# A Survey on Security Threats and Countermeasures in IEEE Test Standards

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

# A Survey on Security Threats and Countermeasures in IEEE Test Standards

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

***Abstract*—The growth in complexity of Integrated Circuits (IC) is supported, amongst other factors, by the development of standardized test infrastructures. The feasibility of both end-of-manufacturing and in-field tests heavily depends on the presence of these infrastructures that give detailed access to the IC. The standard test infrastructures are referred as IEEE Std. 1149.1 (JTAG), IEEE Std. 1500 and IEEE Std. 1687 (IJTAG). The security issues arising from the presence of these infrastructures have been fully exposed in the last two decades. This led to the publication of several practical attacks showing how a non-protected test infrastructure can end into the jeopardizing of the entire system. As a consequence, many countermeasures have been proposed. In this survey, we provide: (i) a taxonomy of the attacks that can be performed exploiting the standard test infrastructures; (ii) a taxonomy of countermeasures inspired by the kind of security primitives that are granted in each case.**

***Keywords—Testing vs Security; Hardware Security; Test Standards.***

## I. INTRODUCTION

The continuous shrinking of the semiconductor technology, always leads to new kinds of defects that can possibly affect Integrated Circuits (IC). The increasing usage of cutting-edge technologies in safety-critical applications, leads to strict requirements on the detection of these defects both at the end-of-manufacturing and in-field. For this reason, the importance of testing has become essential in the production flow of semiconductor-based devices. The presence of testing infrastructures inside an IC is vital to be able to successfully perform testing of complex systems. Moreover, these infrastructures have to be kept accessible by the external world also during the mission of the device, in order to perform in-field test and to be compliant with regulations in the case of safety-critical applications.

Scan chains are the fundamental testing infrastructure. All the flip-flops (FFs) of the circuit are replaced with *scan FFs*, which are serially connected with each other. When the circuit is in test mode, the FFs switch to scan mode. This way they establish a serial connection between the Scan In (SI) and the Scan Out (SO) ports of the circuit. The tester can exploit the presence of the scan chain in order to have deeper controllability and observability on the circuit internal logic. When the circuit is in test mode, test data are shifted into the circuit through the SI port of each scan chain. The circuit is then set to normal mode and run for a specific number of clock cycles. In this phase, the circuit reaches a target state. At this point, the circuit is switched back in test mode and the content of the scan chains is shifted out from the SO ports. Finally, the tester compares test responses with the expected values. Since the complexity of the ICs is always increasing, the ratio of external pins over the number of internal nets gets smaller and smaller. In modern Systems-on-Chip (SoC), the number of needed SI/SO pins would be too big. For this reason, test standards propose an access mechanism based on the Test Access Port (TAP), which is composed of only five pins. The TAP requires the tester to implement a standardized protocol in order to access the inner resources of the IC, such as the scan chains.

The first test standard to be introduced was the IEEE Std. 1149.1, also called JTAG, which was conceived for board testing [1]. The user accesses the board through the TAP port, which is composed of four mandatory pins (TDI, TDO, TMS and TCK) and one optional reset pin (TRST). Each device mounted on the board has its own TAP controller, which implements a 16-states FSM that allows the user to perform basic operations with data that are shifted in and out the device (respectively through the TDI and TDO pins). TMS and TCK signals are broadcasted from the board interface to all the devices. The TDI/TDO signals are connected in order to include all the devices into a daisy-chain serial connection. Each device includes an Instruction Register (IR) and a certain number of Data Registers (DR). The IR is loaded with the instruction to be executed. According to the instruction, a different DR is connected between the TDI and TDO terminals. Each device can execute a set of instructions that are mandated by the standard, such as the BYPASS instruction. The BYPASS instruction connects a single flip-flop between the TDI and TDO ports. When data must be shifted to a certain device in the daisy-chain network, all the other devices are set in BYPASS mode. Other instructions access the Boundary Scan Register (BSR), which is a scan register that allows the tester to reach all the pins of the device. When the device is in test mode, Boundary Scan Cells (BSC) are interposed between the device pins and the internal logic. In the *capture* phase, the BSCs are loaded with new values coming from the external pins or the internal logic. In the *shift* phase, these values are shifted out the device and

new data is shifted in. In the *update* phase, the test data is applied to the external pins or the internal logic. When the EXTEST instruction is executed, test stimuli are sent to the external pins. This way, the board connections, external to the devices, are tested. When the INTEST instruction is executed, test stimuli are applied to the internal logic of the device. The INTEST instruction can also be used in order to access the internal scan chains of the circuit. When the scan chains are connected to the JTAG Boundary Scan, the circuit is tested by serially shifting the test stimuli into the TDI port and the test responses out from the TDO port. Even if the JTAG standard was conceived for board testing, it is still the *de facto* standard for the test interface of all kinds of ICs. The TAP port is used as external interface in the other test standards as well.

The IEEE Std. 1500 has been developed to ease the test of IP cores [2]. SoC manufacturers use to acquire IP cores from different vendors. Each vendor independently integrates the test infrastructure inside its IP core. The IEEE 1500 provides a standardized testing wrapper for this purpose. It includes a wrapper controller with its own instruction and data registers, also including the Wrapper Boundary Scan Register (WBSR). The difference with JTAG, is that a parallel input/output interface is provided on the test wrapper. This way, IP cores can be selectively tested resorting to a parallel access that must be provided on the external pins of the SoC. All the IEEE 1500 test wrappers are connected to the SoC-level test infrastructure, which is accessible through the TAP controller.

The latest test standard is the IEEE Std. 1687, also called IJTAG [3]. It deals with the great number of embedded instruments that are integrated inside modern SoCs. These instruments may consist in Built-In Self-Test (BIST) engines for specific IP cores, voltage or current monitors, temperature sensors, aging detectors, SoC configuration registers, etc. Embedded instruments are connected to a Reconfigurable Scan Network (RSN), made by programmable Segment Insertion Bits (SIBs). Each SIB, if opened, gives access to an instrument or to another reconfigurable sub-network. When an instrument is included in the RSN, its Test Data Register (TDR) becomes part of the test infrastructure. This way, the user can dynamically choose the instruments that he or she wants to include in the RSN. The IJTAG RSN is integrated in the SoC test infrastructure. When the tester wants to access the IJTAG RSN, the TAP controller is instructed to select the RSN between the TDI and the TDO pins of the SoC.

Nowadays, many SoCs integrate security primitives inside. Most processors are coupled with cryptographic co-processors that perform, for instance, encryption operations, random numbers generation and hash functions computation. The Advanced Encryption Standard (AES) is the most used encryption scheme. AES hardware implementations are very commonly found in crypto-processors. The security of these modules depends entirely on the secrecy of the encryption key. This means that secret information is stored inside the SoCs, usually inside secure memories dedicated to the storage of secret keys. Intellectual Property (IP) protection of both hardware design and software source code poses important confidentiality requirements in the SoC design flow. Each stakeholder participating to the production of the SoC wants to keep its IP confidential from the other stakeholders and most of all, from the final users of the SoC. For this reason, it is extremely important to avoid that any sensitive information leaks to undesired entities.

The development of test standards goes decisively into contrast with the need for confidentiality and access restrictions. The test infrastructure typically gives the user highlighted controllability and observability on the internal details of the circuit. Moreover, the daisy-chain configuration, typical of the test infrastructures, ensures that multiple independent entities inside the system share the same data connection. This scenario opens many vulnerabilities when sensitive data are shifted through the network. A typical integrated system, designed without security in mind, is vulnerable to many kinds of attacks that can ultimately lead to huge financial losses of the stakeholders.

From a general point of view, the threats involving the test infrastructure of an IC belong to two main categories (Fig. 1):

i. *External threats*: they come from an unauthorized user that has the control of the device TAP controller.

ii. *Internal threats*: they come from a malicious device or IP core that is planted inside the system by a third-party entity (3PIP). In this case, the malicious device can access data propagated through the infrastructure it is connected to.

Many solutions have been proposed in order to protect ICs against the fraudulent usage of the test infrastructures. Each proposed solution has been conceived to counteract more or less specific threats.

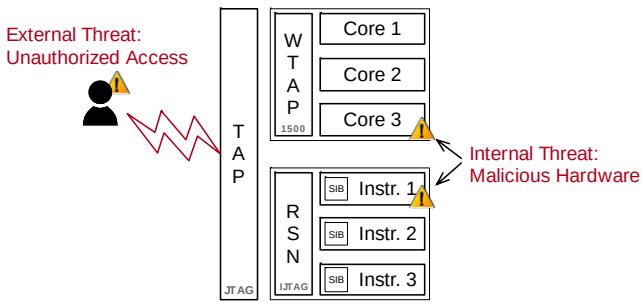


Fig. 1 Threats on the test infrastructure can be originated by (i) an external threat, caused by an unauthorized user accessing the IC; (ii) an internal threat, caused by malicious hardware planted inside the IC.

In this paper, we present a complete taxonomy of the threats and the countermeasures considered in the literature (Section II-III). In Section IV we provide a final discussion and some prospects on the most recent research trends. In Section V we present what we believe are the research challenges that emerge by this survey. In Section VI the conclusions are drawn.

## II. Taxonomy of Threats

Fig. 2 gives the tree structure of the proposed taxonomy. The first level of classification is represented by the two

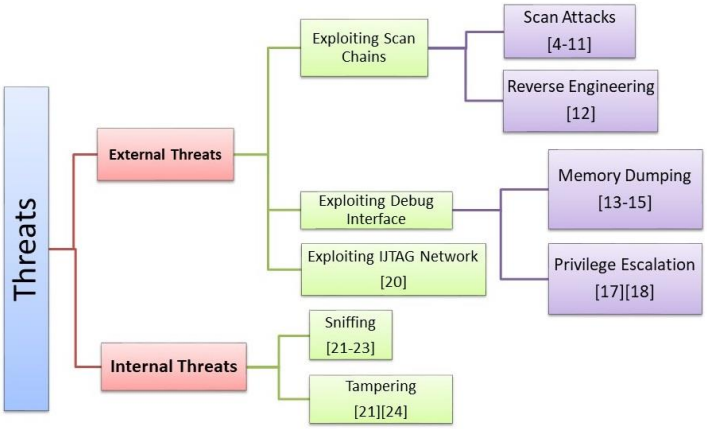


Fig. 2 Taxonomy of the known threats that can be put in place on the standard test infrastructures.

categories that have already been mentioned in the introduction: the external and the internal threats. Each of these two categories is more detailed hereafter according to the action performed by the attacker.

### *A. Untrusted user accessing the test interface*

In this category of attacks, all the components of the system are supposed being trusted. However, if the attacker has physical access to the device, he or she is able to connect to the TAP controller. In this situation, the attacker exploits the highlighted controllability and observability on the devices provided by the test infrastructure. The consequent classification principle is based on the kind of infrastructure that is targeted, namely (i) scan chains, (ii) the debug infrastructure, (iii) the IJTAG reconfigurable network.

#### *1) Exploiting Scan Chains*

Internal scan chains of a target device are accessed through the TAP controller. Scan chains can be connected to the TDI/TDO pins of the test interface executing specific instructions on the TAP controller, such as INTEST. Alternatively, accessing the IEEE 1500 test wrappers of the internal IP cores of a SoC gives access to the internal scan chains of each IP core. In these cases, the attacker can shift arbitrary values into the scan chains. Additionally, any memory element of the circuit connected to the scan chain can be observed shifting its content out. This procedure is the basis of several attacks, called *scan attacks*, which aim to steal secret keys stored inside crypto-processors. Another possibility is to use the scan chain in order to stimulate and observe the responses of the internal logic of IP cores, hence to *reverse engineer* it.

SCAN ATTACKS

When crypto-processors are present inside a device, secret keys are usually stored into integrated secure memories. Even if these secure memories are excluded by the scan chain insertion, scan attacks allow the attacker to retrieve the secret key. Several attacks have been proposed, targeting both symmetric and asymmetric cryptography implementations.

B. Yang et al. presented in 2004 the first scan attack targeting the Data Encryption Standard (DES) [4]. After that, this attack was extended in [5] and executed on an AES crypto-processor. The 128-bit key AES algorithm resorts to 10 rounds of computation to provide the required security. The scan attack exploits the fact that the result of the first round is correlated to the encryption key. In the AES hardware implementations, the result of each round is stored into the *round register*, whose FFs are part of the scan chain. The scan attack can be performed by stopping the execution of the AES after one round of encryption. After that, the circuit is switched to *test mode* and the value of the round register is observed shifting out the content of the scan chain. This procedure is repeated several times until the key can be deduced from the observed data. The attack strategy is to run the encryption of pairs of plaintexts having Hamming distance equal to one. When the Hamming distance of the two related first round results hits some specific values, the attacker can determine one key byte. The encryption of 32 plaintexts is required on average in order to retrieve one key byte. The same procedure is iterated on all the bytes of the secret key. Overall, the attacker needs to apply on average 512 plaintexts to retrieve the 128-bit secret key.

Stream ciphers have also been the target of scan attacks. Y. Liu et al. [6] presented an attack targeting LFSR based stream ciphers. The principle is to run the LFSR for several clock cycles. After that, the circuit is switched to test mode and the content of the LFSR is retrieved through the scan chains. The algorithms developed by the authors bring to light some relations between the internal states of the LFSR. The attacker is thus able to discover the structure of the LFSR and predict the generated keystream.

J. Da Rolt et al. presented in [7] a differential scan attack on Elliptic Curve Cryptography (ECC). The core of the computation in ECC crypto-processors is a point multiplication between the secret key and a scalar value. This operation is performed resorting to several iterations, each involving a different portion of the key. In the target implementation, the attacker is able to observe the result of all the intermediate multiplications resorting to the scan

chains. Exploiting this, it is possible to retrieve part of the secret key. This vulnerability can be exploited in all the use cases of the ECC crypto-processor (e.g. signature generation, key exchange).

J. Da Rolt et al. [8][9] extended the concept of scan attack to advanced test infrastructures. Industrial solutions usually deploy test vector decompressors, test responses compactors, X-masking and X-tolerance, in order to deal with multiple scan chains and complex test infrastructures. Specifically, compacting the test responses inside the chip has often been considered by the EDA (Electronic Design Automation) vendors as a built-in protection mechanism. This paper shows how a differential attack on crypto-processors is feasible, even when these advanced test infrastructures are used. Successively, A. Das et al. [10] performed these attacks on industrial solutions, provided by the main EDA vendors, and proved their vulnerability against the scan attacks.

The described scan attacks are possible as long as the attacker has the capability of switching the circuit from functional mode to test mode and vice versa. For this reason, countermeasures have been developed, based on resetting the scan chains when the circuit is switched from test mode to functional mode. S. Ali et al. [11] conceived a scan attack on AES that is entirely executed in test mode. In test mode, the AES inputs are set through the scan chain, as well as the observation of the round register value. This threat model is conceived in order to overcome the countermeasures based on scan chain reset (or obfuscation) when switching from normal mode to test mode.

#### REVERSE ENGINEERING

The access to the scan chains gives the user the possibility to observe the internal states of the circuit. Exploiting this, an attacker can set specific state values and retrieve the response of the combinational layers of the circuit. Hence, building a database with stimuli/responses couples is possible. A thorough analysis of these data allows the attacker to exactly reverse engineer the netlist of the circuit. Even if this scenario has been assumed as a threat model by many authors, only L. Azriel et al. in [12] showed an implementation of the attack.

### *2) Exploiting the Debug Interface*

Most of the modern ICs integrate a microprocessor. The availability of a debug infrastructure is essential in these systems. The debugging capability must be granted by the hardware designer, in order to assist the software development process. The JTAG interface allows the user to access the debug infrastructure and perform On-Chip Debugging (OCD). If the debug interface is left accessible when the device is sent to the market, malevolent users can exploit it. OCD tools allow the user to tamper the code execution at very low level. This means that security mechanisms implemented at software level can be overcome by OCD. Halting the software execution, a malicious debugger can modify and read the content of specific addresses of the memory, in order to cause unwanted behaviour in the system. All these operations can be easily performed using automated tools and high-level programming languages. Notably, reading memory values is the foundation of attacks aimed at memory dumping and device cloning. On the other hand, the modification of memory values is performed to modify the software execution flow and attain privilege escalation.

#### MEMORY DUMPING

The first memory dumps relying on JTAG were performed in 2006 by S. Willassenn [13] and M.F. Breeuwsma [14]. The objective in both cases was to dump the whole content of a mobile phone memory for forensic purposes. In [13], the target device was a Nokia 5110 mobile phone. The author explains a detailed procedure, in order to access the external flash memory through the CPU JTAG controller and read all data out. However, the procedure presented in [14] is more comprehensive. It shows a more general attack that can be carried out on any portable device with JTAG access. A complete JTAG reverse engineering flow is presented, including the employed technique to find the TAP pins on the board. Once the JTAG infrastructure is accessed, the EXTEST or the DEBUG instructions are selected through the TAP controller. At this point, the attacker is able to send commands to the flash memory and read its content.

F. Domke presented in [15] a reverse engineering procedure to explore undocumented JTAG instructions. Hardware manufacturers usually implement custom instructions in the JTAG infrastructure. They are meant for private in-house utilization, for this reason they are not referred in the device documentation. However, this paper shows a procedure that explores all the undocumented instructions. The final result is that the attacker was able to access the internal scan chains and the internal bus, hence memory data could be read out.

JTAG related vulnerabilities have also affected high-security range devices. In [16], S. Skorobogatov and C. Woods discovered a backdoor in a military chip. The victim device was an ASIC from Microsemi, including a secured FPGA. The authors reverse engineered the JTAG infrastructure and found some undocumented instructions. Through these instructions, it is possible to download or overwrite the FPGA configuration, overcoming all the security features. Exploiting this backdoor, the ASIC producer could virtually retrieve all proprietary designs that their customers synthesized on their products.

#### PRIVILEGE ESCALATION

Penetration testers find serious vulnerabilities on consumer electronics devices on a day-to-day basis. In low-cost devices, producers keep the production costs to a minimum, necessarily scarifying the security concern. For instance, the company in [17] published a JTAG attack performed on a very popular TPLink Router. Once the JTAG interface is found, OCD allows the attacker to halt the execution of the bootloader at any moment. At this point, the memory can be conveniently tampered, in order to force the Linux kernel to run in Single User Mode, i.e. with root privileges.

F. Majéric et al. presented in [18] a JTAG attack exploiting a vulnerability of the Android kernel. Changing

some specific values in the memory, it is possible to unlock the visualization of kernel modules addresses. Knowing the exact memory location of kernel modules is the starting point of several software attacks (i.e. buffer overflow). This vulnerability affected an Android build for Samsung Exynos SoCs and it was patched via software as soon as it was disclosed. However, the authors of this paper showed how, acting through OCD, it was still possible to perform the attack on these SoCs.

*3) Exploiting the IJTAG network*

An attacker who manages to take control of the TAP controller, can also access the IJTAG reconfigurable network, if present. The target of the attacker is to access the configuration register of specific instruments embedded in the SoC. Since the design of the RSN is not known *a priori*, the attacker needs to reverse engineer it and figure out the arrangement of the SIBs. At this point, the attacker can configure them in order to have access to the target instrument.

Hundreds of embedded instruments can be connected to a reconfigurable network. These instruments can be, for instance, BIST configuration registers. BIST engines wrap IP cores and they are activated in order to perform on-line testing. The tester accesses the SoC test infrastructure and writes the right value on the associated TDR, in order to start the BIST procedure. While the BIST is running, the tester can deploy the test infrastructure in order to perform other tasks at the same time. In on-line testing applications, the *tester* is a circuit that schedules the test at the board level or at the SoC level. Since the BIST engines cause high power adsorption from the device under test, their activation must be carefully scheduled during the test phase [19]. If a malicious user is able to access the IJTAG network, many BIST engines can be activated at the same time and possibly cause overheating of the whole circuit. This scenario can lead to a Denial of Service (DoS) attack on the system. Even though any implementation of this threat has never been published, it has been mentioned as a threat in several publications. For example, in [20] the authors mention this menace in order to justify an IJTAG security countermeasure, which is the main contribution of the paper.

Embedded instruments connected to the IJTAG network also comprise SoC configuration registers. These configuration registers may be used to tune SoC parameters, which are determined by the hardware integrator after the production of the chip (e.g. internal voltage levels, clock frequencies). These configurations are part of the intellectual property of the SoC vendor. The authors of [20] mention this scenario as another possible menace affecting non-protected IJTAG infrastructures.

### *B. Malicious hardware*

The actual trend in the IC industry is the globalization of the design and the production. For this reason, the final products come from a design flow that involves many different companies. Third-party companies can provide proprietary IP cores to integrate inside the final SoC. In a typical design flow, the SoC integrator assembles together all the IP cores, coming from different vendors, and designs the SoC level circuitry to grant the correct integration. At this phase, the SoC level testing infrastructure is inserted inside the design. The infrastructure is connected to the test interfaces of each IP core (e.g. TAP controller, IEEE 1500 test wrapper, BIST). How the parties interact in this process is of extreme importance for hardware security purposes. For instance, the SoC integrator does not necessarily have trust in the IP core vendors. Similarly, the IP core vendors does not have trust in each other. However, the IP cores are usually connected to the test infrastructure in a daisy-chain fashion. When the tester sends data to a target IP core through the TAP interface of the SoC, they are shared with other IP cores. The trust level of the SoC integrator with respect to the IP vendors can change according to different scenarios: (i) the IP cores are provided without test wrappers. At this point, the SoC integrator itself provides the IP cores with trusted test wrappers; (ii) the IP cores are already provided with test wrappers. In this case, the SoC integrator cannot have the same trust level in connecting the IP core to the SoC test infrastructure. The same considerations hold at board level, where ICs coming from different parties are mounted on the same board. In general, when test data are shifted through the test infrastructure of an untrusted IP core, there is no certainty that it handles the data according to predefined rules. Two possible threats have been envisaged in the literature so far: (i) the untrusted IP core holds a copy of the test data are shifted through; (ii) the untrusted IP core modifies the value of the test data while they are shifting through.

As far as we know, there is no record in the literature of malicious devices that have been actually found tampering with a test infrastructure. However, some authors have published several attack scenarios involving malicious devices in test infrastructures. These threat models have been largely used by researchers in order to motivate their countermeasures. Therefore, we believe they deserve a section in this survey.

*1) Sniffing*

Each time a user wants to start a communication with a target device connected to the test infrastructure, he or she loads a certain instruction in the IR of the target. The other devices on the same network are programmed in BYPASS mode. A malicious device can be designed in order to store a copy of the data that are shifted through the bypass register. The stolen data can be used by the malicious device in different ways, according to the attack scenario. For instance, test data can be properly filtered in order to store the sensitive information into an internal memory, which is read by the attacker in a second moment. In more complex scenarios, the malicious device can be able to send the data to a remote server. The attacker can process the collected data and retrieve sensitive information about the SoC and the other IP cores.

K. Rosenfeld and R. Karri presented in [21] a threat model involving malicious devices connected to a board level JTAG infrastructure. One of the presented attacks involves a sniffing device recording test vectors sent to another device connected to the same infrastructure. The malicious device

must be upstream the victim in order to make the attack successful. They state that illegally recording test vectors can leak confidential information on the design of the device under test. In a more complex scenario, two colluding devices, one upstream and one downstream the device under test, can record respectively test vectors and responses. This gives even more information about the internal logic structure of the victim.

Many boards on the market embed both a microprocessor and an FPGA, which is used to accelerate part of the computation. The FPGA is configured by downloading a bitstream through the JTAG interface. The content of the configuration bitstream consists in confidential information about the IP core implemented on the FPGA. Moreover, sensitive information about the internal structure of the FPGA is also contained inside the configuration bitstream. If the TAP port of the JTAG is connected to the same network where other devices are connected, the bitstream can be the target of a sniffing attack. In other cases, the configuration bitstream is stored into an external non-volatile memory and it is loaded into the FPGA at the system power-on. In this case, the content of this memory can be accessed through the JTAG interface of the system. It is possible to find track of the importance of this threat model in the technical documentation of FPGA manufacturers. For instance, Altera in [22] presents this kind of threat model, in order to motivate the importance of the security features, which they implement on their FPGA.

S. Kan et al. proposed in [23] a threat model involving malicious instruments connected to the IJTAG reconfigurable network. In this scenario, sniffing instruments can read out confidential configuration data that shifted through the infrastructure. More recently, R. Elnaggar et al. in [24] mentioned the same threat model.

*2) Tampering*

Malicious devices connected to the infrastructure, can modify the content of the shifted data when they are in BYPASS mode. In the case of sniffing attacks, the behavior of the malicious device is completely passive. The sniffing action has no consequences on the behavior of the system. In the case of tampering devices, the behavior of the malicious device is the same as if it was set in BYPASS mode (i.e. it shifts the values from the TDI pin to the TDO pin in one clock cycle). However, the data that are shifted out the TDO pin are different with respect to the data entering the TDI pin. This kind of attack causes a different behavior of the system.

K. Rosenfeld and R. Karri described in [21] a possible data corruption scenario. If the tampering device is upstream the victim one, the data shifted into the victim can be corrupted. If the tampering device is able to smartly elaborate the modification of the data, it can lead the victim device to behave out of the specifications. For example, the target device can be a microprocessor whose firmware is loaded via the test infrastructure. If the content of the firmware is modified while loaded, it can be replaced with whatsoever kind of code, which can cause a very different behaviour of the system.

Another hypothetical scenario presented in [21] can lead to a DoS attack, employing a malicious device. The test infrastructure can be exploited to perform the on-line testing of an IP core inside a SoC. In this scenario, test vectors are stored into an internal memory. When the testing procedure starts, the test vectors are shifted through the test infrastructure and loaded into the device under test. When the responses of the device under test are ready, they are shifted out and compared with the golden ones. In an on-line testing scenario, the comparison is performed on-chip. If the tampering device is downstream the victim one, corrupted responses can be delivered to the comparator. If the tampering device is properly programmed, test responses generated by the device under test can be modified into always being equal to the golden ones. At this point, if the device under test is faulty, the comparator is not able to detect it. This can lead to dangerous situations where the system goes into failure without the possibility for the system to forecast it. However, the malicious device must know the test responses and the exact moment when the test is run, in order to successfully fake test responses.

The same principle can be exploited to threaten data integrity in IJTAG RSNs. In [24], R. Elnaggar et al. presented a threat model involving malicious instruments connected to a reconfigurable network. Untrusted instruments are supposed to be capable of changing the value of specific bits that are shifted through their internal TDRs. This capability can be exploited in order to maliciously change the configuration of the RSN. A possible scenario is the following: a user starts a configuration session in order to include a set of instruments in the RSN. During this process, the configuration bits are shifted through the malicious instrument, which changes the value of specific bits. The result is that an unwanted set of instruments is included in the RSN, without the user even realizing it. Another possibility for a malicious instrument is tampering with input data or with responses involving the victim instrument.

## III. Taxonomy and Classification of Countermeasures

Many countermeasures have been proposed in the literature, aimed to face one or more of the threats described in Section II. In this Section, we propose a taxonomy of countermeasures and we classify the state-of-the-art proposals. A graphical support is presented in the taxonomy tree in Fig. 3.

### *A. Restricted access to the test infrastructure*

This category of countermeasures aims to avoid unauthorized entities to access the test infrastructure. If the user is not authorized, the TAP controller is disabled and the JTAG instructions cannot be executed. This way, further access to the internal IP cores or to the IJTAG reconfigurable network is not possible. As a consequence, the exploitation of the internal scan chains or of the debug infrastructure is prevented. An *authorized user* is defined as someone that access the test infrastructure without causing any damage to all the parties involved in the development of the system.

Two categories of authentication techniques have been identified. One is based on the insertion of a password inside

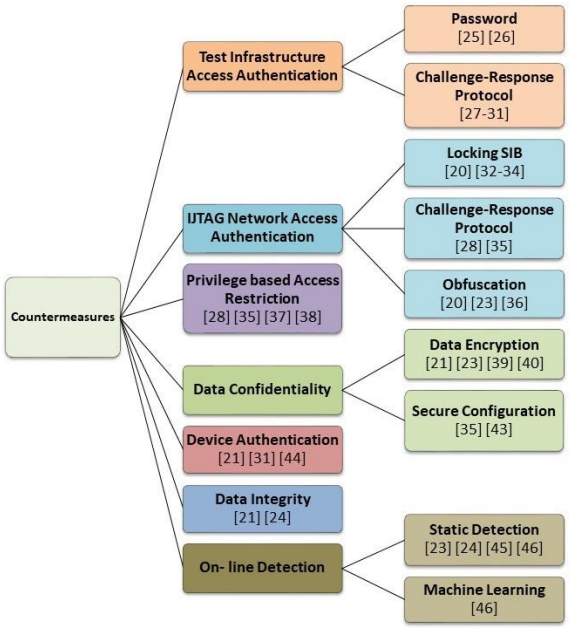


Fig. 3 Classification of the existing countermeasures against the threats on standard test infrastructures

the TAP controller in order to lock or unlock it completely. The other category regroups a series of techniques based on challenge-response protocols implementing cryptographic primitives.

*1) Password-based authentication*

Testing infrastructures protected with locking/unlocking countermeasures have the TAP controller locked by default. In the locked state, the execution of the JTAG instructions is not permitted, except from instructions that do not give access to sensitive data. In order to unlock the TAP controller, a secret password must be shifted inside a dedicated register. If the password is correct, the test infrastructure is unlocked and the authenticated user can access all its functionality. We recall two solutions based on this principle, one targeting the JTAG TAP controller of a generic device, the other one targeting the IEEE 1500 test wrapper of an IP core integrated inside a SoC.

F. Novak and A. Biasizzo [25] presented a solution based on the modification of the TAP controller. The modified TAP controller is able to execute two extra instructions, *LOCK* and *UNLOCK*. When the *LOCK* instruction is executed, the user needs to insert a password inside a special register. As a consequence, the TAP controller goes into a locked state. While in the locked state, the TAP controller decodes all the instructions into the *BYPASS* instruction. This condition does not allow the access to the test infrastructure. When the *UNLOCK* instruction is executed, the user is asked to insert the correct password. If the inserted password matches the one used to lock the system, the TAP controller is unlocked and full access is granted to the user.

G. Chiu and J. Li [26] proposed a solution based on the integration of an LFSR inside the IEEE 1500 test wrapper. The polynomial implemented by the LFSR is secret. When the test infrastructure is in idle state, the test wrapper is locked. The user that wants to unlock the test wrapper must send a seed value to the LFSR. A *golden key* is produced by the LFSR and stored inside the wrapper. After that, the user must shift into the test wrapper a value that matches the golden key produced by the LFSR. If the value is correct, the test wrapper is unlocked. The efficacy of this countermeasure is based on the secrecy of the LFSR structure. Only the user that knows the LFSR polynomial is able to send the right combination seed/key.

All the countermeasures of this category base their efficacy on the presence of a secret element that is shared between the target device and the authorized user. The efficacy can be though easily jeopardized if this secret is leaked to unauthorized entities. It can be observed that in the solution [25] the secret password can be dynamically changed programming each device with a different password. In this case, the leakage of one password undermines the security of only one specific device and it does not affect the whole production. The password can be changed every time the TAP controller is unlocked and relocked. This gives the possibility to the producer to easily change the protection password when needed. In the solution [26] the secret element is the structure of the LFSR. According to the design choices, each IP core can be sold with a different LFSR structure, in order to limit the damage in the case of leakage. However, if the LFSR structure of a specific IP core is leaked, this cannot be changed during the device lifetime.

*2) Challenge-response protocol*

In order to improve the security of the test interface, more complex techniques have been proposed. These techniques are based on challenge-response protocols. The device sends a challenge to the user, which needs to prove his authenticity sending the right response back. The different countermeasures differ according to the kind of cryptographic primitive implemented (e.g. *symmetric* or *asymmetric* cryptography) and the kind of infrastructure needed at the user's side. In the simpler cases, the user directly performs the challenge-response exchange with the device and personally holds the secret key that is necessary for the authentication. There are also more complex solutions where the user needs to obtain credentials from a secure server that holds the secrets required to compute the response. Only if the user successfully authenticates himself with the server, the latter computes the response and sends it to the device. This way the user does not need to directly hold the secrets and the risk of leakage is drastically decreased.

K. Park et al. [27] proposed an authentication protocol for the activation of the TAP controller. The protocol relies on symmetric cryptography and on the verification of the user's credentials based on a secure server. The authentication procedure is based on two steps. In the first step the user is authenticated by the server and obtains a credential that is stored into the device. An authentication between the server and the device is also performed. The device holds a secret key and the server stores a database with the key associated with each device ID. In the second step the user asks the

device for access. The access is granted only if both the user and the device hold a valid credential.

CJ Clark [28] proposed an authentication protocol relying on the computation of the hash function of a random number. The challenge is a random number that is generated by the device. The challenge is sent to the user that appends a secret key to it. The resulting message is hashed with the SHA-256 algorithm. The result is sent back to the device, which verifies its validity computing the hash function itself resorting to the internally stored key. The authentication process is therefore based on the knowledge by the user of the secret key stored inside the device.

A. Das et al. [29] presented an authentication protocol based on Physical Unclonable Functions (PUFs) to protect the access to IEEE 1500 compliant test wrappers. A PUF must be implemented inside the device. A PUF is an element whose behavior depends on physical characteristics that are unique for each single device. Input challenges can be sent to the PUF, which gives unique responses as output. Moreover, the user must have access to a database collecting all Challenge-Response Pairs (CRPs) of the target PUF. The user must send a request to the device when the access to the target test wrapper is needed. At this point, the device sends a random value $\Delta$ to the user. The user searches through the CRP database for two responses having distance $\Delta$. The two corresponding challenges are sent to the device, which are processed by the PUF. The device verifies if the produced responses have distance $\Delta$. In the affirmative case, the access to the test wrapper is granted. The PUF needs to be queried at production time in order to create the CRP database. For this reason, a read-out system for the PUF response must be implemented and permanently disabled after the creation of the CRP database.

R. Buskey and B. Frosik [30] proposed an authentication protocol based on asymmetric cryptography. In this solution, the device holds the public key and a secure server holds the private key. The device sends a challenge to the server together with the device ID and the credentials of the user. The server checks if the user has the authorization for the requested operation. In the affirmative case, the server computes a response using the private key associated with the target device. This response is sent to the device. Finally, the device evaluates the authenticity of the response, using its public key. The challenge-response protocol is based on ECC cryptography.

A. Das, J. Da Rolt et al. [31] proposed another authentication protocol based on asymmetric cryptography. The utilization of the Schnorr protocol is proposed, which implements a signature algorithm based on ECC cryptography, called ECDSA. Using this primitive, an authentication scheme has been designed. The user and the device hold both a public key and a private key. At first, the user sends its public key and ID to the device. The device sends its ID and a challenge to the server in order to verify if the user's public key is linked to a valid certificate. The server retrieves the public key associated with the device ID from its internal storage. After that, it creates a signature using the ECDSA based on the public key itself, the ID and the challenge. This signature is then sent to the device that checks its validity. At this point, the device knows the user's public key. The device then starts the Schnorr authentication procedure. In this exchange, the device, knowing the user's public key, is able to verify the validity of its private key.

The key management is simpler for authentication protocols based on asymmetric cryptography. This is due to the fact that the public key does not need to be kept secret. In the solution [30], for instance, the device stores the public key, and there is no need to keep it secret. The only secret key is the private key, which is stored into a secure server. The solutions based on symmetric cryptography have the advantage of a low implementation cost. For example, the solution [28] is based on hash functions. The implementation cost of hash functions is decisively lower than asymmetric cryptography, such as ECC.

### *B. Restricted access to the IJTAG network*

SoC designers can decide to apply different policies to access embedded instruments connected to the IJTAG reconfigurable network. This network is accessible through the TAP controller of the IC. When a specific JTAG instruction is executed, the RSN is connected between the TDI and TDO pins. However, the policy used to secure the access to the whole JTAG infrastructure can be different with respect to the one used for the IJTAG network. For instance, the designer could be interested in protecting only the IJTAG network, while there is no interest in limiting the access to the TAP controller. In another possible scenario, the access to the IJTAG network is granted to a subset of the entities that have access to the TAP controller. In this case, the secret to access the TAP controller is handed out to authorized users, but only a subset of these users also owns the secret to access the IJTAG network. For this reason, several authentication mechanisms have been proposed in the literature to secure the access to the IJTAG RSN.

When an attacker tries to illegally access the IJAG network, he or she does not know its exact structure. At first, the attacker has to figure out the length of the network in its default configuration. After that, he or she tries to spot the SIBs and to open them, in order to progressively reverse engineer its structure.

Three categories of authentication techniques have been identified. The first one is based on special SIBs that gate the access to private regions of the network. The knowledge of a secret password is necessary in order to open these SIBs. The second category is based on challenge-response protocols that enable the access to the network (or to smaller parts of it) only to authorized users. The last category aims to make more difficult for the attacker to figure out the structure of the network, increasing the complexity of the algorithms used to perform reverse engineering.

#### *1) Locking SIB*

It is possible to restrict the access to specific instruments connected to the IJTAG network hiding them behind special SIBs. These SIBs are locked by default and only authorized users, who know a specific secret, can open them.

J. Dworak et al. [20] proposed a special SIB, called *Locking SIB* (LSIB). The LSIB is a modified SIB with additional logic that enables its opening only if a predefined value is sent to the additional key ports. The key is stored into add-on FFs connected to the RSN. These FFs are connected to the key ports of the target LSIB. When the right value is loaded into these FFs, the target LSIB is opened. LSIBs protect segments of the network that can be accessed only by authorized users that know the secret key. In [32], Gupta et al. showed that the LSIB is extremely flexible. In fact, the authors proposed a parallel IJTAG structure with increased bandwidth and showed that using LSIBs instead of SIBs, the same security level is granted.

H. Liu and V. Agrawal [33] proposed a different kind of LSIB that relies on a Secure LFSR (SLFSR) integrated in the scan network. The SLFSR is placed downstream the LSIB that must be protected. When the LSIB is closed, the SLFSR is activated and the data that are shifted through it are scrambled. This way, the attacker is confused while trying to reverse engineer the network. The parallel output of the SLFSR is connected to the key pins of the LSIB. When the right value is generated by the SLFSR, the LSIB is unlocked. Once the LSIB is unlocked, the SLFSR switches to a simple shift mode and data passing through the RSN are not perturbed anymore. The security of this solution relies on both the knowledge of the secret key to unlock the LSIB and the knowledge of the structure of the SLFSR.

N. Satheesh et al. [34] presented a countermeasure in which the LSIB is unlocked resorting to a PUF-based security module. The security module receives a challenge from the user. This challenge is sent to the PUF, which generates a response. The response is compared with the output of an LFSR. If they are equal, the LSIB is opened. The LFSR produces the same output of the PUF when the user clocks it for $n$ cycles. The value $n$ is a secret that the authorized user must know. The secret $n$ depends on the challenge given to the circuit. Since the PUF has not connection with the external pins of the IC, its CRPs cannot be collected at manufacturing time. The value $n$ associated with each challenge is measured at post-manufacturing running the LFSR until the LSIB is unlocked. This system provides weak security, because its complexity needs to be kept low for the feasibility of the key determination procedure.

The techniques based on LSIBs presented in [20] and [33] rely on a secret password established at design time and hardwired inside the logic of the LSIB. Moreover, the solution [33] also relies on an LFSR whose polynomial is established at design time. For this reason, any leak of this information undermines the security of all the samples that share the same design. The same can be said for the solution [34], even if the behavior of the PUF is different for each device. In any case, retrieving the secret for a single device is not hard, because the complexity of the attack is equal to the complexity of the key derivation procedure executed at design time.

*2) Challenge-response protocol*

The IJTAG network can be also protected by an authentication module that implements a challenge-response protocol. According to the implementation, the authentication procedure can protect the whole RSN, a part of it, or a specific embedded instrument.

The solution proposed by CJ Clark [28] can be applied to the IJTAG network, or to a specific instrument connected to it. The authentication mechanism is the same explained in Section II.A. The difference is that each instrument owns a different key. However, the SHA-256 engine, used to verify the validity of the response, can be shared by all the instruments.

R. Baranowski et al. [35] presented an authentication protocol that gives the access to a secure region of the IJTAG network. In the first step of the protocol the device sends a challenge to the user. The challenge is a random number produced by a TRNG. The user concatenates the received challenge with the keys of the instruments that must be accessed. The obtained message is hashed and the result is the response that is sent back to the device. The device checks if the hash is valid; if this is the case, the user is authorized to access the target instruments. When the authentication is successful, the controller opens the part of the RSN containing the secured instruments.

The techniques based on challenge-response protocols require each instrument to have a secret key associated with it. Therefore, each instrument must manage its access rights independently. For example, in the solution [28], each instrument has to manage the verification of the response sent by the user. This means that each instrument has necessarily an area overhead due to the verification logic. In the solution [35], the authentication controller is centralized. Nonetheless, it is necessary to guarantee a path in the RSN where only the accessible devices are connected. This shows not negligible routing issues.

*3) Obfuscation of the RSN structure*

Another way to secure the access to the IJTAG network is to rise the complexity of the exploration algorithms. The attacker who does not know the design of the circuit tries to figure out the structure of the RSN in order to spot the position of the SIBs and open them in order to access the associated instruments. If the geometry of the RSN is unpredictable, the time required by an attacker to reverse engineer it increases considerably.

A couple of techniques to complicate the structure of the RSN were presented in [20]. In order to make the attack more difficult, trap bits are introduced. These special cells are inserted in the RSN: if the wrong value is updated into them, the output of the cell is irreversibly asserted until a global reset is issued. Trap bits can be connected to a key input of the LSIB, in order to do not allow the attacker to unlock them even if the right key is set in the key bits. Another solution is to use the trap bits to activate an alternative path, in order to put the key bits of the LSIB out of the scan path. This way it is impossible for the attacker to continue forcing the key without a reset of the whole circuit. Trap bits can also be used independently from the LSIBs. For instance, it is possible to connect them to simple SIBs in order to force their closure. Alternatively, they can be set in order to block the shifting of

the RSN. Another technique proposed in [20] is the implementation of hierarchical locks. They are structures that aim to further complicate the unlocking of the LSIBs. The key bits are spread on multiple levels of the network. For this reason, it is necessary to unlock specific LSIBs before being able to access all the key bits that are needed to unlock the target LSIB.

A. Zygmontowicz et al. proposed in [36] other techniques to be combined with the LSIBs. The first one is the introduction of special LSIBs, called *honeypots* (HLSIB). HLSIBs provide access to a sub-network that does not contain any instruments. Instead, a target LSIB is disabled as far as the HLSIB stays open. This gives a fake feedback to the attacker, who may think to have successfully opened the LSIB. In this case, the attacker is motivated to explore the sub-network opened by the HLSIB without knowing that it is a dummy one (i.e. a honeypot). The second proposal is to create a network where some LSIBs are open by default and they need to be closed in order to be able to open other LSIBs. This should confuse the attacker who does not expect to have to reduce the length of the network in order to completely open it. The third proposal is to introduce *switching LSIBs* (SLSIB) that open different networks according to the value that is updated into them. One of the hidden networks is a dead end, the other gives access to the protected instrument. If both the networks have the same length, the attacker don't realize that an LSIB was there, because the length of the network does not change.

S. Kan et al. proposed in [23] a technique that gives the possibility to program the geometry of the RSN at post-manufacturing. The authors proposed the insertion of stub chains. They are additional portions of the scan network that can have different lengths. The configuration of the stub chains is set selecting multiplexers that convey the scan flow on stubs of different lengths. This configuration is decided at manufacturing time using fuses or PUFs. This way, each sample of the device has a different configuration of the stub chains. Therefore, the attacker that is able to reverse engineer the RSN of one device is not able to perform the same attack on all the others.

In the solutions [20] and [36] the countermeasure is coupled to the use of the LSIBs. The time required to open an LSIB with a brute force attack is $2^n$, where $n$ is the number of bits of the secret key. These techniques aim at increasing the attack time. In the solution [23], the structure of the RSN is simply made unpredictable because it is different on each sample of the target device.

### C. *Privilege-based access restriction*

The countermeasures regrouped in this category are an extension of the user authentication techniques. In this case, the users do not have all the same kind of authentication, but they get different privileges on the testing infrastructure according to the trust level they have.

The authentication protocol presented in [28] allows the system to provide different authentications according to the group of JTAG instructions that can be executed. For example, each set of private instructions can be associated with a different key. This way, the users are authorized to use a set of instructions by knowing the associated keys.

J. Backer et al. presented in [37] an authentication mechanism for the debug infrastructure. The debug infrastructure is accessed resorting to the JTAG port of the IC. The aim is to filter out sensible assets that can be retrieved from the system in debug mode. Each asset is tagged in order to indicate its owner. The user must be authenticated to access the debug infrastructure. At the end of the authentication procedure, a *privilege level* is assigned to the user. Finally, the authenticated user can access the debug infrastructure. Each asset that is read from the system is checked runtime to verify that the user's privileges lay him amongst the owners of that asset.

L. Pierce and S. Tragoudas presented in [38] a technique based on a module that manages the authentication protocol and gives the user a privilege level. Moreover, an access monitor filters the update signal of the boundary scan cells. A memory stores the access levels of each resource. When the resource is accessed, its access level is compared with the actual privilege level of the user. The update signal is forwarded only if the access level of the resource is less or equal than the privilege level of the user.

The solution proposed in [35] expects each instrument connected to the IJTAG network to have a secret key, which is used for the user's authentication. At the end of the authentication procedure, the user can access only the instruments that he or she is authenticated for. In order to guarantee this condition, a Secure Scan Chain (SSC) is activated. The SSC has only the cleared instruments connected to it. The other instruments are connected to another portion of the RSN that is not physically reachable by the SSC.

The solutions proposed in [28] and [38] target the JTAG infrastructure. In [28], the privilege of the user determines the kind of JTAG instructions that can be executed, regardless the content of the accessed data. In [38] the user having access to the TAP controller can execute any instruction. However, the content of the accessed data is checked. For example, two users having different privilege levels can both perform debugging, but only one of them may be allowed to access a range of memory locations containing confidential data. The solution [37] is specifically related to the debug infrastructure, which is accessed by the JTAG interface. In the solution [35] the instruments in the IJTAG network must be grouped in different chains, each one accessible only by the users having some specific privileges. The user that wants to obtain the privilege to access a specific SSC must know the secret keys of all the instruments attached to it. In the case in which an instrument belongs to more than one privilege group, it must be necessarily reached by more than one SSC. This may cause non-negligible routing issues.

### D. *Confidentiality of data*

When sensible data are exchanged between the user and the device, the possibility of sniffing from a third malicious entity is possible. This risk is present both in a board environment and inside a SoC, where the malicious entity is

an internal IP core. In addition, the IJTAG networks need to be protected when confidential data risk to be shifted through embedded instruments that are not trusted. Countermeasures to provide data confidentiality have been largely developed by researchers.

We have divided the countermeasures for confidentiality in two categories. The first one is based on the encryption of the data scanned through the testing infrastructure. This way a sniffing device is not able to steal secret information contained in the data. The second category is more oriented to protect the IJTAG networks. The configuration of the RSN is properly modified in order to isolate the untrusted instruments when confidential data are shifted through it.

*1) Data Encryption*

When using this kind of technique, the user needs to encrypt the test data using a secret key. The encrypted data are sent to the target device, which decrypts them. Afterwards, the responses produced by the device are encrypted and sent to the user. Finally, the user decrypts the responses using the same key used by the device. This way, a malicious sniffing device placed either upstream or downstream the target one, does not manage to steal confidential information. The encryption can be performed using different kinds of cipher. The most common are the *stream cipher* and the *block cipher*.

*a) Stream cipher*

All the countermeasures that we are going to present are based on the Trivium stream cipher. This stream cipher is preferred because of its implementation cost. The stream cipher takes as input a seed and generates a pseudo-random bit sequence, called *keystream*. The keystream is then XORed with the message to encrypt. The same is performed for the decryption. If the same keystream is used, consecutive encryption and decryption result in the starting plaintext. The Trivium stream cipher uses a seed that is made by an 80-bit Initialization Vector (*IV*), which is publicly known, and an 80-bit secret key.

K. Rosenfeld and R. Karri proposed in [21] an encryption technique for the JTAG infrastructure. The *IV* is hardwired in the device using fuses that are programmed at manufacturing time. The secret key is derived from a challenge sent by the user. The challenge is sent to the key input of the Trivium cipher. The first 80 bits of the produced keystream are used as response. This response is the secret key that is then used to initialize again the Trivium for data encryption. The user is able to successfully negotiate the encryption key only if the CRPs of the device are known.

Encryption was also proposed in [23] to protect the IJTAG network. The *IV* management is not specified. The secret key is fixed and stored inside the device. The key can be set using fuses or generated exploiting a PUF. In the second case, the output of the PUF should be readable by the external IC pins only at manufacturing time. After that, this connection should be blown in order to avoid the attacker to retrieve the secret key.

K. Rosenfeld and R. Karri proposed in [39] the encryption of test vectors that are fed to an IEEE 1500 compliant IP core. The encryption and decryption of the test vectors is performed with the Trivium stream cipher. The secret key is chosen at the beginning of the session by the user. The key is shifted inside the target core using a dedicated channel that does not go through the other cores.

These different encryption techniques differ only from the key management point of view. In solution [21], the user knowing the fixed *IV* of the target device can successfully generate the secret key. In the solution [23] the key is hardwired. In solution [39], the key is directly decided by the user at each encryption session.

*b) Block cipher*

The encryption techniques based on the block cipher have found their application limited to the encryption of the scan chains so far. However, the same principle can be applied to the whole testing infrastructure. The block cipher takes as input a block of data of *n* bits. Each data block is processed in order to obtain an *n*-bit block of ciphertext as output. The block cipher uses a fixed key, which the authors supposed to be stored inside a *Secure Key Management Unit* (SKMU) present inside the device.

M. Da Silva et al. proposed in [40] the encryption of test data using block ciphers. This proposal targets the protection of the scan chains. The method can also be extended to a wider test infrastructure. The method has been adapted to different scenarios, such as the presence of multiple scan chains. Different implementations of the block cipher have been evaluated in [41] and [42]. Lightweight block ciphers offer a good trade-off between security and implementation costs.

The block cipher technique requires easier key management with respect to the stream cipher. The same key can be used many times in the case of the block cipher. On the other hand, the stream cipher is better fitted to the serial interface of the scan network. In fact, the block cipher technique needs some extra hardware in order to adapt the serial interface of the test infrastructure to the parallel interface of the block cipher. Another drawback of the block cipher technique is the low security level of the encryption configuration. Using this configuration, the encryption is weak. The attacker can know when two plaintext blocks are identical simply observing the equality between ciphertext blocks.

*2) Secure configuration of the IJAG network*

When dealing with IJTAG networks, it is possible to keep confidential data away from untrusted instruments. This is performed acting on the configuration of the network.

The countermeasure proposed in [35] allows the designer to keep the untrusted instruments away from the secure scan chains. If an instrument deals with confidential data, it can be connected to a secure scan chain. This way, the user is sure that sensitive data are not shifted through untrusted devices. Only authenticated users have access to the secure scan chains.

M. Kochte et al. proposed in [43] a design technique for secure IJTAG networks. Secure access patterns can be

generated, such that untrusted instruments are not involved in the network when confidential data are present in the communication. The secure patterns configure the network in order to keep untrusted instruments isolated. When this configuration cannot be achieved due to the structure of the network, a modification of the design is performed. A bypass segment is added in order to redirect the data flow. When confidential data are shifted through the network, the bypass segments are activated and the data are not shifted through the untrusted instruments.

The solution [35] is very efficient when the untrusted instruments do not belong to the set of devices that need the user authentication. In this case, they are not part of the secure scan chain. Thus the confidential data, which are shifted through the secure scan chain, are not exposed to them. The solution [43] is more versatile, because any instrument considered untrusted can be isolated from the confidential data. The main limitation is that this technique is applied at design time, without the possibility to update the security policies at a later stage. Moreover, the insertion of bypass segments does not avoid the possibility to electrically leak the confidential data on the untrusted branch.

### *E. **Device Authentication***

The authentication of the device is fundamental in order to fight the presence of untrusted devices. The user communicating with a target device on a testing infrastructure, needs to be sure that the target is an authentic device, not a fake one. Malicious devices often come from a counterfeiting process. Several countermeasures have been proposed in the literature.

The countermeasure proposed in [21] also deals with the authentication of the device. The user sends a challenge to the device. The challenge is sent to the key port of the Trivium stream cipher. The device computes the response using the initialization phase of the stream cipher (i.e. generating the first 80 bits of the keystream). The user knows the associated response resorting to a database. This way the user is able to check if the device has given the right response. The relation between the challenge and the response depends on the *IV* value of the stream cipher. This value is hardwired in the device using fuses. This configuration is secret and it is set at manufacturing time.

The solution proposed in [31], based on the Schnorr protocol, can be also used for the authentication of the device. This protocol has a bidirectional property that allows the authentication of both the user and the device. The same procedure described in Section III.A can be performed on the other way around to allow the user to verify the authenticity of the device.

J. Dworak et al. proposed in [44] a technique to provide the authentication of a device mounted on a board. Each device owns a unique and secret ID number. When the tester wants to start a communication with a target device, the ID number is requested and checked against the correct one. An attacker, who wants to fake the target device, has to know the right ID associated with it. Hence, the ID must be kept confidential. For this reason, the ID number is encrypted in the transmission, in order to avoid other entities sniffing the JTAG network to steal its value. The encryption is performed by the device. The ID is XORed with a secret key, which must be as long as the ID. At the beginning of the authentication session, the user sends the secret key. In order to protect the key from sniffing, the sent key is obfuscated spreading it inside a random stream of bits. The obfuscation rule is secret and chosen at design time. A hardware module implemented inside the device performs the de-obfuscation of the received key.

The solution presented in [21] does not show high implementation cost, because the Trivium cipher used for computing the response is the same that is used for the encryption of the test data. In [31], the ECC cryptography needed for the implementation of the Schnorr protocol leads to use a high amount of resources. The solution presented in [44] proposes the obfuscation of the key, which does not provide high security. Moreover, the constraint to implement the obfuscation algorithm inside the device at design time does not permit flexibility.

### *F. **Data Integrity***

Granting the integrity of the communication allows the user and/or the device to be sure that the exchanged data have not been modified during the transmission. One of the techniques uses a Message Authentication Code (MAC) appended at the end of each transmitted message. The MAC is a unique signature that is a function of the content of the message. The most used MAC algorithm in this field is the Hash MAC (HMAC). The HMAC is based on hash functions, such as SHA-256. The security of this primitive lies in the shared secret key used by both the user and the device to compute the HMAC algorithm.

The countermeasure proposed in [21] also provides the integrity of the exchanged messages between user and device. This is performed appending a MAC signature to the message. The key used to compute the signature comes from the internal Trivium stream cipher employed for the encryption of the data. A challenge is sent by the user, which is sent to the key port of the Trivium. The *IV* is hardwired with fuses preset at post-production. The Trivium produces the first 80 bits of the keystream and this value is used as secret key for HMAC. The user knows the value of the key because he or she owns the challenge/response pairs that depends on the configuration of the fuses.

R. Elnaggar et al. proposed in [24] a countermeasure that provides integrity in IJTAG reconfigurable networks. Untrusted embedded instruments are supposed to tamper with data, when shifted through their TDR. Therefore, the authors of the present paper proposed to create an alternative path that circumvents untrusted instruments. The alternative path is inserted by the SoC integrator, which is supposed being trusted. The implementation of this solution relies on doubling each untrusted TDR. This way, when data are shifted through the untrusted TDR, the alternative path is activated and the trusted TDR is inserted in the RSN.

The usage of the MAC for integrity is based on a shared secret between the user and the device. The MAC also

provides a weak authentication of the user. An unauthorized user, who does not know the key to compute a valid MAC, can only send invalid messages to the device.

### *G. **On-line detection***

All countermeasures presented so far aim to avoid attacks on the target system. In this section, we present some techniques aiming to detect the execution of the attacks while they are running. This is achieved by on-chip monitoring of the user behavior. When the behavior of the user is considered illegitimate, the system has to be set in protection mode.

Detection techniques can be divided in two categories. The first one comprises all the detection methods based on static rules. As soon as these rules are not respected, the user is considered to be an attacker. The second category comprises methods based on machine learning.

#### *1) Static detection*

Static detection techniques are based on rules that are set at design time. Static detectors are synthetized during the design flow of the device. These detectors usually take as input the patterns sent by the user. If the patterns are not considered compliant to a legitimate behavior, the user is classified as an attacker trying to exploit the circuit.

R. Baranowski et al. proposed in [45] a detection technique for filtering the access to the testing infrastructure. This solution is based on sequence filters that are placed on the TAP controller. They prevent the access to protected instruments and restrain it for instruments that are not completely protected. The filters take as input the sequence of instructions and data at the TDI port to decide whether the access pattern is allowed or forbidden. If the user tries to access a forbidden instrument, the operation is not allowed by the filter. The filters are deactivated by default to enable post-manufacturing test. After that, they can be activated by blowing fuses. Alternatively, an authentication mechanism could be integrated in order to manage the activation of the filters.

In [23] the authors proposed a detection technique for the identification of attempts to reverse engineer the IJTAG reconfigurable network. A checker counts the number of shift cycles that are performed by the user while trying to configure the RSN. A legitimate user knows the structure, hence the length, of the RSN. Therefore, the number of shift cycles necessary to configure the RSN are exactly known. On the opposite, the attacker needs to explore the RSN performing several attempts. The user trying to perform this operation, must perform an inexact number of shift cycles. When the checker detects this situation, the user is considered as an attacker.

X. Ren at al. presented in [46] a detection technique based on representative sequences of instructions. These sequences are chosen at design time as representative of legitimate operations. If the behavior of the user goes sideways for long time with respect to the representative sequences, an attack is detected. In the implementation, a counter is associated with each representative sequence. When all counters stop incrementing, a non-representative sequence is being performed by the user, thus the circuit is subject to an attack.

The countermeasure proposed in [24] can be expanded in order to also detect attempts of data tampering. Genuine instrument responses (not shifted through the internal TDR of the untrusted instruments) are compared with the responses coming from the internal TDR of the instrument. If a difference is detected a tainted bit is set in an extra RSN. When the tester collects the test responses, the presence of a taint bit indicates that an instrument has tried to tamper with some data.

The rules that underlie the static detection techniques must be set at design time. This implies that it is not possible to change the access policies without a complete redesign of the detectors. In the solution [45] the possibility of performing post-manufacturing test using different policies is contemplated. Nevertheless, once the filters are activated, it is not possible to obtain the higher privileges anymore.

#### *2) Machine learning-based detection*

Detection techniques based on machine learning require the implementation of on-chip binary classifiers. They are special circuits that are able to evaluate the sequences of instructions sent by the user and label them as *normal* or *abnormal* behavior. The classifiers must be trained before being operative. During the training phase, instruction sequences belonging to both categories are labeled and sent to the classifier. This way the classifier sets its internal classification parameters. After that, the classifier is able to successfully classify the sequences autonomously in the operative phase. Input data are pre-processed in order to obtain the so-called *feature vectors*. They are a different representation of the data. The feature vectors are the input of the classifier in both the training and operative phase.

In [46], the authors presented two detection techniques based on machine learning. They proposed the deployment of two different classifiers, the *random forest* and the *Support Vector Machine* (SVM).

The random forest classifier is based on decision trees. Each tree takes as input a feature vector (or a part of it) and outputs a binary value that corresponds to its classification. The result of each tree is then sent to a majority voter that establishes the final result. The feature vectors given as input to the random forest classifier are extracted by the executed JTAG instructions. The features are derived from static elements taken by the instruction, plus one transition bit. The transition bit indicates if the transition from the previous instruction to the present one is typical or not.

The SVM is a classifier that defines a decision boundary during the training phase. The decision boundary separates the two classes of samples such that the smallest distance between the decision boundary and any of the samples is maximized. The input of the SVM is a sequence of JTAG instructions. The optimal length of the sequences, which is four instructions, has been determined empirically by the authors.

While the detector based on the random forest classifier is able to provide a classification based on static features of the instruction under execution, the SVM relies on sequences of more instructions. This makes the classification based on the SVM more efficient against attacks that are unknown at the training phase. A common drawback of these two solutions is that each time a new attack is conceived, could be necessary to perform the whole training process again. Moreover, machine-learning techniques show more efficiency if coupled with other protections. This is due to the fact that in some situations the detection can fail because the attack is not recognized. Once the classifier has detected that the user is performing an attack, the system must activate a locking feature or going into a secure mode.

## IV. Discussion

In this section we discuss about three aspects that derive from what has been presented so far. The first one is an evaluation of how state-of-the-art countermeasures are able to protect against the known threats. The second aspect is a brief description of the vulnerabilities that have emerged so far on some of the presented countermeasure. Finally, we are going to discuss about the formalization of the security in the test infrastructures, which we believe being an emerging topic in this field.

### A. *What do we need to reach complete protection?*

In Table 1, an analysis of the protection capability of each countermeasure is reported. The content of the present table is derived by what the authors of each countermeasure stated with respect to its security capabilities. The first conclusion that can be drawn is that there is no countermeasure able to cover all the existent threats. On the other hand, it is possible to conclude that integrating together no more than two different countermeasures is enough to provide a complete protection. For instance, providing both confidentiality and integrity of the communication is enough to protect the test infrastructure from all the know threats.

It is possible to state that when the TAP controller is protected by an authentication mechanism, also the IJTAG reconfigurable network is protected as a consequence. The authentication mechanisms for the IJTAG networks are strictly necessary only when the TAP controller cannot be protected.

The encryption and the integrity of the communication are the only countermeasures that are able to protect from both internal and external threats at the same time. All the other countermeasures have their efficacy limited to only one of these categories.

Table 1 reports a binary criterion for defining the protection of the countermeasures. As we discussed in Section III, many of these countermeasures are declared to be effective against some threats. Nevertheless, the specific implementation leaves space to perform attacks on them, making their security ineffective. In the present analysis we provide an evaluation of the theoretical protection granted by each countermeasure based on the kind of cryptographic primitive that is used. As a matter of fact, the specific way in which the cryptography is implemented leaves space to many possible attacks. We will deal with this aspect in the next subsection.

### B. *Attacks on the countermeasures*

Many of the presented countermeasures can be threatened by known attacks. For many countermeasures we have largely discussed their weak points in Section III. However, some attacks targeting the countermeasures have been presented in the literature.

In [47] the authors showed that the LSIB key logic is subject to *simple power analysis*. This is a side-channel attack

Table 1 Analysis of the protection granted from each countermeasure against the known threats

| | | External Threats, exploiting: | | | Internal Threats | |
|---|---|---|---|---|---|---|
| | | Scan Chains | Debug Interface | IJTAG | Sniffing | Tampering |
| Access Authentication | Password based | Yes | Yes | Yes | No | No |
| | Challenge-Response Protocol | Yes | Yes | Yes | No | No |
| IJTAG Access Authentication | Locking SIB | No | No | Yes | No | No |
| | Challenge-Response Protocol | No | No | Yes | No | No |
| | Obfuscation | No | No | Yes | No | No |
| Privilege based authentication | | Yes | Yes | Yes | No | No |
| Data Confidentiality | Encryption | Yes | Yes | Yes | Yes | No |
| | Secure Configuration | No | No | No | Yes | Yes |
| Device Authentication | | No | No | No | Yes | Yes |
| Data Integrity | | Yes | Yes | Yes | No | Yes |
| On-line Detection | Static | Yes | Yes | Yes | No | Only when JTAG instructions are tampered |
| | Machine Learning | Yes | Yes | Yes | No | |

based on the observation of the power traces produced along the computation time of the IC. They solve this problem proposing to share some key bits in the RSN between multiple LSIBs.

In [48] the authors showed a vulnerability present on all the encryption methods based on stream cipher. The weakness relies on the bad management of the secret key and the *IV* of the stream cipher. The problem is that the chosen management of these parameters allows a malicious user to perform the *two times pad* attack. In this attack, the user can lead the stream cipher to perform two consecutive encryptions using the same secret key and *IV* values. This leads to the easy retrieval of information on the plaintext, performing a simple XOR operation between the two resulting ciphertexts.

### C. *Towards a formalization of security*

The reconfigurable scan networks of the IJTAG standard are modeled resorting to a description language called Instrument Connectivity Language (ICL). This representation is well suited to the development of formal models that facilitate the efficient generation of access patterns. In [49] the authors introduced the *CSU-accurate RSN model* (CAM) that permits the complete formal verification of the RSN. In [50] the same authors enhanced this approach resorting to a model checking technique to verify some security properties of the RSN. More recently, in [51] the authors proposed a new method that detects security violations in the RSN and structurally transforms it into a secure RSN. The security specification for applying this model is based on the definition of a *trust category* for each instrument, together with the definition of a *sensitivity level* of the data that are shifted through the RSN.

The ongoing research on the security aspects of the RSNs was one of the reasons that inspired the introduction of a suite of IEEE 1687 benchmark networks [52]. It is composed of a series of IJTAG networks, described in the ICL language, of different complexities. These benchmarks have recently found their first application as experimental platform for RSN security purpose in [51].

## V. Emerging Challenges

The classification proposed in this survey highlights some trends in this research field. It is evident that the majority of the proposed countermeasures consists in authentication mechanisms for the test infrastructure. Security countermeasures based on user authentication can protect the system only against external threats. The classification of the attacks shows that external attacks are the only ones that have found large implementation in the reality. External attacks can be easily performed, even by not specialized entities. For this reason, the primary effort of the IC vendors is the protection of the TAP controller, in order to try to avoid the plethora of external threats.

As it was largely discussed, the authentication mechanisms are based on symmetric or asymmetric cryptography. Asymmetric cryptography offers superior capabilities in terms of security and key management; however, the implementation cost is very high. In general, authentication mechanisms require the hardware implementation of crypto-circuits. This overhead can be tolerated in high-end products, where budget restrictions are loosened. However, the need for security is starting to affect market sectors where the aggressive cost reduction is not an option. For instance, in the IoT domain, many vendors participate in selling out very low-cost microcontroller-based devices. Microcontrollers are programmed to perform very simple operations (e.g. acquiring data from sensors, performing a simple computation and sending the results over the network). Since the final price of these devices must be kept extremely low, the employment of SoCs, equipped with crypto-hardware dedicated to the authentication of the test infrastructure, is definitely out of the question. This leaves the whole network, where unprotected devices are connected to, vulnerable to attacks. For these reasons, we strongly believe that there is a significant interest for researchers in lightweight authentication mechanisms for testing infrastructures.

As far as lightweight security is concerned, we believe that this concept is not only limited to low hardware overhead of the secure infrastructure. Another important aspect to face is keeping the design effort low. The lack of secure standards in the test infrastructures make the security design a prerogative of big companies, which can allow themselves to invest money in order to build a domestic know-how on security matters. The availability of standardized guidelines and implementations for secure test infrastructures can allow smaller companies to provide protection to their devices, keeping their investments contained.

Analyzing the state-of-the-art, it is also clear that a deeper study on the feasibility of internal threats is required. Even though real-world attacks of this kind have not been witnessed yet, they must not be neglected. We believe that internal threats must be addressed by researchers with the same emphasis that is given to the research on hardware Trojan horses. Due to the importance of these kind of threats, it is also important to develop more extensive protection mechanisms that also include confidentiality and integrity of test data.

## VI. Conclusion

The development of IEEE test standards gave a big benefit to the development of complex SoCs. Users can rely on the simple interface provided by the TAP port and easily access the hundreds of resources and functionalities present inside the system. Granting the access to the test infrastructure while the system is in the field is necessary in order to provide on-line testing and maintenance. On the other hand, several attacks can be performed exploiting the test infrastructure. In this paper, we have provided a proposal of taxonomy for organizing the different kind of threats and countermeasures that have been published so far. The evaluation of the kind of protection provided by each countermeasure shows that a major effort has been performed by the scientific community on the development of complex authentication mechanisms. While these techniques provide a strong protection against unauthorized access to the infrastructure, they lack of protection against internal threats.

Nevertheless, we have also shown how the countermeasures based on encryption and integrity can provide together a more complete protection. Hence, we believe that it could even be possible to integrate encryption and integrity into the test standards themselves, in order to provide a smooth design-for-test flow, equipped with strong security primitives.

## ACKNOWLEDGMENT

This work has been funded by the French Government (BPI-OSEO) under grant FUI#20 TEEVA (Trusted Execution EVAluation) and 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 Standard for Test Access Port and Boundary-Scan Architecture," in *IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001)*, pp.1-444, May 2013.

[2] "IEEE Standard Testability Method for Embedded Core-based Integrated Circuits," in *IEEE Std 1500-2005*, pp. 1-136, Aug. 2005.

[3] "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device," in *IEEE Std 1687-2014* , pp.1-283, Dec. 2014.

[4] Bo Yang, Kaijie Wu and Ramesh Karri, "Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard," *2004 International Conference on Test*, 2004, pp. 339-344.

[5] Bo Yang, Kaijie Wu and Ramesh Karri. (2006). "Secure Scan: A design-for-test architecture for crypto chips". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10), 2271–2276.

[6] Liu Y., Wu K. & Karri R. (2011). "Scan-based attacks on linear feedback shift register based stream ciphers". *ACM Transactions on Design Automation of Electronic Systems*, 16(2), 1–15.

[7] Da Rolt J., Das A., Di Natale G., Flottes M. L., Rouzeyre B. & Verbauwhede I. (2012). "A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures". *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 43–48.

[8] J. Da Rolt, G. Di Natale, M. Flottes and B. Rouzeyre, "New security threats against chips containing scan chain structures," *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, San Diego CA, 2011, pp. 110-110.

[9] 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.

[10] A. Das, B. Ege, S. Ghosh, L. Batina and I. Verbauwhede, "Security Analysis of Industrial Test Compression Schemes," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 12, pp. 1966-1977, Dec. 2013.

[11] S. S. Ali, S. M. Saeed, O. Sinanoglu and R. Karri, "Novel Test-Mode-Only Scan Attack and Countermeasure for Compression-Based Scan Architectures," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 808-821, May 2015.

[12] Leonid Azriel, Ran Ginosar, and Avi Mendelson, "Exploiting the Scan Side Channel for Reverse Engineering of a VLSI Device," in *CCIT Report #897*, May 2016.

[13] Willassen S. (2006). "Forensic Analysis of Mobile Phone Internal Memory." *IFIP — The International Federation for Information Processing*, vol. 194. Springer, Boston, MA.

[14] Ing. M. F. Breeuwsma, "Forensic imaging of embedded systems using JTAG (boundary-scan)," *Digital Investigation*, Volume 3, Issue 1, 2006, Pages 32-42.

[15] Domke, Felix. "Blackbox JTAG Reverse Engineering." (2009).

[16] Skorobogatov S., Woods C. (2012) "Breakthrough Silicon Scanning Discovers Backdoor in Military Chip." *Cryptographic Hardware and Embedded Systems – CHES 2012*. Lecture Notes in Computer Science, vol 7428. Springer, Berlin, Heidelberg.

[17] Senrio. "JTAG Explained (finally!): Why "IoT", Software Security Engineers, and Manufacturers Should Care." https://blog.senr.io/blog/jtag-explained. 2016.

[18] F. Majeric, B. Gonzalvo and L. Bossuet, "JTAG Combined Attack - Another Approach for Fault Injection," *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Larnaca, 2016, pp. 1-5.

[19] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices," *Digest of Papers Eleventh Annual 1993 IEEE VLSI Test Symposium*, Atlantic City, NJ, USA, 1993, pp. 4-9.

[20] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz and M. Thornton, "Don't forget to lock your SIB: hiding instruments using P1687," *2013 IEEE International Test Conference (ITC)*, Anaheim, CA, 2013, pp. 1-10.

[21] 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.

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

[23] 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.

[24] R. Elnaggar, R. Karri and K. Chakrabarty, "Securing IJTAG against data-integrity attacks," *2018 IEEE 36th VLSI Test Symposium (VTS)*, San Francisco, CA, 2018, pp. 1-6.

[25] F. Novak and A. Biasizzo (2006). Security extension for IEEE Std 1149.1. *Journal of Electronic Testing: Theory and Applications (JETTA)*. 22(3). 301–303.

[26] G. M. Chiu and J. C. M. 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.

[27] K. Park, S. G. Yoo, T. Kim and J. Kim (2010). "JTAG security system based on credentials." *Journal of Electronic Testing: Theory and Applications (JETTA)*, 26(5), 549–557.

[28] C. Clark, "Anti-tamper JTAG TAP design enables DRM to JTAG registers and P1687 on-chip instruments," *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Anaheim, CA, 2010, pp. 19-24.

[29] A. Das, Ü. Kocabaş, A. R. Sadeghi and I. Verbauwhede, "PUF-based secure test wrapper design for cryptographic SoC testing," *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, 2012, pp. 866-869.

[30] R. F. Buskey and B. B. Frosik, "Protected JTAG," *2006 International Conference on Parallel Processing Workshops (ICPPW'06)*, Columbus, OH, 2006, pp. 8 pp.-414.

[31] A. Das, J. Da Rolt, S. Ghosh, S. Seys, S. Dupuis, G. Di Natale, et al. (2013). "Secure JTAG implementation using Schnorr protocol." *Journal of Electronic Testing: Theory and Applications (JETTA)*, 29(2), 193–209.

[32] S. Gupta, A. Crouch, J. Dworak and D. Engels, "Increasing IJTAG bandwidth and managing security through parallel locking-SIBs," *2017 IEEE International Test Conference (ITC)*, Fort Worth, TX, 2017, pp. 1-10.

[33] H. Liu and V. D. Agrawal, "Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key," *2015 IEEE 24th Asian Test Symposium (ATS)*, Mumbai, 2015, pp. 91-96.

[34] S. K. K, N. Satheesh, A. Mahapatra, S. Sahoo and K. K. Mahapatra, "Securing IEEE 1687 Standard On-chip Instrumentation Access Using PUF," *2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, Gwalior, 2016, pp. 56-61.

[35] R. Baranowski, M. A. Kochte and H. J. Wunderlich, "Fine-Grained Access Management in Reconfigurable Scan Networks," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 937-946, June 2015.

[36] A. Zygmontowicz, J. Dworak, A. Crouch and J. Potter, "Making it harder to unlock an LSIB: Honeytraps and misdirection in a P1687

network," *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, 2014, pp. 1-6.

[37] J. Backer, D. Hély and R. Karri, "Secure design-for-debug for Systems-on-Chip," *2015 IEEE International Test Conference (ITC)*, Anaheim, CA, 2015, pp. 1-8.

[38] L. Pierce and S. Tragoudas (2013). "Enhanced secure architecture for joint action test group systems." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(7), 1342–1345.

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

[40] M. Da Silva, M. L. 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*. doi: 10.1109/TCAD.2018.2818722.

[41] 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," *2017 22nd IEEE European Test Symposium (ETS)*, Limassol, 2017, pp. 1-6.

[42] M. Da Silva, M. L. Flottes, G. Di Natale and B. Rouzeyre, "Experimentations on scan chain encryption with PRESENT," *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*, Thessaloniki, 2017, pp. 45-50.

[43] M. A. Kochte, R. Baranowski and H. J. Wunderlich, "Trustworthy reconfigurable access to on-chip infrastructure," *2017 International Test Conference in Asia (ITC-Asia)*, Taipei, Taiwan, 2017, pp. 119-124.

[44] J. Dworak, Z. Conroy, A. Crouch and J. Potter, "Board security enhancement using new locking SIB-based architectures," *2014 International Test Conference*, Seattle, WA, 2014, pp. 1-10.

[45] R. Baranowski, M. Kochte, H. J. Wunderlich (2014). "Access Port Protection for Reconfigurable Scan Networks." *Journal of Electronic Testing: Theory and Applications (JETTA)*, 30(6), 711–723.

[46] X. Ren, F. P. Torres, R. D. Blanton and V. G. Tavares, "IC Protection Against JTAG-based Attacks," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. doi: 10.1109/TCAD.2018.2802866.

[47] S. Gupta, J. Dworak, D. Engels and A. Crouch, "Mitigating simple power analysis attacks on LSIB key logic," *2017 IEEE North Atlantic Test Workshop (NATW)*, Providence, RI, 2017, pp. 1-6.

[48] M. Da Silva, E. Valea, M. L. Flottes, S. Dupuis, G. Di Natale and B. Rouzeyre, "A new secure stream cipher for scan chain encryption," *2018 IEEE 3nd International Verification and Security Workshop (IVSW)*, Platja d'Aro, 2018.

[49] R. Baranowski, M. A. Kochte, H. J. Wunderlich, "Reconfigurable Scan Networks: Modeling Verification and Optimal Pattern Generation," *ACM Trans. Design Automation of Electronic Systems (TODAES)*, vol. 20, no. 2, pp. 30-30, 2015.

[50] M. A. Kochte, R. Baranowski, M. Sauer, B. Becker and H. J. Wunderlich, "Formal verification of secure reconfigurable scan network infrastructure," *2016 21th IEEE European Test Symposium (ETS)*, Amsterdam, 2016, pp. 1-6.

[51] P. Raiola, M. Kochte, A. Atteya, L. Rodriguez Gomez, H. J. Wunderlich, B. Becker and M. Sauer, "Detecting and Resolving Security Violations in Reconfigurable Scan Networks," *2018 24th IEEE International Symposium on On-Line Testing and Robust Design (IOLTS)*, Platja d'Aro, 2018.

[52] A. Tšertov et al., "A suite of IEEE 1687 benchmark networks," *2016 IEEE International Test Conference (ITC)*, Fort Worth, TX, 2016, pp. 1-10.

**Emanuele Valea** received his Master's degree in Electronic Engineering from the Politecnico di Torino – Italy – in 2016. He is currently a PhD student at LIRMM laboratory in Montpellier, France. His research interests include hardware security and trust, test of secure circuits and VLSI testing.

**Mathieu Da Silva** received the PhD in Microelectronics from the University of Montpellier – France – in 2018. He is currently employed as IoT Research Engineer at ITK in Montpellier. His technical interests include electronic testing, security and IoT devices.

**Giorgio Di Natale** received the PhD in Computer Engineering in 2003. He is CNRS Director of Research at TIMA laboratory in Grenoble, France. His research interests include hardware security and trust, fault tolerance and VLSI testing. He is Golden Core member of the Computer Society and Senior Member of the IEEE.

**Marie-Lise Flottes** is researcher at the French National Scientific Research Center. She has been conducting her research at LIRMM laboratory in Montpellier – France – since 1990. Her research interests include Design-for-Testability, Design-for hardware security and trust, with a focus since early 2000 on testability and fault tolerance on systems dedicated to secure applications.

**Bruno Rouzeyre** is Professor at the University of Montpellier, France. He conducts his research with LIRMM laboratory in Montpellier. His research interests include several aspects of CAD for digital circuits and are particularly oriented toward optimization, verification, test and test synthesis of digital and secure circuits.