Flexible and Portable Management of Secure Scan Implementations Exploiting P1687.1 Extensions

Michele Portolan, Emanuele Valea, Paolo Maistri, Giorgio Di Natale

Flexible and Portable Management of Secure Scan Implementations Exploiting P1687.1 Extensions

Michele Portolan[1], Emanuele Valea[2], Paolo Maistri[1], Giorgio Di Natale[1]

[1]Univ Grenoble Alpes, CNRS, Grenoble INP[1], TIMA, 38000 Grenoble, France
{name.surname}@univ-grenoble-alpes.fr
[2]LIRMM, Univ. Montpellier, CNRS,
34000 Montpellier, France firstname.lastname@lirmm.fr

***Abstract*— Current Design-for-Test solutions for scan-based testing such as IEEE 1687 offer rich and powerful access to embedded resources. While this is extremely helpful for testing purposes, it also raises serious security concerns as malicious users could exploit it to gain control of the chip or leak sensitive information. Current solutions, such as Scan Encryption, are all hardware based and incompatible with the Automated Test Flow, seriously limiting their applicability, portability and reuse. In this paper, we show how the features of the upcoming IEEE P1687.1 can be leveraged and extended to not only include Scan Encryption as a native feature but also as a portable Embedded IP block. The experimental results show that thanks to the new Standard, a fully Secure Automated Test Flow can be integrated inside existing tools without custom software modifications.**

Keywords—IEEE 1687, IEEE P1687.1, Reconfigurable Scan Networks, Secure Access, Encryption, Automated Test Environment

1 Introduction

The emergence of new standards derived from classical JTAG [1], like IEEE 1500 [2] and most notably IEEE 1687 [3], opens new and exciting opportunities not only for test and debug, but also for Design for Test (DfT). Indeed, the new software infrastructure is now able to support IP-Based approaches also in DfT. Third-party ATPG vectors or algorithms defined over instruments and cores can now be described thanks to Instrument Connectivity Language (ICL) and Procedure Description Language (PDL) and be retargeted to the chip-edge through dynamic topologies. However, these opportunities are not without threats: easy and detailed access to the DfT infrastructure can be used to leak sensitive information, and malicious IPs could jeopardize top-level security strategies.

Several approaches have been proposed to provide security within the test and debug infrastructures, while remaining compliant to the hardware of the over-mentioned standards, but little or no effort has been made over the integration within the EDA software flow. Most of the time the vectors issued from the standard retargeting step need to undergo custom post-processing, which limits their applicability, portability and most of all scalability. This is not due to limitations in the approaches themselves, but rather on the difficulty in describing them in a Standard manner. One such example is Scan Encryption: all data exchanged with the System Under Test is encrypted to protect it from tampering. Even though hardware-wise the system remains standard compliant, the coding and decoding operations cannot be described into standard constructs and therefore have to be processed by specific modules outside of the EDA Tool Flow. As a result, this solution can work only on top-level pre-retargeted vectors, limiting reusability and making this solution close to impossible in interactive setups where the behavior of the system is not known beforehand.

Among the techniques introduced to guarantee the security of the test infrastructures, scan encryption has proven to be easily integrable in the hardware test flow, and to assure confidentiality of test data. Nevertheless, its usage is still to be proven. In this paper, we will show how scan encryption, can be efficiently added inside the Standard Automated Test Flow, by exploiting and extending the features of the upcoming P1687.1 Standard [4]. The paper is structured as follows: after introducing the principles of Scan Encryption and P1687.1 in Section 2, we will show in Section 3 how these approaches can not only cooperate, but also extended to propose a novel management of security solutions. In particular, we will show how Scan Encryption can be used also at the IP level to divide the system in Trusted and Untrusted zones, while still remaining inside the standard EDA flow. Section 4 is dedicated to the experimental validation of the solution by an extensive co-simulation campaign, while lastly Section 5 will draw conclusions and point out evolution perspectives.

2 State of the Art

2.1 Security Actors in Electronics Design

Security is peculiar because it implies a contention: an Attacker is trying to access a system, while a Defender is trying to prevent it. The Design, and Test of an electronics system requires the intervention of a big number of actors: Designers, Implementers, Test Engineers, Users. Etc…, each of whom could potentially be either a Defender or an Attacker, often regrouped as Trusted and

[1]Institute of Engineering Univ. Grenoble Alpes

Untrusted Users respectively. This setup is widely used in EDA for confidentiality management: for instance, a Founder will provide detailed information about his technology to a Synthesis or Place&Route Trusted Tool. A generic Untrusted User (for instance, a third-party designer) will use the Tool as a Black Box, while a Trusted User (ex:, a Verification engineer) will have access to all the confidential information. The assumption of Testing is to allow access, so complete locking of the system is not feasible. The application of Security implies a clear identification of Trusted and Untrusted Users and of the methods to allow/deny access to information.

2.2 Scan Encryption

The scan encryption technique sits on the cryptographic foundation of the symmetric encryption schemes. These primitives provide confidentiality to the communication between two parties: in the test scenario, the two communicating parties are the tester and the device under test. Both of them share a secret key, which the circuit manufacturer properly deploys to authorized testers and, at the same time, stores it inside the target device in a secure memory. All data that is exchanged between the tester and the device is encrypted/decrypted thanks to cryptographic modules that are placed at the TDI/TDO interfaces of the device.

When scan encryption is implemented, the typical test procedure is executed through the following steps:

1. All data that are produced by the tester are fully encrypted with a secret key that is known being associated to the device under test;
2. Encrypted data are shifted through the TDI interface of the device, where they are decrypted through a decryption module that is located at the interface with the rest of the test infrastructure;
3. All results of the test procedure, which are sent through the TDO interface of the device, are first encrypted by a cryptographic module;
4. Encrypted results are retrieved by the tester, decrypted, and analyzed.

Scan encryption can be classified according to the kind of cipher that is employed, namely *stream ciphers* or *block ciphers*. Scan encryption based on block ciphers, or block-based scan encryption, has been introduced in [5], where the PRESENT block cipher has been employed. This work, and its extension in [6], proved that scan encryption based on block ciphers induces a negligible area overhead (less than 0.5%) when implemented on complex SoCs. However, block ciphers need to process test data in blocks of a fixed size (usually 128 bits), thus requiring a careful parsing of test data and their subsequent padding in order to fit a multiple of the block size. Even if it does not cause a significant overhead on the test time, it complicates the interface between the serial test interface and the encryption/decryption modules. Scan encryption based on stream ciphers, or stream-based scan encryption, has been proposed several times [7]: encryption is performed serially when processing the data. This property conveniently fits with the serial structure of test infrastructure interfaces, leading to perfectly transparent encryption/decryption operations that do not impact on the performance of the test procedure. Moreover, the capability of stream ciphers to adapt inherently to any input length makes them ideal candidates for IEEE 1687. In this work, we chose the stream-based scan encryption based on the TRIVIUM cipher for implementing a representative automated test flow that is able to both benefit of the flexibility of the P1687.1 standard and guarantee full confidentiality of the involved test data. Indeed, TRIVIUM is often exploited as reference stream cipher implementation because it is a “trade-off soft spot” which provides good security properties for a very limited hardware overhead [8]. Integration into the Test Flow

While its lightweight hardware implementation is undoubtedly Scan Encryption strongest point, its great Achilles’s heel is its software part. Coding and decoding of vectors need to happen outside of the traditional Test Flow, demanding had-hoc post-processing and imposing constraints on the design. For instance in [9] the authors need to impose a specific hardware architecture (PUK for every protected segment) and an extremely heavy software infrastructure that needs to modify both the input PDL/ICL files and post-process the retargeted vector, completely outside any standardized EDA flow (ex: by manually modifying the shift cycle count). These choices limits its applicability and limits both scalability and portability, putting it outside standard flows.

2.3 P1687.1: Interfaces and Callbacks

The IEEE P1687.1 Working Group aims at extending the IEEE 1687 standard by supporting access to the DfT infrastructure through other interfaces than JTAG [4]. Even though the draft has not been finalized yet, the current proposed solution is based on the concept of Transformation Callbacks: as the data (going to or coming from the System Under Test – the SUT) goes through the different interfaces, the EDA Tool needs to “transform” and adapt it to the new interface. So, for instance, SDR or SIR operations might be transformed in a series of I2C transactions.

Because of the high variability of possible interfaces and the difficulty of extracting their behavior and usage from languages such as ICL or even RTL, the WG is rather proposing to encapsulate these transformations into external functions, the Callbacks, called through a standardized API. The MAST tool [10][11] implements a working solution of such a solution, and it is being used as an example for WG discussions.

3 SCAN ENCRYPTION AS P1687.1

3.1 Top-Level Scan Encoding

As explained in Section 2.2, the principle of Scan Encryption is to encrypt/decrypt all data coming in and out of the JTAG TAP exploiting a Key obtained from a Stream Cypher. Functionally speaking, this is exactly the role of a P1687.1 Translator, obtaining the setup of Figure 1.

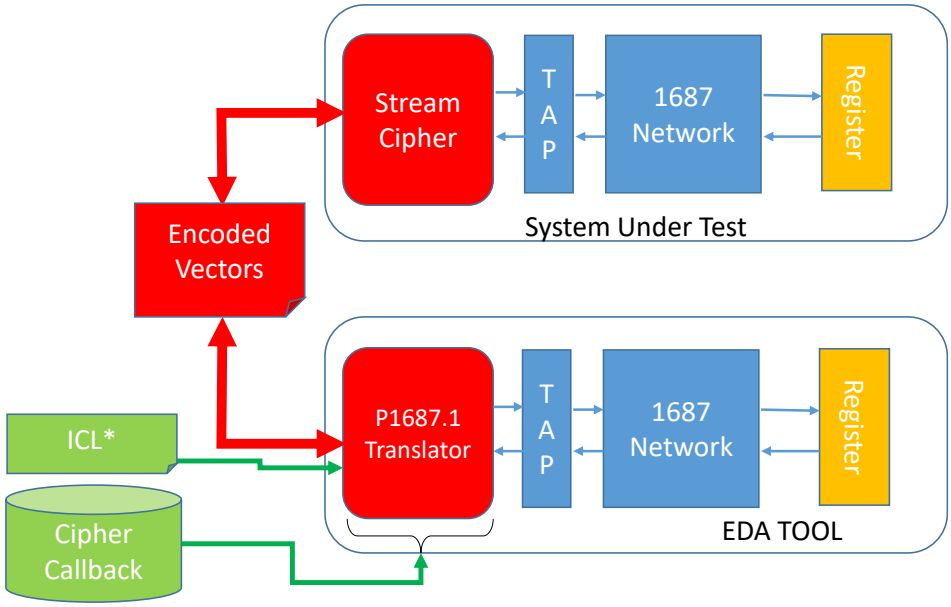


Figure 1: P1687.1 Model of Scan Encryption

The Upper half of the figure depicts the SUT, with a Stream Cypher module before the TAP. In the Lower half, the EDA Tool constructs a mirror of the SUT for the ICL file. In this file, some language extensions (not standardized yet, hence the star) will allow the EDA tool to identify and include the Cypher callback code to handle decoding/encoding of the Vectors (in the Middle) as a P1687.1 transformation. The implementation itself of the callback is extremely simple, as depicted in Figure 2: in the "to SUT" direction, the Translator receives the Plaintext vectors obtained from the retargeting: as their size is known, the Cypher Callback simply needs to generate a Keystream of the correct length and apply it to the Plaintext: most of the times this is obtained through an XOR masking. The same happens in the "from SUT" direction: the plaintext is obtained by applying the same keystream to the encoded data. This approach is compatible with the "Relocatable Vector Format" (RVF) [4][11] currently under discussion in the WG, which encapsulates binary data into specific packets.

The advantage of this setup over legacy approach is that the Encryption is directly handled inside the EDA tool, without the need of external post-processing, therefore allowing the tool to exploit 1687 features both for Dynamic Topology configuration (which directly impacts vector length) and for interactive behavior, as we will point out in the Experimental Section.

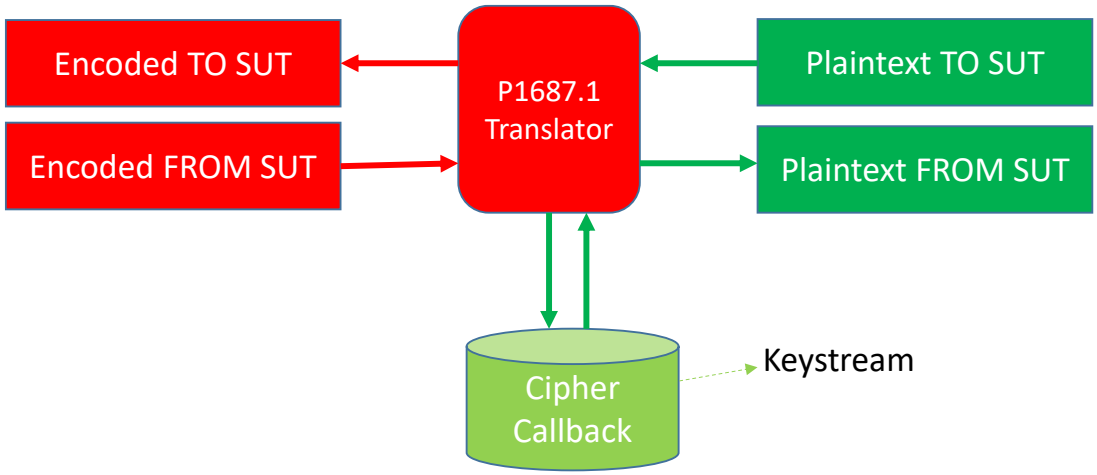
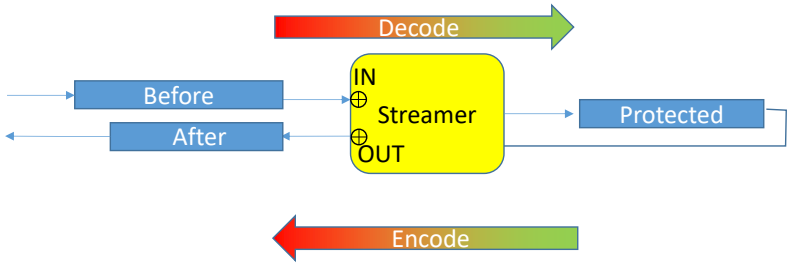


Figure 2: Implementation of the P1687.1 Callback

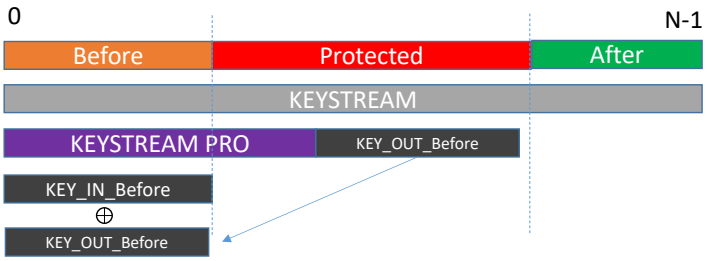
3.2 Inline Scan Encoding

Top-Level protection, as implemented in the previous Paragraph, is effective and simple to set up, but security-wise it is far from optimal. In reference to Figure 1, only the external exchanges (i.e. the Encoded Vectors) are protected, while data inside of the SUT is in plaintext format, therefore vulnerable to simple attacks like snooping or Trojans. Stream ciphers work on a bit-by-bit basis, so from a hardware point of view they can be easily inserted in the middle of the scan chain, as depicted in Figure 3-a. However, this setup is extremely difficult to handle software-wise through traditional approaches. In a classical top-level setup as presented in the State of the Art and implemented in Figure 1, the system is symmetrical: each bit going into or coming out of the SUT is masked exactly once, so coding/decoding is extremely simple when the keystream is known. This is not true anymore in the inline setup: the scan chain segments Before and After the Stream Cipher break the symmetry. While nothing changes for the Protected segment itself, in the "to SUT" direction the bits intended for the non-protected segment AFTER need to go through two encryptions (with different key bits) before reaching their destination. Similarly, "from SUT" bits coming from the non-protected BEFORE section will be encoded twice while being shifted through the Inline Stream cipher. Legacy solutions like [9] impose specific hardware constraints to avoid this behavior, reducing flexibility and augmenting overhead costs.

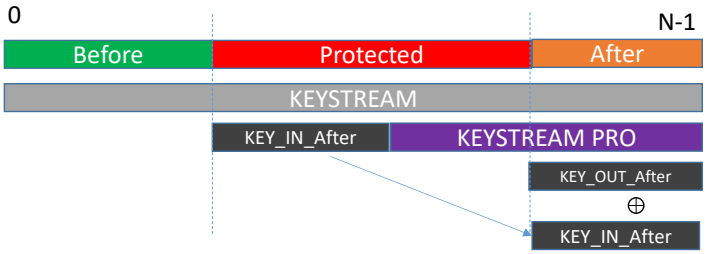
The only way to solve this problem is to know exactly the position of the Stream Cypher in the Scan Chain, as well as the length of each Before, Protected and After section. However, each one of them is potentially a full 1687 sub-network comprising SIBs or ScanMuxes, therefore having variable length. This information is close to impossible to obtain from a set of pre-computed vectors and EDA tools typically do not provide them, making these approaches practically unusable or require heavy post-processing steps as in [9]



a) Inline Streamer setup



b) from SUT (Encode) direction



c) To SUT (Decode) direction

Figure 3: Asymmetric Masking for an Inline Streamer

To solve this impasse, we propose to extend the P1687.1 approach presented in the previous section by defining a new 1687 component called the "Streamer", which can be freely instantiated inside a scan chain and declared to the EDA tool thanks to ICL extensions, replicating the setup of Figure 1. From this information, the Retargeter can extract the exact position of the Streamer inside the active scan chain and the size of the Before, Protected and After segments, as depicted in Figure 3-a.

In the "from SUT" direction, depicted in Figure 3-b, the Streamer needs to encode the "N-bits" Plaintext data coming from the Protected segment before shifting it out, 0 being the bit closest to TDI and therefore the last to be shifted The shift operation will generate a *N*-bit Keystream: the first '*protected_size*' bits, labelled as KEYSTREAM_PRO are used to encrypt the Protected segment, as usual. However, the asymmetrical setup implies that the data coming from the Before segment will go through both the masking Input operator at the beginning of Protected segment and the Output operator, being encrypted twice at different positions in the Keystream: the two masks are labelled *KEY_IN_Before* and *KEY_OUT* and are depicted in Figure 3-b in their position relative the whole Keystream.,. The same happens for the "to SUT" data but in the reverse direction, as depicted in Figure 3-c.

As a P1687.1 extension, we therefore propose a Streamer Callback that can provide a Keystream for a given length, so that the Retargeter can use it to compute the masks detailed above.

4 Experimental Results

4.1 Co-Simulation Setup

We implemented the two approaches described in Section 3 with our MAST tool [10], which implements P1687.1 Callbacks and RVF messages as C++ classes. As a Proof of Concept, we established the an Interactive Co-Simulation Environment of Figure 4, also based on P1687.1. To implement this use case, we chose the Trivium Stream cipher, available in VHDL for the hardware- and in Python for the software-side implementation. The SUT is composed by a 12-bit register driven by a TAP, before which is instantiated a Trivium cipher. The PDL-1 test program makes 5 successive iWrites to the Register and displays on the screen the value obtained through an iGet. It is extremely simple algorithmically, but it requires a full interactive setup to work. The P1687.1 Callback exploits C++ to call and execute the Python Trivium implementation and takes care of encoding/decoding the RVF packets.

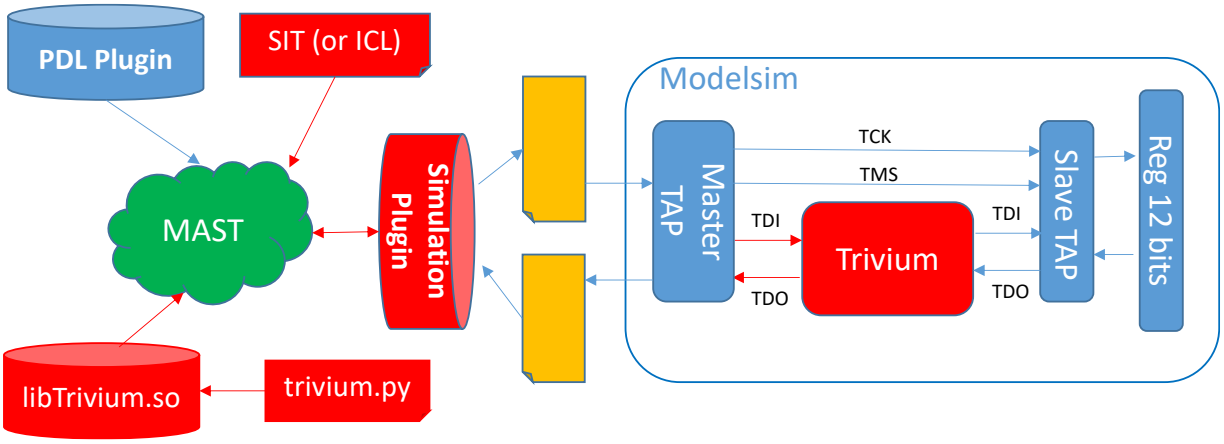


Figure 4: Top-Level Scan Encryption Test Case

The SIT file gives a straightforward hierarchical description of the SUT:

```
1.  TRANSLATOR top Emulation
2.  (
3.  TRANSLATOR Secure Trivium
    "0F62B5085BAE0154A7FA 288FF65DC42B92F960C7"
4.    (
5.     JTAG_TAP TAP 4 1
6.     PDL Console_Incr
7.     (
8.      REGISTER regHI 12 Bypass: "0xABC"
9.     )
10.   )
11.  )
```

The interesting part is line 3: the P1687.1 Translator of type Trivium is instantiated and initialized with two hexadecimal values (respectively, the Secret Key and the Initialization Vector). Thanks to this information, MAST is able to identify and load the Trivium plugin and use it for its Transformations [4]. The Exchange files shows that the data was encrypted:

```
SIR 4 TDI(05);
```

```
SDR 12 TDI(0936);
SDR 12 TDI(0D6D);
SDR 12 TDI(0764);
SDR 12 TDI(0827);
SDR 12 TDI(03FD);
SDR 12 TDI(0DEF);
```

On the other hand, the execution output proves that encoding/decoding was done correctly in both MAST and the SUT:

```
Running 5 iWrites on register regHI
Cycle 1: Wrote 1
      1: Read 54B
Cycle 2: Wrote 2
      2: Read 1
Cycle 3: Wrote 3
      3: Read 2
Cycle 4: Wrote 4
      4: Read 3
Cycle 5: Wrote 5
      5: Read 4
```

As per 1687 specification, the first iGet returns a random number, while in the other cycles the Register maintained the value written in the previous iApply cycle. A direct inspection at Modelsim simulation further proves the correctness.

4.2 Top-Level Scan Encoding and Authentication

This Test Case proves the functionality of the P1687.1 implementation, but its behavior is static: similar results could be obtained through post-processing of the generated vectors. To obtain an unpredictable Test Case we leveraged the Dynamic Authentication setup of [12] where the target Register is behind Secure SIB (S2IB).

The Authentication is based on a random Challenge value, to which a Response must be computed based on the secret key shared between the tester and the device [12]. This implies dynamic decisions made inside MAST, making external vector post-processing for Scan Encryption extremely difficult and cumbersome. On the other hand, thanks to the P1687.1 abstraction, MAST is directly able to use both approaches together by simply describing them in SIT, as follows:

```
1.  TRANSLATOR top Simulation
2.   (
3.   TRANSLATOR Secure Trivium
      "0F62B5085BAE0154A7FA 288FF65DC42B92F960C7"
4.    (
5.   JTAG_TAP TAP 4 1
6.   (
7.    CHAIN Auth_Demo
8.     { [..]}
9.   )
```

For brevity's sake, we omitted in line 8 the detailed description of the Authentication infrastructure: the key getaway is the completely seamless integration of Encryption and Authentication, as it can be seen from the instantiation of the Trivium Scan cipher at line 3, which is not modified.

The execution provides exactly the same results as the first Test Case, proving both the complete transparency of the P1687.1 setup and its capability to react to unpredictable data.

4.3 Inline Scan Encoding

As stated in Section 2.2, the hardware implementation of an inline stream cipher is straightforward: we simply need to instantiate the Trivium block inside the scan chain rather than before the TAP. Please note that the Callback is not the same as in the top level Test Cases, but follows the P1687.1 Extension we defined in Section3.2, even though we were able to reuse most of the cipher-related code. Similarly, the instantiation in the SIT file is almost the same, even though we defined a new "STREAMER" keyword to identify it, as can be seen in line 8:

```
1.  TRANSLATOR top Simulation
2.   (
3.   JTAG_TAP TAP 4 1
4.    (
5.    CHAIN main_chain
6.     {
7.     REGISTER before 12 Bypass: "0xABC"
8.     STREAMER Online Trivium
```

```
    "0F62B5085BAE0154A7FA 288FF65DC42B92F960C7"
9.    PDL Console_Incr
10.   {
11.      REGISTER regHi 16 Bypass: "0x1234"
12.     }
13.    REGISTER After 8 Bypass: "0x56"
14.    }
15.   )
16.  )
```

The console output is still the same, proving the portability of the approach, while the values in the Exchange files are encoded as specified in Figure 3.

To further prove the flexibility of the approach, we replicated several times the Streamer block in different configurations: daisy-chained, behind SIBs, nested, etc… In those scenarios, multiple masking steps are needed at different and unpredictable points of the topology, a situation almost impossible to handle by classical vector post-processing. The results were conclusive: in all cases MAST was able to handle coding/encoding correctly with only the information extracted from the SIT descriptions.

4.4 Trusted and Untrusted Zone

From a Security point of view, thanks to our setup the system can be easily split in Trusted and Untrusted zones, as highlighted in Figure 5 for an Inline Streamer: the Trusted Tool (MAST in this example, in the orange-dotted rectangle) in the middle verifies the Credentials of the User (i.e., the cryptographic key in the SIT/ICL description), allowing him to access the Trusted Hardware zone behind the Streamer (in the Green dashed rectangle). Everything else is freely accessible, as part of the Untrusted Zone (in the Red boxes).

This scheme is completely transparent from the user's point of view and allows plug-and-play packaging and re-use of Secure IP as part of the Standard 1687/P1687.1 flow, a major advantage over current ad-hoc implementations.

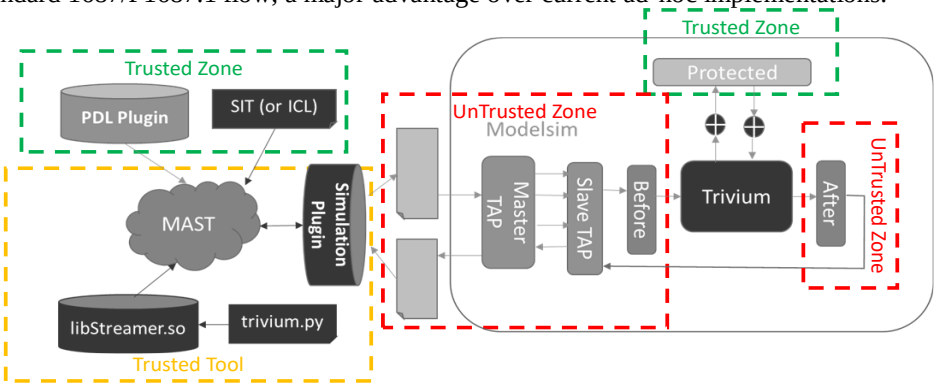


Figure 5: Trusted and Untrusted Zones

4.5 Applicability and Scalability

If compared to literature approaches, our solution is completely transparent from the User's point of view: everything is included inside the standard P1687.1 flow, without any pre- or post-processing steps as in [9]. Moreover, our experiments demonstrate the support of interactive bi-directional execution on the SUT, which to the author's best knowledge is a unique feature while both literature and EDA focus on offline generation. Last but not least, the MAST tool has already been proven to be able to both scale up to extremely complex systems and to be portable on most Test Execution platforms (Desktop, ATE, Embedded)[11], providing an immediate transfer path for the proposed approach.

5 Conclusions and Perspectives

In this paper, we first showed how Scan Encryption can be added to Automated Test Flow thanks the features of the upcoming IEEE P1687.1 standard, and then extended the approach to cover multiple inline scan encryption, therefore obtaining a flexible and truly plug-and-play solution that can be used to add security to IP-based design. To our knowledge, this is the first time that seamless integration of vector encryption exists in an Automated Test Flow. All solutions were proven by hardware co-simulation and even though we used the MAST tool for this step, they could be easily replicated in any 1687/P1687.1 compliant setup equipped with any symmetric cryptographic primitive.

Acknowledgments

This work has been partly funded by the French Government under the framework of the PENTA HADES ("Hierarchy-Aware and secure embedded test infrastructure for Dependability and performance Enhancement of integrated Systems") European project.

REFERENCES

[1] IEEE Std 1149.1-2001, “IEEE Standard Test Access Port and Boundary-Scan Architecture”, IEEE, USA, 2001.

[2] IEEE std 1500 - Standard for Embedded Core Test - http://grouper.ieee. org/groups/1500/.

[3] IEEE Std 1687-2014, “IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device”, IEEE, USA, 2014

[4] M. Laisne et_al., “Modeling Novel Non-JTAG IEEE 1687-Like Architectures”, 2020 International Test Conference (ITC20), November 2020, Washington DC, US

[5] M. Da Silva et al., "Scan chain encryption for the test, diagnosis and debug of secure circuits," 2017 22nd IEEE European Test Symposium (ETS), 2017, , doi: 10.1109/ETS.2017.7968248.

[6] M. Da Silva et_al, "Preventing Scan Attacks on Secure Circuits Through Scan Chain Encryption," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, doi: 10.1109/TCAD.2018.2818722.

[7] K. Rosenfeld and R. Karri, "Attacks and Defenses for JTAG," in IEEE Design & Test of Computers, Jan.-Feb. 2010, doi: 10.1109/MDT.2010.9.

[8] M. Da Silva et_al., “Stream vs block ciphers for scan encryption”, Microelectronics Journal,2019,doi:10.1016/j.mejo.2019.02.019.

[9] Thiemann et al , “On Integrating Lightweight Encryption in Reconfigurable Scan Networks,” , Proc European Test Symp. (ETS 2019)

[10] M. Portolan, "A Novel Test Generation and Application Flow for Functional Access to IEEE 1687 instruments", Proc European Test Symp. (ETS), 2016.

[11] ---, “The Automated Test Flow, the Present and the Future”, IEEE Transactions on Computer-Aided Design (TCAD), DOI: 10.1109/TCAD.2019.2961328, December 2019

[12] M. Portolan et_al, “Dynamic Authentication-Based Secure Access to Test Infrastructure”, 2020 European Test Symposium (ETS 2020), 2020

Michele Portolan is Senior Associate Professor at Grenoble-INP/Phelma. He is an IEEE Standard Society member, and received his PhD from Grenoble-INP in 2006. His research interests focus on EDA solutions for Design for Test Standards and the reliability of electronics systems.

Emanuele Valea received his PhD degree in microelectronics from the University of Montpellier, France, in 2020. He is currently a research engineer at CEA-List, Grenoble, France. His research interests include hardware security and trust, emerging hardware architectures for cryptography and security-related aspects of VLSI testing and reliability.

Paolo Maistri got his PhD in Computer Science engineering from Politecnico di Milano (Italy). He is currently a CNRS researcher and his activities focus on hardware security: design of cryptographic accelerators, attacks to hardware implementations such as side channel and fault analysis, and development of countermeasures at architectural level.

Giorgio Di Natale is CNRS Director of Research, and director of TIMA laboratory in Grenoble (France). His research interests include hardware security and trust, secure circuits design and test, and reliability evaluation. He serves as chair of the TTTC, he is Golden Core member Computer Society and Senior member IEEE.