. 6 Stream-based solution applied on multiple scan chains regardless of test compression

Scan encryption	SESC (without TRNG implementation) **Error! Reference source not found.**						SEBC **Error! Reference source not found.**					
Ciphers	AES-128 CTR			TRIVIUM			PRESENT-128 CBC			SKINNY-64-128 CBC		
# scan chains	Area (µm²)	Power (µW)	Freq. (MHz)	Area (µm²)	Power (µW)	Freq. (MHz)	Area (µm²)	Power (µW)	Freq. (MHz)	Area (µm²)	Power (µW)	Freq. (MHz)
1	48,118.20	81.68	10	5,408.52	34.01	10	10,915.96	71.69	10	10,171.52	61.47	10
2	48,128.60	81.70	10	5,553.60	34.02	10	12,134.84	143.4	20	11,390.4	130.6	21.25
4	48,243.00	82.46	10	5,851.04	34.16	10	11,909.16	215.1	30	11,164.72	199.8	32.5
8			10	6,453.20	34.55	10	11,789.56	358.5	50	11,045.12	338.1	55
16			10	7,615.92	35.14	10	11,777.08	645.2	90	11,032.64	614.7	100
32			10	9,999.60	36.37	10	11,472.36	1,218.7	170	10,727.92	1,167.9	190
64							11,440.12	2,365.8	330	10,695.68	2,274.4	370

Tab. 2 Scan encryption with stream cipher and block cipher applied to multiple scan chains designs

The countermeasure can be applied regardless of the test compression technique. The proposed solution can also be applied when a test decompressor is implemented at scan-input, and a test compressor at scan-output. The ATE generates compressed stimuli used to test the circuit under test. The ATE encrypts, by stream ciphering, these generated stimuli following the same test procedure as described before. The encrypted compressed stimuli are scanned in the circuit and decrypted with the keystreams generated for the scan-inputs. The decrypted test stimuli are then applied to the test decompressor. The test responses are compressed before being encrypted on-chip with the keystreams generated for the scan-outputs. Finally, the ATE decrypts the compacted test responses in order to compare them with the expected ones.

The number of possible keystreams depends on the used stream cipher. For instance, the TRIVIUM can compute up to 64 keystream bits in one clock cycle. Therefore, 32 parallel test data can be decrypted at scan-input and 32 parallel test data encrypted at scan-output. For block ciphers in CTR mode, this number is fixed by the ratio between the encrypted block size and the number of rounds for the crypto-algorithm (i.e. the number of clock cycles needed to encrypt a block). For PRESENT-128, the encryption of 64 bits is done in 31 rounds. As a consequence, PRESENT is able to generate two bits of the keystream in one clock cycle. Thus, the solution with PRESENT CTR can only be applied on a single scan chain. Considering the AES-128 CTR, 128 bits are encrypted in 10 rounds. The stream cipher is therefore able to generate 12 keystreams in one clock cycle, encrypting up to 6 scan chains.

Apart from the limit on the number of keystreams, other perspectives can be considered to increase the number of scan chains to be encrypted. One of them is to run the stream cipher at higher frequency in order to increase its throughput. Due to the low combinational complexity of the stream cipher, the frequency increase is a potential solution. Another solution is to implement several stream ciphers even though a bigger area overhead needs to be tolerated. For both perspectives, the cost in power consumption increases also in the same way.

Concerning the SEBC, the extension is not trivial and requires the implementation of a dedicated architectural solution, proposed in [23]. The solution consists in decrypting/encrypting whole *scan slices* at a faster frequency than the scan operations frequency. Scan slices are the set of scan flip-flops of the same rank within the scan chains.

Typically, the scan operations are performed at low frequency to avoid the issues due to overconsumption and overheating within the circuit. This feature is exploited to encrypt/decrypt the data of the test slices at higher frequency, while shifting the test data in the chains at lower frequency. The considered block ciphers, PRESENT and SKINNY, can encrypt slices with length up to 64 bits, corresponding to their block size, at the cost of an increase of the power consumption. Beyond the encryption of 64 scan chains, additional block ciphers must be implemented.

We present in Tab. 2 the experimental results of both SESC and SEBC proposed solutions applied on multiple scan chains. We consider a frequency of 10 MHz for the scan operations. In the following experiments, we use an estimation of the power consumed by the proposed solutions considering the implemented ciphers (AES-128 CTR and TRIVIUM for SECS, PRESENT and SKINNY for SEBC). The power consumption estimation is obtained after the synthesis of the scan encryption countermeasure at the frequency defined in Tab. 2.

The stream cipher decrypts/encrypts test data at 10 MHz regardless of the number of scan chains (columns SESC, Tab. 2), implying a marginal cost in power consumption (about 82 µW for AES- 128 CTR and 35 µW for TRIVIUM). With block ciphers encryption (columns SEBC, Tab. 2), the cost in power consumption increases with the number of scan chains. For the encryption of four scan chains, the power consumption is already 2.5 times higher with block ciphers encryption compared to AES-128 CTR encryption (215.1 µW vs 82.46 µW), and six times higher compared to TRIVIUM (215.1 µW vs 34.16 µW). The difference is even greater for the case of 32 scan chains: the TRIVIUM performs the scan encryption at 10 MHz, consuming 36.37 µW, while PRESENT and SKINNY operates at 330 MHz (2,365.8 µW) and 370 MHz (2,274.4 µW) respectively. However, the generation of several keystreams from the TRIVIUM to encrypt several scan chains has an area cost. To encrypt 32 parallel scan chains, the area cost of SESC with TRIVIUM represents 9,999.60 µm², equivalent to the cost (10,727.92 µm²) of SEBC with SKINNY.

E. Summary

Overall, the scan encryption ensures protection against scan attacks, against reverse engineering through scan chains, and can be used to prevent attacks from scan IOs as well as from malicious cores [9]. An attacker is not able to set the circuit in a desired state, nor to observe internal states of the circuit without knowing the encryption secret key. Moreover, the presented SESC does not show the *two-time pad* vulnerability, making it more secure than the state-of-the-art SESC techniques.

The establishment of the encrypted test communication between an authorized user and the secure circuit is possible with the shared knowledge of the secret key. Consequently, the user with the possession of the secret key is automatically authenticated by the secure circuit. Only authorized users can set and read test data to/from the network.

The proposed countermeasures also present advantages compared to existing scan attacks countermeasures. One of them is that it is still possible to perform in-field diagnosis and debug only for authorized users knowing the key, as opposed to the simple countermeasure consisting in disconnecting the test accesses. Another advantage is the possibility to reuse the key management unit of the system under test if already implemented in the circuit for the management of crypto-accelerators for instance. Previous countermeasures based on a secure protocol using cryptographic primitives, such as in [13]–[15], need to have a dedicated key management.

Conversely to the Built-In Self-Test solutions which may reduce fault coverage, the scan encryption scheme does not impact the test coverage of the original circuit. Moreover, extra hardware for test data encryption is tested without extra delay.

Tab. 3 summarizes the comparison between SESC and SEBC. In order to bring the best comparison possible, we give the pros and cons of the two solutions for implementations achieving the best performance in both cases. Regarding the experimental results, the SEBC is performed by SKINNY-64-128 in CBC mode, while the SESC is performed by TRIVIUM. The SEBC is evaluated for two cases: the solution being optimized or not with the insertion of test points in the original circuit in order to reduce the test time overhead. The SESC is also evaluated for two cases: the TRNG being already implemented or not.

An advantage of the SESC is that the integration in test daisy-chains implies no issue compared to the SEBC. The optimization of the SEBC permits to avoid the integration issue, but it is then not applicable on a non-modifiable core.

SESC has another advantage compared to SEBC concerning the off-chip decryption of the test responses. As shown in Section VII.B, SESC does not require the off-chip decryption of the test responses, while SEBC has to do it due to the confusion and diffusion properties of the block cipher.

When a TRNG is already implemented in the device, another advantage of the SESC compared to the SEBC is the area and test time costs for single scan chains. However, in the case where a TRNG is not available in the device for the scan encryption, the SEBC can be preferred to the SESC due to the important area cost of the TRNG.

Concerning the implementation on multiple scan chains, the SESC has a lower cost in terms of area and power.

Scan encryption	**SEBC** (SKINNY-64-128 CBC)		**SESC** (TRIVUM)	
	Test points insertion for test time optimization:		TRNG already implemented:	
	Not optimized	Optimized	Yes	No
Security				
Scan attacks **Error! Reference source not found.**–**Error! Reference source not found.**	Protected		Protected (two times pad not possible)	
Reverse-engineering through scan side channel	Protected		Protected	
Malicious core **Error! Reference source not found.**	Protected		Protected	
Global features				
In-field diagnosis & debug	Yes		Yes	
Key management	Re-use the key management already implemented		Re-use the key management already implemented	
Integration				
Integration in test daisy chain	Possible issue with the padding of test data	No issue	No issue	
Appl. on non-modifiable core	Yes	No	Yes	
Test coverage				
Original circuit	No impact		No impact	
Secure test infrastructure	Functional test with test patterns of the original circuit		Functional test with test patterns of the original circuit	
Off-chip decryption of the test responses	Required		Not required	
Costs for single scan chain				
Area	10,171.52 µm²	+ Insertion of test points (< 500 µm²)	5,408.52 µm²	+ ~ 31,200 µm² for TRNG
Test time	Depends on the scan length (multiple or not of the block size)	Marginal cost (256 clocks cycles)	Clock cycles required for the initialization phase (1 232 clock cycles)	+ time to initialize the TRNG
Cost for multiple scan chains				
Area	~ 11,000 µm²		From 5,553.60 µm² to 9,999.60 µm²	+ ~ 31,200 µm² for TRNG
Power	From 130.6 µW to 2,274.4 µW		~ 35 µW	+ power consumption of the TRNG
Limit on the number of chains processed by one cipher	From 2 to 64 scan chains		From 2 to 32 scan chains	

Tab. 3 Comparison between scan encryption with stream cipher and block cipher

However, the limit on the number of chains processed by one cipher is lower, from 2 to 32 scan chains, compared to the limit for the SEBC which is from 2 to 64 scan chains.

Therefore, the choice of the SESC or SEBC depends on the original circuit where the protection is implemented: if a TRNG is already set up, if the scan chain length is a multiple of the encrypted block size, if the original circuit is modifiable in order to insert observation points. The number of scan chains to encrypt must also be taken into account.

VIII. CONCLUSION

Access to the scan network can be achieved by the test access ports on the circuit boundary, or by using malicious cores connected to the test daisy-chain. The access to the scan network can be exploited by an attacker to carry out scan attacks and reverse engineer an IP design. To prevent these threats, a solution consists in encrypting the scan network using stream or block encryption schemes.

Several solutions have been proposed based on stream ciphers. However, they present the weakness of using several times the same keystream to encrypt different test data. This paper details a solution based on stream cipher encryption without this vulnerability.

The paper also details another approach, which is based on lightweight block ciphers. The scan content is encrypted in that case with a secret key developed for the current activity, and shared with the authorized users using the key management already present in the circuit.

We have also drawn a comparison between both scan encryption solutions. On the basis of the proposed comparison, it is not possible to designate an absolutely preferred technique. Many factors must be evaluated by the designer in order to make an optimal choice. For instance, the availability of a TRNG in the circuit likely moves the designer choice towards the SESC technique. On the other hand, the multiple scan chains scenario could make the designer prefer the SEBC technique, on the grounds of major parallelization capabilities.

ACKNOWLEDGEMENT

This project has been funded by the French Government (BPI-OSEO) under grant FUI#20 TEEVA (Trusted Execution EVAluation).

REFERENCES

[1] IEEE Standard for Test Access Port and Boundary-Scan Architecture," in *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)* , vol., no., pp.1-444, 13 May 2013

[2] IEEE Standard Testability Method for Embedded Core-based Integrated Circuits," in *IEEE Std 1500-2005* , vol., no., pp.1-136, 29 Aug. 2005

[3] IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," in *IEEE Std 1687-2014* , vol., no., pp.1-283, 5 Dec. 2014

[4] B. Yang, K. Wu and R. Karri, "Secure Scan: A Design-for-Test Architecture for Crypto Chips," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2287-2293, Oct. 2006.

[5] J. DaRolt, G. Di Natale, M. Flottes and B. Rouzeyre, "Scan Attacks and Countermeasures in Presence of Scan Response Compactors," *2011 Sixteenth IEEE European Test Symposium*, Trondheim, 2011, pp. 19-24.

[6] J. Da Rolt, G. Di Natale, M. Flottes and B. Rouzeyre, "Are advanced DfT structures sufficient for preventing scan-attacks?," *2012 IEEE 30th VLSI Test Symposium (VTS)*, Hyatt Maui, HI, 2012, pp. 246-251.

[7] S. S. Ali, O. Sinanoglu, S. M. Saeed and R. Karri, "New scan-based attack using only the test mode," *2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC)*, Istanbul, 2013, pp. 234-239.

[8] J. Daemen and V. Rijmen. The Design of Rijndael. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[9] K. Rosenfeld and R. Karri, "Attacks and Defenses for JTAG," in *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 36-47, Jan.-Feb. 2010.

[10] M. Doulcier, M. -. Flottes and B. Rouzeyre, "AES-Based BIST: Self-Test, Test Pattern Generation and Signature Analysis," *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, Hong Kong, 2008, pp. 314-321.

[11] A. Schubert, W. Anheier, "On Random Pattern Testability of Cryptographic VLSI Cores", *Journal of Electronic Testing: Theory and Applications* archive, June 2000, Volume 16, Issue 3, pp 185–192.

[12] G. D. Natale, M. Doulcier, M. Flottes and B. Rouzeyre, "Self-Test Techniques for Crypto-Devices," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 2, pp. 329-333, Feb. 2010.

[13] G. Chiu and J. C. Li, "A Secure Test Wrapper Design Against Internal and Boundary Scan Attacks for Embedded Cores," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 126-134, Jan. 2012.

[14] Das, A., Da Rolt, J., Ghosh, S., Seys, S., Dupuis, S., Di Natale, G., … Verbauwhede, I. (2013). Secure JTAG implementation using schnorr protocol. *Journal of Electronic Testing: Theory and Applications* (JETTA), 29(2), 193–209.

[15] L. Pierce and S. Tragoudas, "Enhanced Secure Architecture for Joint Action Test Group Systems," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 7, pp. 1342-1345, July 2013.

[16] X. Wang, D. Zhang, M. He, D. Su and M. Tehranipoor, "Secure Scan and Test Using Obfuscation Throughout Supply Chain," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1867-1880, Sept. 2018.

[17] K. Rosenfeld and R. Karri, "Security-aware SoC test access mechanisms," *29th VLSI Test Symposium*, Dana Point, CA, 2011, pp. 100-104.

[18] S. Kan, J. Dworak and J. G. Dunham, "Echeloned IJTAG data protection," *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, Yilan, 2016, pp. 1-6.

[19] M. Da Silva, E. Valea, M. Flottes, S. Dupuis, G. Di Natale and B. Rouzeyre, "A New Secure Stream Cipher for Scan Chain Encryption," *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, Costa Brava, 2018, pp. 68-73.

[20] De Canniere, C., & Preneel, B. (2005). TRIVIUM Specifications. ECRYPT Stream Cipher Project, Report, 30, 2005.

[21] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, P. Paillier and I. Verbauwhede. PRESENT: An Ultra-Lightweight Block Cipher. CHES 2007, LNCS 4727, pp. 450–466, Springer-Verlag Berlin Heidelberg 2007

[22] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., … Sim, S. M. (2016). The SKINNY Family of Block Ciphers and its Low-Latency Variant MANTIS. IACR-CRYPTO-2016.

[23] M. Da Silva, M. Flottes, G. Di Natale and B. Rouzeyre, "Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

[24] Altera. (2009). White Paper Protecting the FPGA Design From Common Threats. Memory, (June), 1–5.

[25] Synopsys. (2015). DesignWare True Random Number Generator Core.