A new secure stream cipher for scan chain encryption

Mathieu da Silva, Emanuele Valea, Marie-Lise Flottes, Sophie Dupuis, Giorgio Di Natale, Bruno Rouzeyre

A new secure stream cipher for scan chain encryption

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

***Abstract*—The accessibility to the internal IP cores of Systems on Chip (SoC) provided by the testing infrastructures is a serious security threat. It has been known for many years that the scan chains can be exploited to retrieve secret keys of crypto-processors. Encryption of the scan chain content is one of the proposed techniques to overtake this threat. Many proposals are based on stream ciphers, due to their moderate area cost compared to that of block ciphers. Stream ciphers encrypt data serially with a keystream generated from an Initialization Vector (*IV*) and a secret key. Stream ciphers have a crucial limitation concerning the encryption of different data with the same keystream, called two times pad. Not enough caution in the *IV* and secret key management has been exercised in previous proposed works. In this paper, we show how the existing implementations can be exploited to perform a scan attack bypassing the encryption of the scan data. We also present a new implementation of scan chain encryption with a stream cipher, based on the *IV* generation by a True Random Number Generator (TRNG). Finally, we show that this new implementation is robust against the aforementioned attack.**

***Keywords*—Scan attacks; Stream Cipher; Two Times Pad; Scan chain encryption**

I. INTRODUCTION

The ongoing breakthrough of the semiconductor industry relies on the shrinking of the technology node. For this reason, the complexity of Integrated Circuits (IC) grows exponentially. Alongside, the probability of encountering faults on silicon dies is not negligible; testing every single die after manufacturing is thus mandatory. Design for Testability (DfT) is used to improve fault coverage and reduce test costs. Scan design is the most popular DfT approach to deal with sequential circuits.

The use of scan chains gives external users amplified controllability and observability on the internal resources of the ICs. This precious feature for testing purposes jeopardizes however the security of the system. The most known attack that exploits the scan chains involves crypto-cores implementing the Advanced Encryption Standard (AES) [1]. If the internal registers of the core are linked to the scan chain, the intermediate results of the encryption can be observed through the test infrastructure. An attacker simply needs to feed the AES core with a series of plaintexts and observe the content of one of the internals registers (i.e. the round register) after the first step of each computation. These attacks, known as scan attacks, have been extensively studied in the literature [2-5]. Authors show in [2] for instance that it is possible to retrieve the whole 128-bits secret key of an AES core by applying 512 chosen plaintexts on average and scanning out the content of the round register after the first round, thus breaking the security of the whole system.

To overcome this antagonism between test and security several countermeasures have been proposed in the literature. The encryption of the test data shifted to or from the scan chain is one of them. It has the advantages of not impacting the fault coverage achieved with traditional scan design and preserving test and diagnostic facilities. The input test vectors are encrypted off-chip by the tester using a secret key. Inside the chip, the decryption is performed using the same key so that initial test patterns are applied to the logic under test. Test responses stored back to the scan chain are encrypted before being shifted out from the scan chain. The tester must then use the test response encryption key to decrypt the responses off-chip. Doing so, an attacker who does not know the secret key is not able to manage the test encryption. Consequently, he/she will not be able to control the IC to a specific state nor to make sense out of the collected data.

Stream ciphers have traditionally been the preferred solution for such test data encryption. This is due to their smaller area footprint and their easier management for serial encryption compared to block ciphers [6][7].

All the countermeasures based on stream ciphers proposed in literature rely on the Trivium cipher [8]. The input of the cipher is a seed, which is used to initialize a Non-Linear Feedback Shift Register (NLFSR) that produces a pseudo-random keystream. The keystream is then XORed with the input plaintext. The result is the ciphertext that, from an external point of view, is undistinguishable from a random bitstream.

A requirement of stream ciphers is that the same seed must be used only once. When several data are encrypted with the same keystream, the encryption loses indeed its security and attacks can be performed (*two times pad attack*).

We show in this paper how the existing countermeasures based on a stream cipher can be attacked. More specifically, we show how the scan chain attack on the AES becomes even easier exploiting this vulnerability. We also present a new implementation of the scan chain encryption with a stream cipher. The proposal is characterized by a new management of the initialization seed that is robust against the attack described in this paper.

The remainder of this paper is organized as follows. In Section II, we provide a background on some key concepts: *1)* the scan chain attack on the AES, *2)* the state-of-the-art on the stream-based encryption for securing the test infrastructures, *3)* the vulnerability that characterizes the existing countermeasures. In Section III, we describe the proposed solution. In Section IV, the results in terms of performance and security are presented. Section V draws some conclusions.

II. BACKGROUND

A. Scan attack on the AES

The AES [1] encryption is performed in multiple rounds of computation (e.g. 10 rounds for the 128-bits AES). In each round, the input text is processed through substitution and permutation operations. After several rounds, these operations ensure confusion and diffusion properties. At the end of each round, the text under encryption is stored into a round register.

The scan attack targets the result of the first round of the computation. It consists in scanning out the content of the round register for observation at the end of the execution of the first round. Here follows a description step by step of the differential scan attack procedure to retrieve one key byte [2], m_1 and m_2 being two 128-bits plaintexts that differ on only one byte B such that $B(m_2) = B(m_1) + 1$:

1) The plaintext m_1 is given as input to the AES core;
2) The circuit is switched to test mode after the first round is completed;
3) The content of the round register c_1 is shifted out through the scan chain;
4) The system is switched back to normal mode;
5) A plaintext m_2 is given to the AES core and the same procedure as before is performed to retrieve another result c_2;
6) The Hamming distance between the two results is computed.

If the Hamming distance between two partially ciphered text c_1 and c_2 is equal to specific values, it is possible to derive one byte of the secret key. The strategy used by the attacker is thus to try many plaintexts couples out, until one of these specific values is hit. On average, the attacker needs to try 32 plaintexts out before being able to figure the key out. This procedure is iterated until the whole secret key is computed. According to the experiments performed in [2], 512 plaintexts on average are enough to be able to retrieve a 128 bits key.

More advanced scan chain structures [9], such as partial scan and insertion of response compactor for X-masking, make the previous scan attack ineffective since the round register of the AES is not entirely observable. However, improved scan attacks [3][4] can deal with these advanced test infrastructures. Experimental data show that only 64 plaintexts on average are sufficient to retrieve the 128-bit secret key of an AES.

In the previous attacks, the plaintexts are applied when the circuit is in normal mode. A simple countermeasure consists in forcing the reset of the circuit as soon as it is switched to test mode. The scan attack described in [5] is carried out entirely in test mode, circumventing this simple countermeasure. The differential scan attack is performed shifting the required plaintexts into the circuit through the scan chain.

B. Securing the scan chain with a stream cipher

The scan chain encryption techniques based on stream ciphers that have been proposed in the literature so far rely all on the Trivium stream cipher. The Trivium [8] stream cipher is based on an NLFSR that takes a seed as input and produces a pseudo-random keystream as output. The seed is made by an 80-bit secret key and an 80-bit Initialization Vector (*IV*). While the key is secret, the *IV* can be public. The only condition is that the same *IV* should not be used more than once.

Scan chain encryption techniques have been proposed for several test infrastructures. When dealing with SoCs, each IP core has its own scan chain infrastructure. Connecting IPs to a bigger infrastructure compliant to a specific standard makes them reachable by an external user. For instance, the device may embed a test interface that is compliant with the IEEE Std. 1149.1 [10] (JTAG), known as Test Access Port (TAP). Single IP cores are then wrapped with a test wrapper compliant with the IEEE Std. 1500 [11]. IP cores can also be reachable by the reconfigurable network compliant with the IEEE Std. 1687 [12] (IJTAG). Each element attached to the network is gated by a Segment Insertion Bit (SIB) that can be opened or closed according to the desired configuration.

K. Rosenfeld and R. Karri propose in [13] an encryption technique that targets the JTAG infrastructure. The *IV* of the Trivium cipher is hardwired on the device using fuses. The configuration of the fuses is established at manufacturing time and never changes during the device lifetime. The secret key is established before each encryption session with a challenge-response protocol. The user sends a challenge to the device. The device sends the expected challenge response to the key input of the Trivium. The stream cipher is run in order to produce the first 80 bits of the keystream. These bits are the final key that is used as secret key during the encryption session. If only trusted users know the configuration of the fuses for the *IV*, they are the only ones who can predict the key used during the session.

In [14] the same approach is used for securing the test procedure of IEEE Std. 1500 compliant IP cores. The encryption is performed with the Trivium stream cipher. The management of the *IV* is not specified by the authors. The secret key is chosen randomly by the user and loaded into the cipher through a dedicated scan chain designed in order to avoid that other IP cores sniff the key.

S. Kan et al. propose in [15] the encryption of the IJTAG reconfigurable network. The Trivium stream cipher is also used for that purpose. The proposed implementation for the secret key and the *IVs* is either with fuses or with Physical Unclonable Functions (PUFs).

C. Exploitation of the two times pad

The stream cipher is considered secure as far as the used keystream is unpredictable from the attacker. The same keystream has also to be used only once. If this condition does not hold, an attack, called *two times pad*, is possible.

Let assume two texts t_1 and t_2 encrypted using the same keystream $S(k, IV)$. If the attacker is able to retrieve the two resulting ciphertexts, the following operation is possible:

$$(t_1 \oplus S(k, IV)) \oplus (t_2 \oplus S(k, IV)) = t_1 \oplus t_2$$

The XOR operation between two ciphertexts results in a XOR operation between the two original plaintexts.

Considering again the scan attack on the AES crypto-core and assuming that t_1 and t_2 are two test results scanned out of the AES scan chain, the two times pad attack allows to obtain directly the Hamming distance used to retrieve the AES secret key.

Assuming that the stream cipher can be reset between two test sessions, the attack will be carried out as follows:

1) A plaintext m_1 is sent as input to the AES block;
2) After the first round of computation, the test response to m_1 is stored in the scanned round register and the circuit is switched to test mode;
3) The content of the scanned round register is shifted out of the scan chain and is encrypted by the stream cipher before being delivered to the circuit output;
4) The circuit is reset, in order to force the stream cipher to generate the same keystream again;
5) The same procedure from points 1) to 3) is performed using a second plaintext m_2 related to m_1 as detailed in section II.A;
6) The two encrypted test responses are XORed.

The obtained result is equivalent to the Hamming distance of the two unencrypted test responses. This means that the encryption of the test responses is totally useless for protecting the AES engine from the scan attack.

The three implementations of the stream-based encryption presented in Section II.B for securing scan chains can all be exploited to perform the two times pad attack.

The technique in [13] allows the key to be controlled by the user (via a challenge-response procedure) and the *IV* is fixed. This means that if the user sends the same challenge twice to the device (no matter the content of the challenge), he is sure that the same key is used twice. Therefore, the same keystream is produced twice and the attack can be performed.

The proposal described in [14] allows the user to directly set the stream cipher key (no challenge-response to set the key). The two times pad attack is easily performed in that case by keeping the key constant during the scan attack.

The countermeasure presented in [15] is based on a stream cipher whose secret key and *IV* are either hardwired with fuses, or given by a challenge-response procedure based on PUFs. The set of secret keys and *IVs* are unique for each device. Otherwise, the authors do not discuss about changing the values of the keys or the *IV* between encryption sessions.

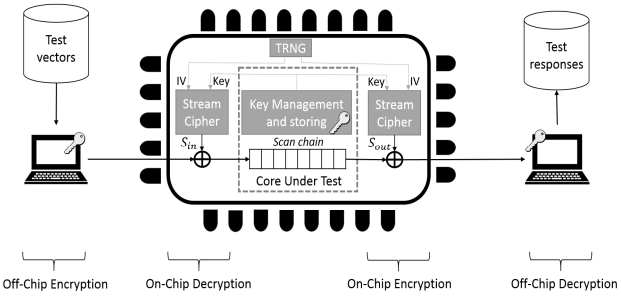

Fig. 1 Principle of the proposed solution based on stream cipher. A TRNG produces the *IV* that is used to seed the stream cipher together with the secret key.

The scan attack can thus be applied without any consideration to the encryption done on the test responses.

III. PROPOSED SOLUTION

We propose a new stream-based test data encryption approach for preventing scan attacks. Firstly, we present the principle of the solution and how the secret key and the *IV* initializing the stream cipher are managed. We describe then the global architecture of the countermeasure with the Trivium implementation.

A. *Principle*

We assume that the original circuit embeds at least one crypto-core, a secure storage for all the secret keys, and a Secret Key Management Unit (SKMU). We also assume that the circuit implements a scan chain and that at least some Flip-Flops (FFs) of the crypto-core are in the scan network. Otherwise the scan attack on the crypto-core cannot be considered, but neither can be the test of that core.

Fig. 1 describes the countermeasure consisting in adding stream ciphers at the input and the output of the scan chain. An attacker unaware of the secret key used for encryption of the test data is not able to set the circuit to a desired state, nor to plainly read the circuit state. Only users with the knowledge of the secret key can access the scan content for debugging purpose.

The secret key of the stream ciphers is stored and managed by the SKMU of the protected crypto-core. By re-using the key management of the original circuit, the solution does not introduce new issues for handling the secret key. The SKMU delivers a dedicated key to authorized users for encrypting the scan content. The *IV* used to initialize the stream cipher is generated by a TRNG. The random *IV* is delivered to the external tester via the scan output for the tester to be able to properly encrypt test patterns. The *IV* is totally random and different after each circuit reset, but does not represent a secret. The key represents the secret, known only by authorized testers.

By initializing the stream ciphers with a different *IV*, the same keystream is not generated twice to encrypt different data. The proposed solution does not present thus the two times pad limitation presented in Section III.B, preventing an attacker to carry out differential scan attacks.

The first step consists in generating test vectors for the original Core Under Test (CUT) and computing the expected test responses by simulation. Before any scan operation, the stream cipher is initialized by generating a random number used as *IV*. The tester encrypts the test vectors off-chip with the random *IV* scanned out of the circuit. Once the stream cipher initialization is finished, the tester can perform the CUT testing by scanning in the encrypted vectors. Each encrypted test vector is first decrypted with the keystream S_{in} generated by the stream cipher at scan input. After that, it is shifted in the scan chain of the CUT. The test vectors are applied to the circuits in order to obtain the CUT test responses. During the shift out operation, the test responses are encrypted with the keystream S_{out} generated by the stream cipher at scan output. The encrypted test responses are scanned out of the circuit in order to be decrypted off-chip by the tester. Once decrypted, the CUT test responses can be compared with the expected ones.

B. Implementation

We propose to encrypt the scan chain with the Trivium stream cipher, generating a keystream from an 80-bits secret key and an 80-bits *IV*. First, this stream cipher presents a low cost implementation. Secondly, an alternative implementation allows increasing the keystream throughput (see Fig. 2). Instead of implementing one stream cipher at scan-in and another at scan-out, the implementation of one Trivium is sufficient to generate two different keystreams for a marginal additional cost of 3 AND gates and 11 XOR gates [8].

One Trivium stream cipher generates thus the keystream S_{in} for the decryption process of the test vectors at scan input and the keystream S_{out} for the encryption process of the test responses at scan output.

The proposed solution has an initialization process, which implies a test time overhead at the beginning of the test procedure. First, the TRNG must have sufficient entropy before generating any random bit, implying a first test time overhead noted $T_{TRNG\ init}$. Secondly, once the randomness of the generated bits is ensured, the *IV* is shifted into the register and, at the same time, scanned out to the tester. The second test time cost, noted $T_{IV\ shifting}$, is due to the time to shift the *IV*. Finally, the last test time overhead $T_{SC\ setup}$ is due to the initialization of the stream cipher with the *IV*, generated beforehand, and the secret key, securely stored in the circuit. Concerning the Trivium stream cipher, $T_{IV\ shifting} = 80$ clock cycles since the *IV* is 80 bits long, and $T_{SC\ setup} = 1152$ clock cycles. Once the initialization process is completed, the stream cipher generates the two keystreams to encrypt test data and there is no more test time overhead during the application of the whole test sequence composed of several test vectors.

The global architecture of the proposed solution is represented in Fig. 2. The scan chain encryption is composed of a TRNG, a shift register containing the IV, the stream cipher and the control unit. The control unit manages the initialization process and the encryption performed by the stream cipher.

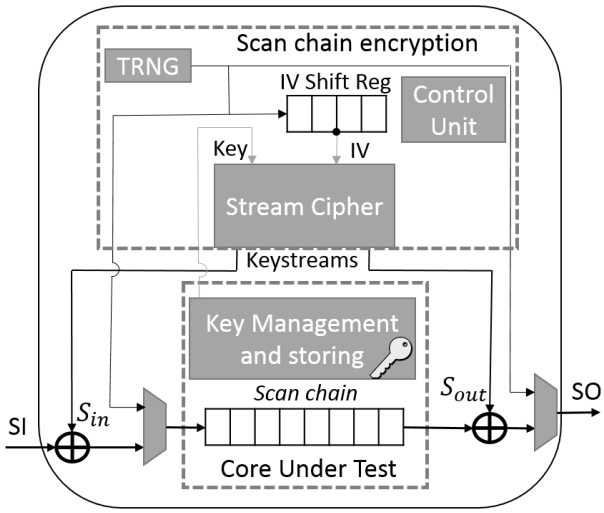

Fig. 2 Architecture of the scan chain encryption based on stream cipher

After a circuit reset, the control unit starts the initialization as soon as the circuit switches from normal mode to test mode. During the entire initialization process, the scan chain is kept inaccessible since the stream cipher does not generate the keystreams. Both the scan input SI and the scan output SO are connected to the TRNG, thanks to the implemented multiplexers. This way, an attacker is not able to send desired data, since the circuit scan chain is connected to the random bitstream generated by the TRNG. An attacker is also not able to observe the internal states of the circuit. He observes the bitstream generated by the TRNG, including the *IV* value. The *IV* is firstly stored in the shift register, before initializing the stream cipher. Once the stream cipher setup is finished, the multiplexers are switched to the SI and SO. The stream cipher generates then the two keystreams. The keystream S_{in} is XORed bit-to-bit to the data (test vectors) at scan input and the keystream S_{out} is XORed bit-to-bit to the test responses of the circuit. The control unit manages the stream cipher encryption during test mode. If the circuit switches to normal mode, the control unit stops the scan encryption. As soon as the test mode is asserted again, the stream cipher resumes the keystreams generation.

Next section presents the experimental results with the implementation of the Trivium stream cipher. However, any stream cipher can be plugged in the presented solution. If a better stream cipher is proposed in terms of security and/or performance, it can replace the Trivium in the proposed solution.

IV. EXPERIMENTS

The proposed countermeasure requires the implementation of a TRNG, a shift register, a stream cipher and a control unit. We have resumed the area cost of these different submodules in Tab. 1, as well as the time to initialize them. The cost of the TRNG is evaluated to 15 000 Gate Equivalent (GE) from the

Submodules	Area cost (Gate Equivalent)	Initialization time (clock cycles)
TRNG	15 000	$T_{TRNG\ init}$: undefined
IV Shift Register	300	$T_{IV\ shifting} = 80$
Trivium	2 048	$T_{SC\ setup} = 1\ 152$
Control Unit	252	/
Total	***17 600***	$\boldsymbol{T_{TRNG\ init} + 1\ 232}$

Tab. 1 Cost of the submodules composing the proposed countermeasure

Synopsys DesignWare IP library [16]. The other submodules imply an area cost of 2 600 GE: 300 GE for the *IV* shift register, 2 048 for the Trivium stream cipher generating 2 keystreams, and 252 GE for the control unit. Concerning the test time overhead, the TRNG initialization to ensure a good randomness of the bitstream is not specified in the specifications. The *IV* shifting and the stream cipher setup have an initialization time of 1 232 clock cycles.

The most expensive part of the solution is the TRNG. However, if the circuit already embeds a TRNG (this is almost always the case as far secure circuits are concerned), this one can be used in test mode by our proposed solution. In this case, the TRNG will not introduce any area overhead. We consider thereafter only the cost introduced by the proposed solution without implementing a dedicated TRNG.

Experiments are conducted on for the protection of 5 circuit examples: a Triple-DES core, two pipelined AES cores in the 128 bits and 256 bits versions, a RSA-1024 core and a LEON3 processor. The synthesis has been performed using a 65nm library with Design Compiler [17]. The test coverage has been evaluated with the ATPG tool TetraMAX [18].

A. Area cost

Without considering the TRNG implementation, the proposed solution with Trivium stream cipher implies a total area of 532 combinational cells and 382 FFs. This area cost represents an overhead of 5 409 µm².

Tab. 2 reports the area of each original circuit after regular scan insertion and the overhead introduced by the scan encryption with the Trivium stream cipher. For the smallest circuit, the Triple-DES core, the area cost represents an overhead of 2.88%. For the LEON3 processor, the largest circuit, the proposed solution induces only 0.28% area overhead.

B. Test time cost

The test time overhead introduced by the proposed solution is just due to the initialization time at the beginning of the test procedure. Without considering the TRNG initialization, the scan encryption with Trivium stream cipher requires 1 232 clock cycles of initialization.

Tab. 2 reports the test time overhead of the proposed solution on each original circuit. The row tagged as "Test Cov." reports the test coverage of the circuit. The row "Test Time" reports the test time to apply the whole test sequence and the overhead introduced when the stream cipher encryption is applied. For each circuit, the ATPG has generated the test patterns to achieve 100% of stuck-at fault coverage, except for the LEON3 processor. In that case, the test pattern generation has been stopped at 70% of fault coverage due to limitation of the ATPG tool in terms of memory allocation.

The test time overhead induced by the initialization of the proposed countermeasure represents a marginal cost compared to the entire sequence needed to test a circuit.

The scan encryption does not affect the fault coverage of the original circuit since the test sequence applied on the scan chain of the CUT is the one generated by the ATPG.

The architecture of the scan encryption needs also to be tested. The regular scan chain insertion cannot be applied on the architecture. Otherwise, the internal states of the stream cipher could be analyzed and the secret key revealed. The test of the testing infrastructure is performed in a different way than the classical scan insertion. The stream cipher is tested functionally with the test patterns of the original circuit, which are decrypted at the scan input, and with the test responses encrypted at the scan output. Since the Trivium stream cipher is based on a shift register, a potential fault is easily propagated on the keystreams. Therefore, during the decryption/encryption of the test data, the proposed solution is tested simultaneously with the original circuit.

We have validated this assertion on the circuit examples (Triple-DES, AES-128, AES-256, RSA and LEON3). The test sequence of each circuit is applied to the scan chain encryption based on the Trivium stream cipher. The CUT test patterns are processed by the keystream at scan-input and the CUT test responses are processed by the keystream at scan-output. As expected, while the sequence detects in the first instance the CUT faults, the sequence covers also 100% stuck-at faults in the original circuit and the additional Trivium circuitry. No additional test patterns are needed to test the proposed countermeasure.

C. Security

Compared to the previous countermeasures based on a stream cipher in [13-15], the generation of the keystream is different at each circuit reset due to the random *IV*, preventing differential scan attacks [2-5]. All the test data passing through the scan chain are encrypted with the secret key shared only to authorized users. The scan content is therefore not observable and controllable for an attacker outside or inside the circuit. Even during the initialization process, the controller disables the connection to the scan chain, preventing any clear bitstream to be scanned in or out.

Regarding the security of the Trivium stream cipher, the cryptanalysis is presented in [8]. It presents no correlation between the keystream bits. Even if an attacker is able to discover a part of the keystream, he will not be able to retrieve the secret key, nor to predict the future keystream, nor to discover the previous generated keystream. In others terms, even if an attacker resets the scan chain content and shifts out the generated keystream, he will not be able to recover the

Circuit	Triple-DES		Pipelined AES128		Pipelined AES256		RSA 1024		LEON3	
	Scanned Circuit	Scan Encrypt. Overhead (%)	Scanned Circuit		Scanned Circuit		Scanned Circuit		Scanned Circuit	
Cell Area (µm²)	187 494	+2.88	367 926	+1.47	669 193	+0.81	468 415	+1.15	1 902 095	+0.28
Test Cov.	100%		100%		100%		100%		70%	
Test Time (clock cycles)	687 101	+0.18	1 944 877	+0.06	4 559 845	+0.03	39 405 239	+0.003	11 612 051	+0.01
Encrypt. Test Cov.		100%		100%		100%		100%		100%

Tab. 2 Cost of the scan chain encryption with TRIVIUM stream cipher on multiple circuits

secret key from the keystream. Moreover, the revealed keystream is not useful to decrypt encrypted test data obtained in another encryption session since a different keystream is used. The cryptanalysis shows also that attacks to guess and determine the internal states of the stream cipher are very complex due to the non-linearity introduced in the keystream generation.

A limitation of the stream cipher is the keystream period. The Trivium stream cipher has a finite period of 2^{64} bits before re-generating the same keystream. The limit is reached after the encryption of 1 million Terabytes. Even if the attacker is able to run the stream cipher at 1 GHz, corresponding to a frequency far too high for scan chains, it will take 317 years before reaching the period. This limitation is therefore not a security concern.

V. Conclusion

Scan chains are a potential threat for circuits embedding a secret, such as crypto-cores. In this paper, we have highlighted a vulnerability presented in previous countermeasures based on stream ciphers. The use of the same *IV* and the same secret key to encrypt several test data leads to a lack of security against differential scan attacks.

We have proposed a new countermeasure based on stream ciphers, taking into account this weakness by generating a random *IV* at each circuit reset. The stream cipher generates thus always a different keystream, making ineffective the differential scan attacks. The solution allows to use the scan chains for both testing and debugging/diagnosis only by authorized users. Experiments have been conducted with the Trivium stream cipher. They report a marginal cost in area and test time.

Acknowledgment

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

References

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

[2] B. Yang, K. Wu and R. Karri. “Secure scan: a design-for-test architecture for crypto chips”. In Design Automation Conference (DAC), pp. 135-140, 2005.

[3] J. Da Rolt, G. Di Natale, M.-L. Flottes and B. Rouzeyre. “Scan Attacks and Countermeasures in Presence of Scan Response Compactors”. In European Test Symposium (ETS), pp. 19-24, 2011.

[4] J. Da Rolt, G. Di Natale, M.-L. Flottes and B. Rouzeyre. “Are advanced DfT structures sufficient for preventing scan-attacks?”. In VLSI Test Symposium (VTS), pp. 246-251, 2012.

[5] S. S. Ali, O. Sinanoglu, S. M. Saeed, and R. Karri. “New scan-based attack using only the test mode”. In International Conference on Very Large Scale Integration (VLSI-SoC), pp. 234-239, 2013.

[6] M. Da Silva, M.-L. Flottes, G. Di Natale, B. Rouzeyre, P. Prinetto and M. Restifo. “Scan chain encryption for the test, diagnosis and debug of secure circuits”. In IEEE European Test Symposium (ETS) pp. 1–6, 2017.

[7] M. Da Silva, M.-L. Flottes, G. Di Natale and B. Rouzeyre. “Experimentations on scan chain encryption with PRESENT”. In International Verification and Security Workshop (IVSW), pp. 45–50, 2017.

[8] C. De Canniere and B. Preneel. “TRIVIUM Specifications”. ECRYPT Stream Cipher Project, Report, 30, 2005.

[9] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann. “OPMISR: The foundation for compressed ATPG vectors”. In IEEE International Test Conference (TC), pp. 748–757, 2001.

[10] Committee, I. S. (1990). IEEE Standard Test Access Port and Boundary-Scan Architecture. IEEE Std (Vol. 2001).

[11] IEEE Standard Testability Method for Embedded Core-based Integrated Circuits. (2012). IEEE Std 1500-2005.

[12] The IEEE Standards Association. (2014). IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device.

[13] K. Rosenfeld and R. Karri. . “Attacks and defenses for JTAG”. In IEEE Design and Test of Computers, 27(1), 36–47, 2010.

[14] K. Rosenfeld and R. Karri “Security-aware SoC test access mechanisms”. In IEEE VLSI Test Symposium (VTS), pp. 100–104, 2011.

[15] S. Kan, J. Dworak and J. G. Dunham. “Echeloned IJTAG data protection”. In IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST), 2016.

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

[17] Synopsys, Design Compiler. [https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html]

[18] Synopsys, TetraMax. [https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/test-automation/tetramax-atpg.html]