SEcube™: an Open Security Platform - General Approach and Strategies

Antonio VARRIALE[1], Giorgio DI NATALE[2], Paolo PRINETTO[3], Bernhard STEFFEN[4], Tiziana MARGARIA[5]

[1] Blu5 Labs Ltd, Blu5 Group, Ta Xbiex, Malta – av@blu5labs.eu
[2] LIRMM, CNRS, Montpellier, France – giorgio.dinatale@lirmm.fr
[3] CINI Cyber Security National Lab & Politecnico di Torino, Torino, Italy – paolo.prinetto@polito.it
[4] Chair of Programming Systems, TU Dortmund, Dortmund, Germany – bernhard.steffen@tu-do.de
[5] University of Limerick and Lero, The Irish Software Research Centre, Limerick, Ireland – tiziana.margaria@lero.ie

Abstract *The SEcube™ (Secure Environment cube) platform presented in this special track is an open source security-oriented hardware and software platform constructed with ease of integration and service-orientation in mind. Its hardware component is a SoC platform: a single-chip design embedding three main cores: a highly powerful processor, a Common Criteria certified smartcard, and a flexible FPGA. The software components include several libraries of ready-to use components that provide developers with different entry levels to adoption. The software is modular, and available as API or as services in an advanced model driven design environment. This way, security experts can avail of the open source character, and verify, change, or write from scratch the entire system, starting from the elementary low-level blocks, but at the same time we support also developers who use the predefined primitives and can experience the SEcube™ as a high-security black box. This paper explains its aims and architecture, while the other papers detail the unique aspects of the overall platform.*

Keywords — Security, Hardware Platform, Software Development Environment, Model Driven Design, Open-Source.

1 Introduction

Security is a key concern to any mission and business critical service and application. Wherever there is data or proprietary or personal content there is a need of mechanisms and provisions in place to ensure adequate users' privacy, prevent and handle the commercial and legal issues related to security threats, and to safeguard the business stakeholders.

The implementation of a suitable security layer usually requires special skills in several disciplines, including mathematics and cryptography. When the security target is provided as a combination of hardware and software solutions the system complexity increases and further skills are required (i.e. electronics, physics, informatics) to manage the integration and prevent all the possible attacks. At the same time the security may have a big impact on pre-existing systems and solutions, most of the times forcing the final users to change their habits. All these aspects make the security a very complex topic from the development stage to the final usage.

In this paper and track we describe the architecture and the main characteristics of the SEcube™ (Secure Environment cube) platform, that is new in being consequently open and service-oriented. It is easy to integrate and capable of hiding significant complexity behind a set of simple and high-level services, that are accessible as APIs, the lowest level being the hardware itself. Also the hardware is open: its description and technical details are completely available, including the hardware netlist, the package, examples of schematics in order to plug the chip into possible boards. The complete software stack is open source as well, including libraries of services at different abstraction level offering palettes of pre-defined services, and the service APIs, up to the model driven development environment itself, that that enables the easy creation of new (secure) applications and products.

While the open platform concept is popular in the software domain (think of Linux, Java, and its spread in software development communities, like in bioinformatics), it is not yet commonly adopted for hardware, especially for security purposes, where the solutions are usually provided as a black box sometimes coming with a certification and always closed to any kind of disclosure or customizations.

There are a few security-oriented open platforms available on the market. Some of them are focused on the evaluation of the system robustness against external physical attacks (e.g., Side-Channel attacks, power crypto-analysis, etc.), such as the Sasebo board [1] and the ChipWhisperer [2].

Other platforms based on ARM processors, like Juno ARM Development Platform [3] and the open source USB device provided by InversePath [4], allow creating general purpose software applications, including security-oriented solutions. Nevertheless, they are based on application processors and there are not specific security elements to be fully controlled or customized by the developers.

Finally, there are single chips realized as a combination of one FPGA and one CPU, like Zynq proposed by Xilinx [5] or Excalibur based on Altera technology [6]. Nevertheless, in both the cases the platforms are more suitable at prototyping stage, since they are not cost-efficient, and a specialized security element, like a smart card, is still missing.

This paper describes in detail the SEcube™ hardware (Section 2) and software (Section 3) architecture, gives a brief introduction to the DIME environment that provides the eXtreme Model Driven Development design and verification capability (section 4), and concludes in Section 5 with an introduction to the other papers of this track. They provide in-depth descriptions of the security primitives and protocols, the design environment, several case studies, a first glimpse of how to deal in this overall context with property verification and enforcement.

2 The SEcube™ Hardware Architecture

As shown in Fig. 1, the hardware device is a single chip, which embeds three hardware components: a powerful CPU, a flexible FPGA and an EAL5+ certified smart card. The software library is developed as a free and open-source SDK, provided with user documentation.

Internally, the SEcube™ device is a multi-module chip integrated in a 9mm x 9mm BGA package. As shown in Fig. 2, it is a heterogeneous platform consisting of three elements.

The first element is a high-performance ARM Cortex M4 RISC CPU produced by ST Microelectronics [7]. It provides the following features:

- 2 MBytes of Flash memory
- 256 KBytes of SRAM
- 32 bit parallelism
- operating at frequency of 180 MHz
- dedicated FPU (Floating Point unit)
- Internal TRNG (True Random Number Generator)
- Hardware Crypto Accelerator
- Low power consumption

This CPU has been selected among many ARM based micro-controllers, since it offers several features that make it suitable for high-performance and security-oriented solutions. For example, it supports the Cortex CMSIS implementation that provides, among the others, the CMSIS-DSP libraries: a collection with over 60 DSP functions for various data types. The CMSIS-DSP library allows developers to implement complex, real time operations using the embedded harware floating point unit.

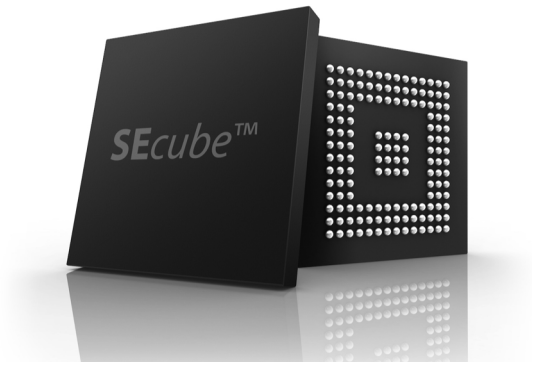

Fig. 1 The ***SEcube™ Chip***

In addition, the CPU provides several peripherals such as SPI, UART, USB2.0 and SD/MMC, which ease the hardware integration in diverse devices. For example, a secure USB device can be easily realized using the USB2.0 and the SD card interfaces, respectively.

On the security side, a TRNG (True Random Noise Generator) embedded unit, hardware mechanisms like MPU (Memory Protection Unit), and privileged execution modes allow implementing the security strategies required by a certified secure controller (e.g., privileged memory areas, key generation, etc.).

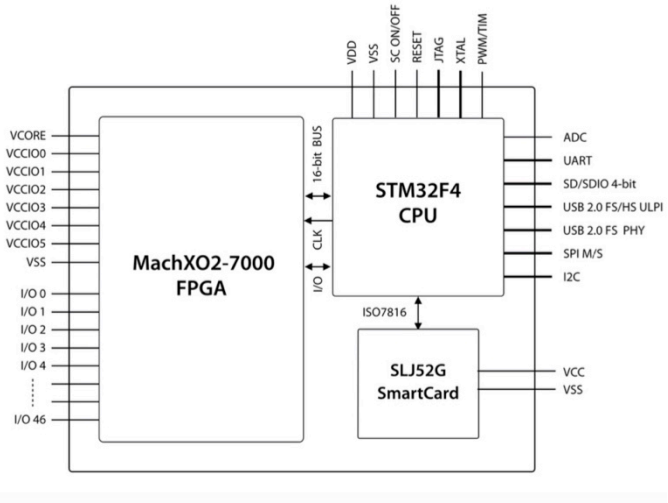

Fig. 2 ***SEcube Hardware Architecture***

For programming, debug and testing operations, the CPU provides a standard JTAG interface that can be permanently disabled once the development cycle is over, protecting all the sensitive information through a physical hardware lock.

The FPGA element, a Lattice MachXO2-7000 device [8], is based on a fast, non-volatile logic array providing the following main features:

- 7000 LUTs
- 240 Kbits embedded block RAM
- 256 Kbits user flash memory
- Ultra low-power device.

The FPGA exposes 47 general purpose I/O which may be used as a 32-bit external bus able to transfer data at 3.2 Gb/s. Inside the chip, it is linked to the CPU through a 16-bit internal BUS, able to reach a data transfer rate of 1.4 Gb/s. A CPU-FPGA clock line is also provided in order to simplify the clock domains synchronization.

In order to limit the number of pins and the BGA package size, the FPGA JTAG is connected just to the embedded CPU, which manages both the debug and the programming operations. As a positive side effect, the FPGA configuration can be implemented by means of customized, high-security techniques. For example, the programming stream can be encrypted and signed through dedicated algorithms. The CPU and/or the smartcard elements can decrypt and verify it before being injected in the FPGA.

The third component inside the SEcube™ device is an EAL5+ certified smartcard [9], based on a secure chip produced by Infineon that provides the following features:

- ISO7816 interface
- JavaCard Platform, Global Platform 2.2
- 128 KB Flash
- EC, ECDH up to 521 bit (HW accelerator)
- RSA up 4096 bit (HW accelerator)
- AES128/192/256 (HW accelerator).

As shown in Fig. 2, the CPU is connected to the embedded smartcard through a standard ISO7816 interface. The smartcard does not expose any interface outside the SEcube™ chip. This architectural decision provides high-grade and certified security functionalities behind a simpler and easy-to- use application interface.

Combined together, the above three components, allow to build a heterogeneous computing architecture and create the foundations to build a very flexible open source security platform.

2 The SEcube™ Software Libraries

The software side of the platform library consists of a multi-level, open source, collection of libraries available as an SDK [10, 11], together with a verification-oriented model driven design environment (currently DIME [12]). The libraries, especially if in conjunction with this design environment, allow developers who are not willing or able to produce the security primitives and protocols themselves to exploit the ready functions provided (as services or API) within the SEcube™ platform and experience the platform as a high-security black box. Conversely, security experts in the security domain can enjoy the openness and good documentation to verify, change or rewrite the pre-existing software starting from basic low-level blocks or even redefine entirely the whole system.

Leveraging the platform thought, we intend to create and nurture over time a community for developers at the different levels of security competence and in different application domains. This will ease the project, knowledge, and resource sharing and provide the collectivity of members with specialized support tailored to their needs.

From the architectural point of view, the software is divided in two main parts, depending on where physically the code runs: the Device-side relates to the SEcube™ based hardware device (e.g. USB secure token), while the Host-side relates to the appliance hosting the device (e.g. Laptop).

In this scenario the SEcube™ hardware device acts as a powerful coprocessor which provides a secure and fully controlled execution environment. All the functionalities implemented in the SEcube™ are thus exported to the host system through an open source secure RCP (Remote Call Procedure) protocol which is encapsulated in the SDK.

Before describing all the functional details (section 4), a general device and host side overview is given here.

2.1 Device-Side Software

The device side, software provides the libraries of basic functionalities that are executed on the embedded microprocessor. According to their purposes, the libraries cover three layers:

Functional *Layer0*. Closest to the device, **software drivers** offer basic functionalities to manage and access internal peripheral (e.g. TRNG, internal Flash memory, timers, etc), external devices connected to the CPU, i.e., FPGA and Smart Card, and the external communication interfaces as well (i.e. USB, UART, SPI and GPIOs). This level, in particular, is entitled to discover possible SEcube™ devices connected to the host system and create a bidirectional communication channel. These functionalities are exposed through a simple set of APIs:

- `L0_get_dev_list`, which is in charge of discovering connected SEcube based devices, and
- `L0_tx_rx`, which implements a low level send/receive operation on the channel.

Functional *Layer1*. At the intermediate level, the **core functions** constitute the basic primitives for implementing secure applications. This layer provides basic cryptographic algorithms and various utilities, like functions for power management.

In addition, it is used to secure the communication channel after a successful login, which can be easily implemented as a multi-factor authentication thanks to the three hardware elements inside the chip and the password provided by the user.

The Layer1 exposes several functions to manage both the device provisioning and the user/admin operational processes:

- pin and primary keys initialization (e.g., `L1_factory_init` and `L1_initLogin`)
- login (e.g. `L1_loginAdmin`, `L1_loginUser` and `L1_changePinUser`)
- logout (e.g. `L1_logoutAdmin` and `L1_logoutUser`)
- information retrieval (e.g., `L1_readDSN`, `L1_getAlgorithms`, and `L1_getKeyList`)
- encryption/decryption/signature verification (e.g., `L1_crypto_init`, `L1_crypto_setIV`, `L1_crypto_update` and `L1_crypto_finit`)
- key management (e.g., `L1_injectKey` and `L1_deleteKey`).

Functional *Layer2*. At a higher level, the **security abstraction layer** allows developers to create secure software and services for the applications (e.g., secure file system [10]), avoiding the need to understand in detail the low-level hardware and security mechanisms.

When the security target does not require a hardware implementation, the SEcube device can be virtualized by a software library which provides all the layers above running on the host.

2.2 Host-Side Software

On the host side, the software is tailored for existing devices (e.g., laptops or Desktop PC) that see the SEcube hardware as an external peripheral and use it for the specific functionalities offered, like cryptographic hardware acceleration. For this communication, the SEcube device is seen as a closed black box providing utility services. The host starts the service request by sending the related command and the optional data packets, through a proper interface, according to a custom protocol.

Also the host side code is open source. It is designed to be both scalable, e.g., for dealing with multiple devices, and portable on different operating systems, thus limiting the usage of and isolating platform-dependent modules.

The host-side software runs on top of the host operating system, and I structured in two main levels.

Layer0 implements the basic functionalities to communicate with the SEcube, including (a) sending/receiving command and data packets from/to the device, (b) segmentation of raw data streams into protocol-compliant packets, (c) functions implementing standard cryptographic algorithm and (d) low-level error management functions. Moreover, commodities functions are also provided, such as low-level data manipulation in dealing with possible endianness mismatches between the host side and the SEcube™ embedded CPUs.

Host-side software relies directly on the Operating System calls and it supports many OSs and platforms, including Microsoft Windows, Unix-like environments, and MacOS. To improve portability, OS-dependent sub-modules (e.g., communication interface, file system, etc.) are easily identifiable.

Layer1 is built over the Layer0 and provides a higher abstraction library, like multi-factor login, secure communication channel, cryptographic algorithms and key management. As we see in Fig. 3, currently, the Layer 1 library includes 16 functions, of which 6 concern login/logout (called `log` in the picture, which is their abstract type) 4 the key management (`key`), 4 the cryptography (`crypt`) plus 2 utility functions (`utils`). These functions can be combined to implement more complex security mechanisms at higher level, as detailed in [12,] [13], and [14].

The Layer1 also allows developers to manage several SEcube™ devices at the same time, providing dedicated operation control flows (one command/response session per communication channel), which allow encoding and decoding commands for the individual SEcube™ target.

As shown in [10] and [11] and sketched in the next section, the Layer 1 services are the basic building blocks for developers to create complex and tailored security primitives. As shown in [12] and [14], design of secure solutions based on these services eases the understanding and reduces the time to market drastically. The key to this speed and increased confidence in the correctness and security of an application is a model-driven design that includes these ready but customizable security services and protocols already at design time, on the models of any application. How the DIME environment supports this is explained in the next section.

3 Modeling Environment

DIME (the DyWA Integrated Modeling Environment) is an integrated solution for rigorous model-driven development of sophisticated and high assurance web applications. They are modeled in a simplicity-driven fashion that focuses on describing *what* application is sought (descriptive), instead of *how* the application is realized (prescriptive). Further design goals are agility and security as well as quality assurance. It is a consequent refinement of the realization of jABC4 (Java Application Building Center 4, [15]) for process modeling and DyWA (Dynamic Web Application [16]) for domain modeling and data persistence. The application workflow models in DIME are graph models: nodes are called SIBs, and represent executable functionalities, whose labeled outgoing edges, called branches, lead to the logically next appropriate SIB to execute.

In the spirit of its predecessors, DIME empowers prototype-driven application development following OTA (One Thing Approach [17]) and XMDD (Extreme Model-Driven Design [18]) by putting Subject Matter Experts (potentially non-programmers) in the core of the development process. Hence, different aspects of an application are described with the respective most adequate form of model. All these models are interdependently connected, collectively

shaping the One Thing model in a very formal yet easy to understand way. This is supported to the extent that the application can be one-click-generated to a running product.

The models created with DIME are transformed into code in a generation step where the complete target application is assembled according to the model's control flow and to the code of the elementary blocks, called SIBs. The target of this product generation is the DyWA framework, which constitutes the actual runtime environment, supports the deployment phase, and manages data persistence. However, the runtime platform, programming language, and frameworks are a matter of the corresponding (full) code generator, and may be changed without touching the models.

Consistent model-driven design together with the generative, service-oriented product assembly provide users with early prototyping of executable web applications as well as explicit support for product evolution, this due to the agile nature of version management for data management and persistency in DyWA. Altogether, the approach has the potential to tremendously push development cycles in an agile but consistent manner.

For model design, a family of Graphical Domain-specific Languages (GDSL) is tailored to express specific aspect of typical web applications:

- **Data models** cover the design of domain models based on common concepts like classes, attributes, and uni- or bi-directional relations between elements.
- **GUI models** specify the structure of (re-usable components of) web pages and the data binding in data sensitive user interface components.
- Different **Process models** types span the core business logic, data retrieval (search queries) as well as dynamic access control (security guards, particularly interesting for this platform).

Relations between these aspects are modeled by cross-referencing the models, this way creating hierarchical model structures.

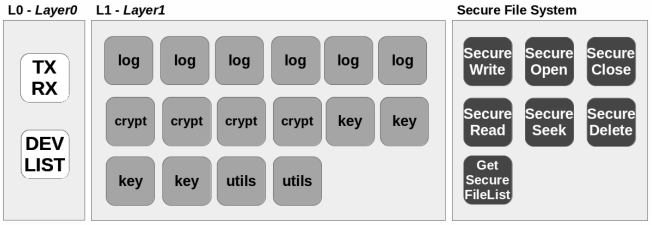

Fig. 3 ***L0, L1, and SFS SIB palettes for the SEcube***

Such models are constituted by connected basic model components called SIBs (Service Independent Building Blocks) that either encapsulate other existing models or link to implementations, in form of code, or some form of API calls, e.g. services, RPC, other local or remote libraries. SIBs are the units of model re-use, and are well suited to represent library elements. For instance, the SEcube Layer0 and Layer1 API collections are seen in DIME's **Diagram Editor** as SIB palettes. Fig. 3 shows this for the Layer0 and Layer1 services, as well as for the Data at rest palette (Secure File System) of services described in [10], with a graphical representation of the SIBs within each layer. Each SIB in the L0 and L1 palettes corresponds to one of the functions discussed in the previous Section, with the L0 and SFS system SIBs spelled out and the L1 SIBs labeled according to their abstract types in DIME.

The inner logic flows of complex SIBs, once designed, are process models, represented as SLGs in DIME. The collection of available process models and the creation of cross-references are found in the **Models View**. The **Data View** lists data types and type attributes in a structured fashion. The **Properties View** deals with attributes and parameters. The **Model Validation View** manages the syntactic and semantic checks that provide guidance for the user and ensure correctness wr.t the current set of properties, as discussed in more depth in [14].

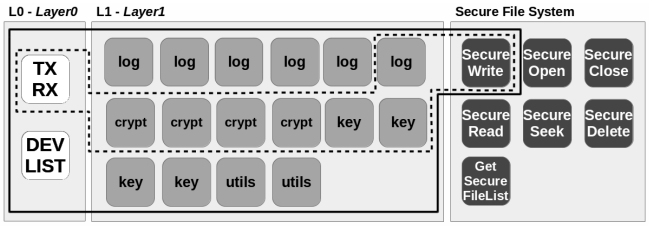

Fig. 4 ***Uses(x)*** *relation of the* ***SecureWrite*** *SIB of the Secure File System (SFS) palette*

Fig. 4 shows the subsets of L0 and L1 palettes used to implement a Secure Write operation within the Secure File System palette of the SDK described in [10].
As frequent for higher level functions, the internal behaviors of the Secure Write SIB are context dependent, and use different sets of functionalities in different contexts. This information and the ability to analyze and visualize it easily are important for any design decision that propagates across functionalities. For instance, when performing an impact analysis of changes in the underlying platform libraries. the Uses(x) relation is useful: similar to a call graph, or to a reachability analysis at the SIB level, Uses(x) for a given SIB returns the set of SIBs that occur in its Service Logic.

- The solid line encloses the SIBs used when the device is first inserted, **at connection time**. In this case, the logic flow internal to the Secure Write operation foresees that the device needs to be found using the L0 `get_DEV_LIST` SIB, then the L1 login procedures must be executed, followed by cryptographic and key handling functions, using also the utility functions. The communication happens all the time via the L0 transmit/receive functionality `TX_RX`
- For further SecureWrite operations carried out **after the first** one, the login has already taken place and the chip

has already the correct key set. In this case the logic internally executed is simpler, and uses only the SIBs enclosed in the dotted line. In this case, they are mostly cryptographic functions, and of course the L0 transmit/receive functionality `TX_RX`.

Already this simple example shows how the inner workflows of otherwise quite elementary operations can become orchestrations in the secure case, and how it may be useful to be able to guarantee properties of those flows, especially if context plays a role.

Throughout the development life-cycle, DIME design-time well-formedness and consistency, enabling one-click-generation and direct deployment of a sound and secure web application at any time. Continuous change and evolution are supported by means of iterative model modification, re-generation and re-deployment.

4 This Track

Addressing the path to End-to-end Security and cyber security, from the hardware to application, this special track acknowledges that security and cybersecurity are an increasing concern for all the actors in the IT and societal space. One of the limiting factors to integrated security is the lack of coordination between the different layers of security that individually cover the layers of the IT stack. Today, hardware, operating system, middle-ware, virtualization layer, and the many layers and components that constitute a user-facing application, including data and persistency, and the communication over networks, are still developed largely independently, with little inherent integration. Data at rest, data in motion, communication access and governance, as well as system evolution (e.g., through updates) are managed individually, too often under the responsibility of different actors, different technologies, different products and vendors. In this context, concerns of holistic security are growing, and call for a better end to end integration and communication across the different layers.

This paper illustrated how the partnership of different actors (an SME vendor in security with leading edge research institutions in hardware and software) has managed to create the SEcube™, an open source platform that integrates all these layers. The research and technology contributions concern the model driven and flexible integration and customization of security capabilities, features, and assets, rooted in the hardware device, and used and preserved in the software layers.

Other 4 papers of this track provide in-depth descriptions of the security primitives and protocols [10, 11], the design environment [12,14], several case studies in different application domains and technology spaces, ranging from web applications [12] to computer vision and robotics [14], and a first glimpse of how to deal with property verification and enforcement [14] in this overall platform .

Additionally, paper [19] concerns the Italian National Cyber Security Framework, a methodology that aims to offer to the organizations a volunteer approach to cope with awareness, management, and reduction of the cyber security risk. The Framework approach is deeply tied to the risk analysis rather than to technical standards. It is a generalization of the US NIST Framework for Improving Critical Infrastructure Cybersecurity and it has been realized in alignment with the NIST guidelines.

Acknowledgement

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie).

References

[1] Toppan Ltd, “Side-channel Attack Standard Evaluation Board: SASEBO”, http://www.toptdc.com/en/product/sasebo/, [Online; accessed 19-May-2016].

[2] C. O’Flynn, and D. C. Zhizhang. "ChipWhisperer: An open-source platform for hardware embedded security research." In: *Constructive Side-Channel Analysis and Secure Design*. Springer International Publishing, 2014. 243-260.

[3] Arm LTD, “Juno ARM Development Platform”, http://www.arm.com/products/tools/development-boards/versatile-express/juno-arm-development-platform.php, [Online; acc. 19 5.2016].

[4] Inverse Path Srl, “USB Armory”, https://inversepath.com/usbarmory.html, [Online; ac. 19-May-2016].

[5] L. H. Crockett, R. A. Elliot, M. A, Enderwitz, and R. W. Stewart. The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc. Strathclyde Ac. Media, 2014.

[6] Altera Corportation, “Excalibur Devices”, https://www.altera.com/products/general/devices/arm/arm-index.html, [Online; accessed 19-May-2016].

[7] ST Microelectronics, "STM32F4 Series Data Sheet - DocID022152 Rev 7", March 2016

[8] Lattice Semiconductor, “MachXO2™ Family Data Sheet – DS1035 – v3.2”, May 2016

[9] Infineon, "Infineon Chip Card & Security ICs Portfolio", October 2015

[10] A. Varriale, P. Prinetto, A. Carelli, and P. Trotta, “SEcube™: Data at rest & data in motion protection,” Proc. Int. Conf. on Security and Management (SAM), part of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[11] G. Di Natale, A. Carelli, P. Trotta, and T. Margaria, “Model driven design of crypto primitives and processes,” Proc. Int. Conf. on Security and Management (SAM), part of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[12] S. Boßelmann, J. Neubauer, S. Naujokat, and B. Steffen, “Model driven design of secure high assurance systems: an introduction to the open platform from the user perspective,” Proc. Int. Conf. on Security and Management (SAM), part of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[13] A. Varriale, E. I. Vatajelu, G. Di Natale, P. Prinetto, P. Trotta, and T. Margaria, “SEcube™: An Open-Source Security Platform in a Single SoC,” Design & Technology of Integrated Systems in Nanoscale Era (DTIS), 2016 11th IEEE Int Conf. on, April 2016.

[14] G. Airò Farulla, M. Indaco, A. Legay, and T. Margaria, “Model driven design of secure properties for vision-based applications: A case study,”

Proc. Int. Conf. on Security and Management (SAM), part of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press

[15] B. Steffen, T. Margaria, R. Nagel, S. Jörges, and C. Kubczak, "Model-driven development with the jABC," In: *HVC 2006*, Haifa, Israel. Vol. 4383. LNCS. Springer, 2007, pp. 92–108.

[16] J. Neubauer, M. Frohme, B. Steffen, and T. Margaria, "Prototype-Driven Development of Web Applications with DyWA," In: *Proc. of 6th ISoLA*. LNCS 8802. Springer, 2014, pp. 56–72.

[17] T. Margaria, and B. Steffen. "Business Process Modelling in the jABC: The One-Thing-Approach," In: *Handbook of Research on Business Process Modeling*. IGI Global, 2009.

[18] T. Margaria, B. Steffen. "Service- Orientation: Conquering Complexity with XMDD". In: *Conquering Complexity*. Springer, 2012, pp.217–236.

[19] R. Baldoni, and L. Montanari, "End2End CyberSecurity, based on a strong Public-Private Partnership," Proc. of the Int. Conf. on Security and Management (SAM), part of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.