Dynamic Authentication-Based Secure Access to Test Infrastructure

Michele Portolan, V. Reynaud, Paolo Maistri, Régis Leveugle

► To cite this version:

Michele Portolan, V. Reynaud, Paolo Maistri, Régis Leveugle. Dynamic Authentication-Based Secure Access to Test Infrastructure. European Test Symposium 2020, May 2020, Tallin, Estonia. 10.1109/ETS48528.2020.9131571 . hal-02887467

HAL Id: hal-02887467

<https://hal.science/hal-02887467>

Submitted on 1 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial| 4.0 International License

Dynamic Authentication-Based Secure Access to Test Infrastructure

Michele Portolan, Vincent Reynaud, Paolo Maistri, Regis Leveugle

Univ Grenoble Alpes, CNRS, Grenoble INP¹, TIMA, 38000 Grenoble, France

{name.surname}@univ-grenoble-alpes.fr

Abstract— The complexity of modern Systems-on-Chips is steadily increasing, which poses hard challenges for testing. In order to be able to face those challenges, several standards have been proposed through history, such as the latest IEEE 1687 on Reconfigurable Scan Networks (RSNs), which allows dynamic configuration of the test infrastructure for an easier access to embedded instruments and data. This ease of access, however, may constitute a serious threat from the point of view of security, as it may be used by an attacker as an entry point to the internal state of the circuit, especially if the test infrastructure is reused for life-time testing. Some approaches exist to protect the access, but their performances and security levels are limited by the legacy view of test as a static process. In this paper, we propose an innovative solution that exploits the dynamic nature of the IEEE 1687 standard to obtain an Authentication-based Secure Access framework able to provide a trusted and personalized interface to the test infrastructure depending on user-defined security levels.

Keywords—*Reconfigurable Scan Networks, Secure Access, Authentication, Automated Test Environments*

INTRODUCTION

Testing circuits after manufacturing and during their lifetime has become an increasingly important challenge for designers over time. The need for efficient and cost-effective ways to ensure that devices are fully functional, and the possibility to debug at any moment for diagnosis purposes, have led to the standardization of architectures and protocols allowing deep access into the circuits. Standards such as IEEE 1149.1 [1] and IEEE 1500 [2] were thus introduced, aiming to increase test performance, observability, and controllability. In order to face the increasing complexity of Systems-on-Chips (SoCs), those standards have continuously evolved. In recent years, IEEE 1687 [3] has introduced the dynamic reconfiguration of the scan chain, where specific segments can be included or excluded at will to target specific subparts of the circuit in the most efficient way.

On the other hand, these test facilities may create an important security backdoor into the circuit, which may be used by malicious users. Possible outcomes may consist in leakage of sensitive and critical data [4], illegal tampering of circuit behavior [5], or theft of Intellectual Property. Therefore, to seal this security breach it is mandatory to implement a protection layer over the test infrastructure. Few solutions currently exist [6][7][8], where the access to the test infrastructure is granted only after the User has successfully

performed some kind of authentication thanks to at least one secret key.

These solutions, however, are either vulnerable to simple threats such as replay or man-in-the-middle attacks, or require custom procedures which are difficult to implement in the classical Test Automation Flow, which effectively limit their usability in a real industrial scenario regardless of their technical merit. For widespread adoption, complete and transparent integration into the traditional generation and application test flow is paramount.

In this paper, we propose a comprehensive solution that provides transparent and efficient access to sensitive resources, balancing the need for accessibility with the imperative of security. Section I summarizes the current State of the Art: the novelties and potential of the IEEE 1687 standard are presented alongside the main security threats and the current solutions. In Section II, we provide an in-depth analysis of the shortcomings of the implementation of security in the legacy test flows, and present our solution to solve them. Section III presents the experimental setup we developed to prove the feasibility and measure the performances of the new approach, while lastly Section IV draws conclusions.

I. STATE OF THE ART

A. IEEE 1687: Dynamic Infrastructure and Execution Flow

Test is structured around a precise and well-structured Ecosystem, depicted in Figure 1: one or more Tools regroup information coming from the Design Phase with DfT information (such as ICL and PDL files for 1687) and use it to generate a Pattern Set, which is then used by dedicated Automated Test Equipment (ATE) to execute the test on the circuits issued from fabrication.

Figure 1: The Ecosystem of Automated Testing

¹Institute of Engineering Univ. Grenoble Alpes

In this scheme, all intelligence is regrouped in the Generation phase, while Patterns are simply a collection of input vectors/expected outputs to be executed by the ATE. This process is optimized for factory testing, where the Key Performance Indicator is speed: testing time must be minimized to reduce costs. In this context, the IEEE 1687 standard brings two important novelties: the introduction of dynamic-length scan chains and the native support for functional operations.

The first aspect is allowed by the introduction of “ScanMux”es in the chain, usually regrouped in what is called a Segment Insertion Bit (SIB): by modifying the value of the controlling Scan Register, it is possible to add or remove elements (usually called Segments) from the scan chain. The resulting topologies are much more complex than classical scan chains, and are therefore usually referred to as Reconfigurable Scan Networks, described in the new Instrument Connectivity Language (ICL) defined by the standard.

The second aspect, i.e. native functional instrument operation, is enabled in the standard by the Procedural Description Language (PDL), and is potentially much more disruptive. In its simplest form, called PDL-0, it simply replicates vector operations at register level and is well adapted to the flow represented in Figure 1. In its full form (PDL-1), it allows interactive behavior: data can be collected from the System Under Test, and used inside algorithms in an overlay language (e.g., TCL) to both modify the execution flow and decide the new data to be sent to the System Under Test (SUT). This interactive loop is not possible inside the legacy flow of Figure 1: Pattern Languages, in the middle of the flow, have only limited provisions for flow control and dynamic data handling.

While most EDA companies have been focusing on efficient usage of SIB-based topologies to optimize pattern size and application time, little work has been done on interactive PDL execution. One of the few exceptions is the “Manager for System-Centric Test” (MAST), presented in [9] and depicted in Figure 2.

Figure 2: 1687 MAST Functional Access Flow [9]

Instead of generating a static Pattern Set, MAST implements a rich Execution Backend able to implement several features usually reserved to the Generation step [10]: the tool maintains a Model of the internal state of the SUT and uses it to dynamically generate the vectors only when needed. This Model incorporates topology-enabling elements such as the SIB, whose configuration is handled internally by the tool with no user intervention. By exploiting MAST, a User is able to provide his own PDL-1 algorithm (depicted in the top right-hand corner of Figure 2), and have it executed on the SUT in a completely automated way, regardless of the internal DfT infrastructure.

B. Scan Chain Securization: Locking SIBs

The security potential of the SIB has been recognized quite early: the Locking Segment Insertion Bit (LSIB) [6], depicted in Figure 3.a, needs a specific binary condition to unlock the access to the lower connected scan segment.

Figure 3: Locking SIB and Secure SIB Implementations

Access to the segment is therefore not automatic, but needs a secret Boolean condition to unlock the access. This key is spread and hidden inside the DfT infrastructure so that, in principle, an attacker would thus need to know its value and exact position to unlock the LSIB. In practice, however, the key will be stored in plain text inside the Pattern Set: this setup is vulnerable to Replay or Man-in-the-Middle type of attacks, where the key can be inferred from the data.

C. Scan Chain Securization: Fine Grained Access Management

When using an insecure channel, such as a scan chain, the key should never be transferred in plain text. The Fine-Grained-Access (FGA) solution [7] uses a challenge-response protocol to perform the authentication before granting access

to a sensitive scan chain segment. In this scheme, the two parties share a secret, used to generate a one-time transaction token, while the secret keys themselves are never transmitted on the insecure medium, making replay attacks ineffective.

Figure 4: FGA Challenge response protocol

Furthermore, FGA is also able to provide a personalized access to the reconfigurable scan network by using instrument-level keys: users have to know the key for each instrument they want to access. The protocol is based on few critical points, which are highlighted in the following list:

- **Configuration vector:** each secured instrument is represented in this vector by a dedicated bit, which needs to be set to 1 in order to target (access) the corresponding instrument.
- **Challenge:** a value generated randomly by the system, specific to the current session.
- **Challenge resolution:** the process where the user and the controller are performing the cryptographic computation in parallel. The challenge is concatenated with the key of the first instrument targeted in the configuration vector and the obtained vector is then processed by a hashing primitive (such as SHA256); if other instruments are targeted, the procedure is repeated for each instrument, using the previous result as input, until the final response is obtained.

The authorization controller checks if the user's response is correct; it then allows the user to access the secure scan chain. Details on the practical implementation can be found in [7]. The unique purpose of this secure scan chain is to propagate the opening authorization to SIBs using the specially introduced Secure Segment Insertion Bit (S²IB) visible on Figure 3.b.

D. Scan Chain Securization: Segment Set Authorization Keys

In [8], the authors propose some modifications to the FGA approach with the objective to allow usage of reprogrammable memory for the secret key and reduce the average authentication time.

To obtain a faster authentication, the instrument keys are replaced by configuration keys or Segment Set Authorization Keys (SSAK). These new keys can resolve any authentication in just one cryptographic computation. However, the number of theoretical valid keys becomes exponential to the number of instruments: if we consider n as the number of instruments, the number of valid SSAKs would be equal to 2^n , whereas the number of FGA instrument keys would be just n . Furthermore, the usage of read/write memory for such a number of keys would be too expensive, so the authors propose the procedural generation key mechanism presented in Figure 5. All the configurations keys of a circuit instance are dynamically generated from the unique secret key of the circuit and the related configuration vector. This configuration vector has to contain the list of targeted instruments and can also contain an identification number if the keys need to be different for each user. The generation consists in an encryption of the configuration vector using the circuit key as encryption key. The security provider can distribute credentials composed of configuration and SSAK to the users. On the other side, the Circuit Key is securely stored in the reconfigurable memory of the Authorization Controller.

Figure 5 SSAK's procedural key generation and distribution

Concerning the authentication protocol, only the challenge resolution is performed. The controller receives the configuration vector from the user, and then thanks to the procedural key generation, it is able to compute the associated SSAK, using its embedded encryption processor. Then, still using the same encryption hardware, the controller can resolve the challenge with this SSAK. On the user side, the process is easier as the SSAK is already known, so the user only needs to encrypt the challenge with the key contained in the credentials.

Once the authentication is done, the controller needs to unlock the S²IBs targeted by the user. Figure 6 shows the scheme of the SSAK solution architecture: the different S²IBs are linked together by the so-called secure scan chain, driven by the authentication controller.

Figure 6: SSAK Authentication architecture

The advantage of this approach consists in a faster challenge resolution, as the user only needs one cryptographic operation. The protocol allows for different cryptographic primitives such as symmetric ciphers or hashers, thus supporting a large choice of algorithm implementations.

E. Summary and Open Issues

In terms of hardware overhead, the LSIB approach is flexible and less expensive in case of small implementations; it is also compatible with the traditional Test Flow as its usage can be expressed in terms of plain vector operations. However, its overall security level may be insufficient due to the plain key exchange.

FGA and SSAK solutions have both a more secure protocol for authorization. In addition, they are quite efficient even for a large number of protected instruments. As the number of instruments increases, SSAK becomes clearly the more efficient and secure alternative, thanks to the constant authentication overhead.

From this data alone, FGA and SSAK seem to have an overwhelming advantage. But this comparison does not take into account a fundamental point: the dynamic nature of the challenge/response protocol cannot be expressed in terms of a Pattern Set, making these approaches incompatible with the legacy Automated Test Ecosystem. As a result, their actual implementation requires a significant and custom development by the user, effectively limiting their applicability.

II. AUTOMATED SECURITY PROTOCOL

In this section, we propose an automated setup that is able to fully exploit the security features of the SSAK authentication scheme. First, Subsection A will introduce the Use Case and highlight the problems and limitations of authentication implemented using legacy approaches. Subsection B will then introduce a preliminary solution, where the protocol is completely implemented in terms of 1687 functions, but which still implies an active role for the User. Lastly, Subsection C will present our fully automated solution.

A. Problem Setting: Legacy Implementation

The Use Case is depicted in Figure 7: User A wishes to run a PDL Algorithm on an embedded Target, which is part of an SSAK-controlled Secure Section of the DfT infrastructure.

Figure 7: Secure Access Use Case

A direct implementation of the legacy Testing Ecosystem would result in the setup of Figure 8: while the PDL code can be automatically translated into a Pattern Set by standard EDA Tools, the User would still need to wrap it with the SSAK Authentication protocol.

Figure 8: Legacy SSAK implementation

This development would be specific to the given test host (ATE, embedded controller, etc.), with little or no portability. Moreover, User A would need a detailed knowledge of the Authentication scheme to implement it.

B. PDL-1 based Authentication Solution

As explained in the previous paragraph, the main portability issue for the implementation of a challenge/response protocol is the impossibility for the classical Automated Test Flow to handle interactive execution in a portable way. For this reason we decided to implement the authentication scheme using the MAST tool, obtaining the setup of Figure 9.

Figure 9: PDL-1 Authentication flow

The Security Provider, in the top half, defines the authentication strategy and is responsible for both implementing the Secure Section in hardware and providing a PDL-1 wrapper taking care of the challenge/response protocol.

Thanks to MAST, the wrapper can mix C++ algorithms with PDL-1 operations to access the internal registers of the SSAK controller. User A simply needs to insert his own code inside the wrapper and provide it to MAST for execution. This scheme presents several advantages over the legacy solution:

- Both the hardware and software parts of the protocol are implemented by the same person;
- User A does not need an active knowledge of the SSAK algorithm to use it;
- As everything is implemented in 1687 terms, the solution does not depend on the execution host and is therefore fully portable.

This setup is a significant improvement over existing approaches both in terms of ease of implementation and portability. Anyway, it still presents three critical points:

- Authentication is not completely transparent from the User's point of view, as an explicit wrapping step is still needed;
- Security is sub-optimal: the Wrapper is provided as source code, so a malicious User might access the secret key through reverse-engineering and gain access to segments that are not part of his SSAK set.
- As SSAK controls one or more S²IB, the User code might interfere with MAST internal topology resolution algorithms [9]

C. Fully Automated Authentication

To obtain a solution that is really both flexible and secure, the User must not have access to the Authentication code, and should only have access to the part of the shared secret he is allowed. We therefore developed the setup depicted in Figure 10, inspired by state-of-the-art protocols such as SSH [11].

Figure 10: Fully Automated Authentication

In this Setup, the Security Provider implements the SSAK protocol as an extension library for MAST: whenever the tool needs to access a segment that is in the Secure Section, it will trigger an authentication transaction. The user will just receive a Certificate composed of the SSAK keys for the Secure targets he is allowed to access [8], which will be passed by MAST to the SSAK Protocol library. If the challenge is successful, MAST will grant access to the secure section, otherwise it will raise a Security Exception. This exception is informative to update correctly MAST internal model and

notify the user; it is not able to affect the SSAK controller, to which MAST has no direct access.

This scheme is optimal from several points of view: security-wise, secret sharing is limited to a minimum, while the challenge/response protocol ensures no plain-text key transfer. From a performance point of the view, the integration of the protocol inside MAST allows the tool to fully leverage its strength in terms of topology resolution and concurrent execution. Last but not least, User experience is also optimal, as security is handled automatically and transparently.

III. EXPERIMENTAL SETUP

As a proof-of-concept, we implemented the two proposed approaches for the System Under Test of Figure 6 on two representative platforms:

- A bench-top environment, where an FPGA-implemented 1687 system is accessed through an USB-to-JTAG dongle;
- An embedded environment, where a 1687 system implemented in the Programmable Logic of a Xilinx Zynq SoC is accessed from the embedded ARM processor.

The aim is twofold. On one hand, we want to show that the test of a circuit secured by [7] or [8] is improved by the usage of MAST. On the other hand, we want to prove the flexibility of the approach by porting it to a different setup with little or no user intervention.

In reference to Figure 6, we consider that the User needs to execute different PDL codes on the SSAK-protected Instruments 1 and 3, and on the non-protected Instrument 2. The functions will be called I1_PDL, I3_PDL and I2_PDL respectively.

In the Authentication flow of Figure 9, the User wraps both I1_PDL and I3_PDL using the PDL-1 template given by the Security provider. The resulting I1_PDL_Wrapped and I3_PDL_Wrapped functions are inserted in the MAST execution flow. As both Wrappers imply different configurations to the SUT's internal topology, they need to be executed separately to avoid contention over SIB configuration or SSAK authentication. The resulting execution flow is depicted in Figure 11: execution of PDL code in the Secure Section is performed correctly, with Authentication (marked in green) happening at the beginning and end of each wrapper. Non-protected code is run at the end, when the Secure Section has been closed. The timeline is extracted from MAST log file, and simplified for easier interpretation.

T	0	2	4	6	8	10	12
I1_PDL_Wrapped							
I13_PDL_Wrapped							
I2_PDL							

Figure 11: PDL-1 Authentication flow timeline

Even though the execution is correct, it is far from optimal as the access to the Secure Section did not allow MAST to

execute the three codes concurrently [9]. Also, SSAK allows to authenticate and open several Secure Segments in a single operation [8], but we cannot exploit this feature because it would imply a manual merge of I1_PDL and I3_PDL in a common PDL function.

We then implemented the Fully Automated Authentication of Figure 11 by implementing the SSAK algorithm inside the Configuration phase of the MAST tool [9]. We then executed two cases:

- Full Permission, where the User has a SSAK certificate allowing access to both I1 and I3;
- Partial Permission, where the User has a SSAK certificate allowing access to I1 only.

The resulting executions are resumed in Figure 12: the advantages in terms of optimization and execution time are obvious for the Full Permission case (a). By implementing SSAK inside its own configuration algorithm, MAST is able to parallelize the execution of PDL algorithms in the Secure Section and in the Normal Section, obtaining a significant execution time gain.

In the Partial Permission use case (b), MAST is able to differentiate between legitimate and illegitimate access: MAST correctly detects an authentication error when trying to access Instrument 3, denies its execution and raises a security exception (depicted by the red case at timestamp 3), without perturbing the execution of code in Instrument 1 and 2, as can be seen by the green box at timestamp 4. Depending on the security scenario, this error might be kept silent or escalated, once more without any intervention by the User.

a) Full Permission use case

b) Partial Permission Use case

Figure 12: PDL-1 Authentication flow timeline

As MAST can run on both Intel-based PCs and ARM embedded cores, the three versions of the Use Case were run on both Desktop and Embedded scenarios with equivalent results simply by adapting the compilation chain. The change of platform was completely transparent for the User, demonstrating the flexibility of the platform.

IV. CONCLUSION

Testing and Security have been historically two antithetic domains: where the former strains for largest observability and controllability, the latter aims at the opposite. Solutions for their coexistence are appearing, aiming at building a bridge between the two, but regardless of their scientific and technical value, their applicability in real-life scenarios is limited by their poor compatibility with legacy Execution Flows.

In this paper, we demonstrated how it is possible to provide a comprehensive solution in the framework of the IEEE 1687 standard that leverages state-of-the-art solutions from both Test and Security, eventually providing a transparent, configurable, and efficient solution to this problem.

ACKNOWLEDGMENTS

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

REFERENCES

- [1] IEEE Std 1149.1-2001, "IEEE Standard Test Access Port and Boundary-Scan Architecture", IEEE, USA, 2001.
- [2] IEEE std 1500 - Standard for Embedded Core Test - <http://grouper.ieee.org/groups/1500/>.
- [3] IEEE Std 1687-2014, "IEEE Standard for Access and Control of Instrumentation Embedded within a Semiconductor Device", IEEE, USA, 2014
- [4] Sk Subidh Ali, Samah Saeed, Ozgur Sinanoglu, Ramesh Karri. New Scan-Based Attack Using Only the Test Mode and an Input Corruption Countermeasure. 21th IFIP/IEEE International Conference on Very Large Scale Integration - System on a Chip (VLSI-SoC), Oct 2013, Istanbul, Turkey. pp.48-68, ff10.1007/978-3-319-23799-2_3ff.
- [5] Mark Barnes, "Alexa, are you listening?", MWR Info Security, <https://labs.mwrinfosecurity.com/blog/alexa-are-you-listening/>, Aug 1st 2017.
- [6] Jennifer Dworak, Al Crouch, John Potter, Adam Zygmuntowicz, Micah Thornton, «Don't Forget to Lock your SIB: Hiding Instruments using P1687», IEEE International Test Conference, 2013.
- [7] B. Rafal, K. Michael A and H.-J. Wunderlich, "Fine-Grained Access Management in Reconfigurable Scan Networks," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, pp. 934-947, 2015.
- [8] Marc Merandat, Vincent Reynaud, Emanuele Valea, Jerome Quevremont, Nicolas Valette, Paolo Maistri, Regis Leveugle, Marie-Lise Flottes, Sophie Dupuis, Bruno Rouzeyre, Giorgio Di Natale, "A Comprehensive Approach to a Trusted Test Infrastructure," in Internation Verification and Security Whrokshop, Rhodes 2019.
- [9] M. Portolan, "A Novel Test Generation and Application Flow for Functional Access to IEEE 1687 instruments", Proc European Test Symp. (ETS), pp. 1-6, 2016.
- [10] M. Portolan, "The Automated Test Flow, the Present and the Future", IEEE Transactions on Computer-Aided Design (TCAD), DOI: 10.1109/TCAD.2019.2961328, December 2019
- [11] OpenSSH homepage, <https://www.openssh.com/>, regularly updated.