Securing bitstream integrity, confidentiality and authenticity in reconfigurable mobile heterogeneous systems

Securing bitstream integrity, confidentiality and authenticity in reconfigurable mobile heterogeneous systems

Alberto Carelli, Carlo Alberto Cristofanini, Alessandro Vallero, Cataldo Basile, Paolo Prinetto and Stefano Di Carlo
Politecnico di Torino, Control and Computing Engineering Department, Contact: stefano.dicarlo@polito.it

***Abstract*—The mobile application market is rapidly growing and changing, offering always brand new software to install in increasingly powerful devices. Mobile devices become pervasive and more heterogeneous, embedding latest technologies such as multicore architectures, special-purpose circuits and reconfigurable logic. In a future mobile market scenario where reconfigurable systems are employed, new security concerns are introduced. In particular, protecting the Intellectual Property of the exchanged soft cores is a serious concern and their integrity, confidentiality and authenticity must be preserved. In this paper we propose an architecture suitable for the secure deployment of soft cores in FPGA based mobile heterogeneous systems where multiple independent actors are involved. Finally, we provide a prototype implementation of the proposed architecture.**

I. Introduction

In the rapidly evolving mobile application market, new technologies, players, devices, and platforms emerge just about every month [1]. This constant shift in the business landscape and competitive environment creates a demand but also an opportunity for the rapid introduction of new technological solutions. Few can ignore this opportunity, since mobility will eventually have a role in most digital products and services.

Examples of this exciting world are well known. Apple with the iPhone and its Apple App Store has been the first shaking this market. Over time, its market has been joined by several competitors with their own devices, operating systems and applications stores such as Google Android Market, Nokia Ovi Store, BlackBerry App World, etc. As a consequence, also the sales of mobile apps are continuously increasing.

Among the motivations of this pervasiveness there is the availability of increasing computational power of the mobile platforms. Advancements from generation to generation include many-cores architectures or specialized electronic circuits such as Graphics Processing Units (GPUs). However, system designers are continuously looking for alternative solutions in order to optimize the overall power consumption. In this context, reconfigurable computing may represent a potential answer.

Reconfigurable systems embedding field programmable gate arrays (FPGAs) are emerging as an interesting and useful class of systems in several embedded application domains requiring frequent and remote hardware upgrade [2], [3]. Moreover, the introduction of FPGA dynamic partial reconfiguration, i.e., the possibility to reconfigure at run-time only specific blocks of logic without affecting the remaining parts, unfolds new applicative scenarios [4]–[6].

The mobile application market is looking interestingly at this technology too [7]–[9]. Dynamic partial reconfiguration enables to conceive scenarios in which existing logic can be reconfigured at run-time with application-specific soft cores to assist the software execution. This introduces a new mobile application paradigm that exploits hardware on-demand to minimize computational resources with positive effects on the overall system complexity. Several mobile applications can benefit from hardware acceleration cores deployed on reconfigurable hardware.

However, this new design scenario introduces serious security threats. The use of reconfigurable computing requires moving and storing the hardware Intellectual Property (IP) of the soft cores (i.e., an FPGA biststream file) over possibly insecure channels and repositories. An adversary able to intercept the bitstream may violate the confidentiality of a soft core, revealing it to the public domain or selling it in order to gain improper profit. Moreover, the attacker could also tamper with the hardware description trying to inject malicious functionalities in the hardware core to prevent its proper behavior or to introduce security threats in the system. FPGA vendors already provide bitstream encryption features to help embedded system designers to protect the confidentiality of their know-how and their soft cores [10]–[13]. Several publications propose improved mechanisms to provide bitstream authentication and confidentiality [14], [15]. However, available solutions fit a scenario in which the embedded system designer is the only entity able to produce and deliver reconfigurable hardware descriptions to a remote system. In this paper we address a much more complex scenario in which several independent parties are involved. The parties involved here are mainly the End User, the Software Providers and the Hardware Vendor. In this case several security concerns arise: the confidentiality of soft cores must be maintained across the various actors involved during the exchange of the bitstream. Moreover, the integrity of the bitstream must be guaranteed to avoid malicious alterations, while maintaining the possibility to produce software and reconfigurable hardware descriptions from multiple Software Providers, not directly related with the Hardware Vendor. To addres this problem we propose a secure

bitstream exchange protocol and its implementation using a commercial secure reconfigurable System-On-Chip.

The paper is organized as follows: Section II details the model of the considered scenario. Section III presents the related works. Section IV describes the proposed solution, while Section VI analyzes its security. Finally, Section VII concludes this paper.

II. Assumptions, models and requirements

We model our scenario considering three major actors:

The *Software Provider* (SP) develops a software application to be sold on one or more *Application Stores*. The store is able to reach the customers, i.e., the end users, through a publicly accessible service (e.g., a web server). The application contains both the software executable code and the soft core(s) to load onto the reconfigurable logic.

The *Hardware Vendor* (HWV) is the entity who designs and sells the *Hardware Accelerator Platform* (HAP), i.e., the physical device on which the application sold by the SP is executed. The HAPs here considered are Systems-on-Chips (SoCs) embedding state-of-the-art FPGAs.

The *End User* (EU) is the owner of the device (e.g., smartphone, PDA, etc.) which is equipped with the HAP. After a legitimate purchase from the SP, he is entitled to download and use the application on its hardware.

A. Security requirements

The target for the SP is to preserve the authenticity, the integrity and the intellectual property of all soft cores deployed together with the respective software executable code of an application. The goal of our paper is to consider the authenticity and integrity solely of the hardware cores.

Preserving the authenticity and the intellectual property of all hardware cores deployed with a software application requires fulfilling the following security requirements:

- *Bitstream confidentiality*: among the actors, only the SP is able to read the bitstream in plaintext. However, the SP might trust the HWVs because of legal contracts or Non Disclosure Agreements (NDAs).
- *Bitstream authentication*: only users who bought a legitimate copy of the software must be able to use the related FPGA bitstream files to configure the HAP.
- *Bitstream integrity*: the bitstream must preserve its integrity, in order to avoid that bitstream files delivered to the end users are corrupted, intentionally or not.

If any of these requirements is not met, the authenticity and the integrity of the bistream can be compromised.

B. Attack and adversary models

The attacks here considered try to subvert the security requirements expressed in Section II-A.

First, attacks may aim at accessing a version of the bitstream in plaintext. Proper defense mechanisms must be employed to avoid the disclosure of the bitstream, guaranteeing the confidentiality. These attacks might take place during the exchange over the network links used to deploy the application or by tampering directly with the end user device or directly with the HAP. Additionally, attacks might target to subvert the content of a bitstream. In this case its integrity is compromised. Legitimate users might notice an incorrect execution or the execution of malicious functions not foreseen with the original application due to the violation of the bitstream integrity.

In order to implement the attacks just described, two main adversaries shall be considered.

A *remote adversary* attacking one or more network links existing among the actors during the deployment of the application. He can perform a Man-In-The-Middle (MITM) attack. A *local adversary*, i.e., malicious user who can physically access the end user device (e.g., a competing firm might want to break the protections of an IP bitstream). In this case, a Man-At-The-End (MATE) attack can be carried out.

III. Related works

Previous related works mainly addressed the problem of secure remote update of FPGAs employed in embedded systems. Wallinger et al. proposes an interesting definition of this problem, together with a review of the state-of-the-art of feasible attacks against FPGAs during the configuration and upgrade phases [16].

Bitstream confidentiality is nowadays offered by most FPGA vendors through bitstream encryption facilities [10]–[13], [17], [18]. The bitstream is encrypted with a symmetric key shared between the FPGA and the system designer. The key setup is demanded to the system designer that usually stores the key in a dedicated volatile memory before shipping the system. The memory storing the key is designed to prevent physical attacks and is backed-up by a battery to maintain its content. Bitstream encryption is an effective solution to protect designer's IP against cloning or reverse engineering and IP disclosure [19].

Bitstream integrity is usually accomplished by vendors by means of Cyclic Redundancy Checks (CRC) [20], [21]. However, CRCs are not designed to detect malicious modifications of a bitstream in a cryptographic sense [22]. They are not collision resistant and therefore, even when coupled with encryption, they do not guarantee adequate security levels. In [23] the bitstream checksum for code integrity is performed remotely on FPGAs. Solutions based on cryptographic hashing primitives have been therefore proposed [24]–[26].

In [25] different authenticated encryption algorithms have been evaluated and the dual-pass Counter with CBC-MAC (CCM) has been identified as the best choice for implementing a bitstream authentication mechanism. Nevertheless dual-pass authenticated encryption algorithms do not befit bitstream processing since they lead to configuration process time increasing. Dual-pass authenticated encryption algorithms separate authentication and encryption procedures and therefore require significant overhead.

In [26] the author presents an approach based on generic composition, which involves two symmetric-key encryption cores running in parallel to provide both authentication and

confidentiality. An interesting solution combining both bitstream confidentiality and authentication is proposed in [14]. One of the main contributions of this paper is to address downgrading attacks.

The main drawback of available solutions is that they rely on a single secret shared between the system designer and the target device. The system designer is therefore the only entity in charge of providing updated bitstreams. This mechanism, if not properly modified is unable to satisfy the security requirements introduced in this paper (see Section II). In the proposed scenario, several software providers must be able to generate dedicated bitstreams for a single device. The confidentiality and integrity of these bitstreams must be preserved. To the best of our knowledge, this problem has not been properly addressed yet, thus motivating the study proposed in this paper.

IV. Protocols and secure information exchange

Fig. 1 shows the architecture of the system for the deployment and execution of a mobile application supported by reconfigurable logic.

Four entities are involved in this architecture:

1) the *software provider* (SP) producing the software application (SW) and the related bitstream (BS);
2) the *hardware vendor* (HWV) producing the HAP used to instantiate the soft cores described by the BS and required to accelerate the SW. It is integrated in the end user platform;
3) the *store* providing the infrastructure to sell and deploy applications developed by software providers;
4) the *end user device* representing the system owned by the end user and executing the SW. It communicates with the HAP. The HAP is considered a trusted area, where the bitstream is safe also when in plaintext.

In our scenario, we assume that:

- the end user has an account on the STORE, linked with a payment system recognized and accepted by the STORE itself. Thus, he is able to purchase applications available in the STORE.
- The HAP is identified by a unique code (e.g., serial number - defined here as id_{HAP}) and store a secret cryptographic key (K_{HAP}), both known by the HWV. The key is not accessible from the outside.
- All the involved entities are able to establish secure communication channels among themselves.

The steps to realize the workflow presented in Fig. 1 are described below:

1) the end user contacts the STORE to buy the application (which comprises both SW and BS) developed by the SP, to be executed on the end user device. The STORE redirects the user to the payment system, which performs the monetary transaction. When the transaction is successfull, the STORE can initiate the procedure to obtain the BS requested by its client. To serve this request, the client must send the information needed to identify its HAP, i.e., id_{HAP}. Since the communication involves sensitive data (the credit card number, the id_{HAP}), the client and the store communicate using a secure channel that ensures confidentiality, data integrity and authentication.
2) After the transaction, the STORE notifies the purchase to the SP sending the information about the HAP (id_{HAP}) of the client that will receive the application. Also in this case, the communication is secured.
3) The SP fetches the BS related to the application bought and send it together with the id_{HAP} to HWV within a secure channel.
4) After receiving the BS, the HWV encrypts it using the cryptographic key K_{HAP} related to the id_{HAP}. To ensure data integrity and authentication a keyed-hash message authentication code (HMAC) is used to exchange the BS. The signed and encrypted BS is sent back to the SP.
5) The SP sends back to the STORE the ciphered bitstream using the available communication channel;
6) The STORE routes the ciphered bitstream and the software executable code to the end user device, where the application will be installed. At every execution, the HAP will load the BS on the reconfigurable logic relying on the embedded controller for the decryption using the key K_{HAP} and for the security checks (e.g., autheticity).

A similar workflow can be employed for an update of an already-installed application. In that case, when the SP releases a new version of the BS, the proposed infrastructure can be used as well:

1) the existing SW contacts the SP looking for updated versions;
2) if a new update is available, the client sends its id_{HAP} to SP through a secure channel able to provide confidentiality, data integrity and authentication;
3) the SP connects to the HWV via a a secure channel (that ensures confidentiality and data integrity and authentication) and sends the updated version of BS with the id_{HAP};
4) HWV ciphers BS using the know cryptographic key available and signs it to guarantee data integrity and authentication. The BS is sent back to the previous nodes;
5) the client downloads and stores the ciphered bitstream BS. At every execution, its HAP Controller decrypts it and updates the soft core on the reconfigurable logic.

V. Hardware architecture and implementation results

The End User Device is a normal laptop/desktop PC which is connected to the Storage Medium (microSD card) accessible also from the HAP. For the prototype of the HAP equipped with a reconfigurable architecture, we employed a particular chip, SEcube™. SEcube™ is a system-on-chip, embedding three different devices interconnected within a single package: a microprocessor, an FPGA and a SmartCard. Although not

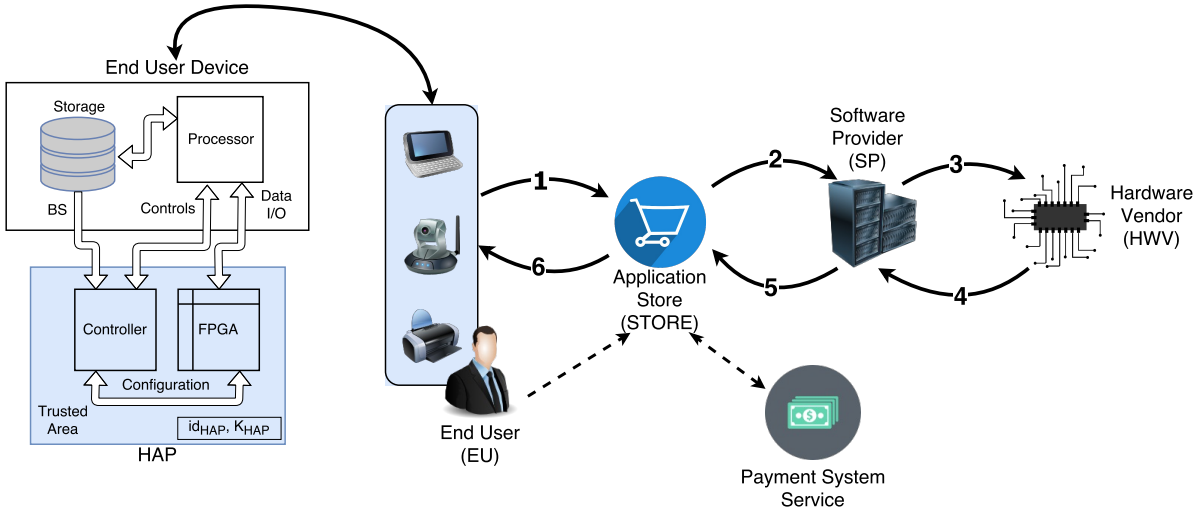

Fig. 1. The workflow for bitsream exchange

very large, the reconfigurable logic device available is a Lattice MachXO2-7000 low-power FPGA. It is directly connected through a 16-bit wide bus with a 32-bit ARM Cortex M4 low-power processor [27]. The microprocessor has a SD/SDIO interface able to communicate with a microSD card, which is used as a storage medium to temporarily store the encrypted bitstream. According to Fig. 1, for the HAP we employed SEcube™ chip, in particular the Controller is the ARM Cortex M4 microprocessor and the reconfigurable logic is the MachXO2-7000 FPGA.

The End User Device executes the market application, which is the client application. The server application of the STORE, SP and HWV reside in the same PC, as separate asynchronous processes. More in detail, the implementation of these software has been carried out using Python v3.5.2. The module asyncio has been employed to offer the client/server and asynchronous communication functionalities. The communication among these processes passes through different port of the same IP address. The messages exchanged are encrypted and signed using the functions of the library PyCryptodome.

VI. Security Analysis

From Section II-B, we see two types of attackers to counteract: man-in-the-middle (MITM) and man-at-the-end (MATE) attackers.

MITM is a technique where an active attacker is able to intercept and understand the content of the messages exchanged between two arbitrary networking nodes. Once the attacker modifies the flow of messages, he can also make changes, delete and create completely new fake messages impersonating one of the communicating parties.

This is a standard security problem in computer networks and there are provably secure solutions to protect against these attacks: the channel protection techniques. In fact, MITM attacks can be neutralized using strong peer authentication mechanisms to avoid impersonation, symmetric data authentication and integrity to avoid the message forging and alteration, and symmetric data encryption to ensure confidentiality of exchanged data.

These techniques use the agreed key with symmetric encryption algorithms (e.g., AES, RC2) to ensure the data confidentiality, and symmetric data integrity and authentication algorithms (i.e., a keyed digest or HMAC using a cryptographic hash function such as SHA-1). It is worth remarking that the proofs of the security of these solutions hold under the *infeasibility* hypothesis. The infeasibility is associated to the derived computational cost, impossible to sustain for an attacker, in order to dechiper the secret information needed by cryptographic algorithms or to invert digest algorithms. That is, we assume that state-of-the-art cryptographic algorithms are used with opportune key lengths. For instance, currently a 128-bit security is required from encryption algorithms (e.g., AES128), that is, the best attack should be a brute force attack on 2^{128} key space, and at least 80-bit security for the digest algorithms (e.g., SHA-256).

When we state that two peers communicate using a k-secure channel, we indicate that the peers perform strong authentication and agree on a symmetric key k that is then used to use symmetric data integrity and authentication algorithms and symmetric encryption algorithms to secure the exchanged data.

The implicit assumption in a MITM scenario is that both the endpoints are trusted entities. However, this assumption is no longer valid in the case where also MATE attackers are interested in obtaining the bitsream. Therefore, MATE attacks are more difficult to address. The MATE attacker has no restriction on the tools and techniques to use to reverse-engineer and then to tamper with the software (e.g., debuggers, emulators) that cannot be trusted to store/embed secret data or routines. System libraries and general purpose libraries could be potentially controlled by the MATE attacker, along with the operating system. In this case, the attacker can use

system calls, the input/output subsystem, the network stack, the memory management subsystem and possibly others for its purposes. Therefore, the communication with the HAP mediated by software and drivers can be compromised by the attacker, thus the data exchanged can be altered. The attacker controls also the hardware of the platform. Every memory location can be read and written, including the processor registers. The attacker also controls the program storage medium, as a consequence he can read and change any of the stored bits. This means that nothing can be considered secure in the user's environment. The only part of the user's platform that can be considered secure is the HAP, the stored information and the routines executed within the HAP are considered confidential. Finally, we assume that the MATE has no interest to perform DoS on its platform and especially on the HAP (e.g., by repeatedly sending invalid bitstreams).

A. Secure BS exchange protocol

The first step of the protocol for bitstream IP protection specified in Section IV represents a typical e-commerce scenario very widespread nowadays where a user is assumed to have a credit card, a SSL/TLS-enabled browser installed in his environment, and an account on the application store of the platform. The store relies on payment services that adhere to the Payment Card Industry Data Security Standard (PCI DSS) [28] to connect to the financial world. The PCI DSS imposes high security requirements for merchants and payment servers that store, process or transmit payment cardholder data when implementing a robust payment card data security process.

When two communicating peers A and B share a symmetric key k:

- only A and B are able decrypt messages encrypted with k (ensuring confidentiality);
- if A receives a message and it can decrypt it using k, A deducts that the message was encrypted by B (symmetric authentication), analogously for B.

In this scenario, the HAP serial number is readable only by the end user, the STORE, the SP, and the HWV. In fact, the HAP identifier is encrypted with the key shared between the client browser and the STORE, then the STORE encrypts it with the key shared with the SP. Finally, id_{HAP} is again encrypted with the key shared with HWV and sent to the Hardware Vendor. All the MITM attackers cannot read the HAP serial number if strong encryption algorithms are used. Additionally, impersonation attacks are impossible as the peers perform symmetric authentication during the connection establishment.

Moreover, to avoid modifications to exchanged messages that can lead to DoS attacks, data authentication and integrity mechanisms are used like keyed digests or HMAC [29].

The actual implementation of these secure communication channels does not require the development of ad hoc techniques. There are valid implementations of general purpose channel protection mechanisms, the most widespread being the SSL/TLS protocol [30] working at the transport layer of the ISO/OSI stack, the IPsec protocol [31] working at network layer, and other application layer methods, usually message protection techniques (e.g., WS-Security). In our case the SSL/TLS approach is the best one, because it does not require additional software or any previous knowledge of the other communicating party, it is integrated with the browsers, etc. Practically, it is "the solution" for web-based scenarios or scenarios where the peers do not have previous agreements.

More important, the BS is read in clear only by the SP and the HWV. In fact, in the step 4, the bitstream is encrypted with a key that is shared between the SP and the HWV. Then, the HWV encrypts the BS using the secret key K_{FPGA} shared with the HAP of the End User Device, thus no one but the HAP, whose ID was provided in the step 4, is able to load and use it. Therefore, MITM and MATE attacks that aim at reading the plain bitstream are avoided.

For the last step, the bitstream received is encrypted until it is moved to the HAP and loaded on the FPGA, while the software application is usually downloaded in cleartext, since other software protection techniques are displaced to protect the intellectual property.

These considerations prove that only the end users that bought the software are able to use the corresponding BS. Moreover, also the users are not able to see the plaintext of the bitstream.

B. How hard is to recover the K_{HAP}?

Attacks can be focused among the transmission nodes. However, the secure encrypted channel guarantees that the bistream exchanged preserve its confidentiality.

Physical attacks represent another type of possible attacks targeting directly the end user device. In this case the attacker aiming to find the symmetric encryption key must be necessarily a MATE owning the device of interest.

In the case of non-invasive physical attacks, both active or passive, the destruction of the device is not necessary, however they generally require a longer time to be accomplished. Brute-force attacks have already been considered under the infeasibility hypothesis. Side-channel attacks can still be employed, however they are not always possible and require specialized equipment. The SEcube™ chip adopted for the prototype is secure against the differential power analysis attack as shown in [32]. Timing attacks can be neutralized simply resorting to a constant-time implementation of the bitstream decryption. Invasive physical attacks destroy the device and require sophisticated and expensive equipment, knowledgeable attackers and long time to be carried out.

In our scenario, a physical attack might target the End User Device, the HAP or the storage medium. The MATE attacker should first bypass any external protection to reach the internal circuitry. Any attack to the End User Device or the storage medium is neutralized with the encryption algorithm, since these parts of the end user platform are considered not trusted. Accessing these peripheral will give to the attacker the possibility to find the ciphertext of bitstream. But the data must still be deciphered, by reversing the key. Possible criticalities

are on the HAP, since the bitstream is deciphered within this device and the key is stored here.

Supposing a successful attack, only a single device is compromised. Every device stores a unique serial number and cryptographic key. In this way, an attacker wanting to inject vulnerabilities or malicious hardware must repeat the attack on other HAPs. This means that also other devices shoud be acquired, leading to higher cost to realize the attack. Also, the effort to exploit the attack is linear.

However, if one bitstream is compromised its description does not remain confidential, but could be disclosed to the public. Moreover, this security breach gives the possibility to the attacker to recover the cryptographic key and the unique serial number for that specific device. Knowing this information any other potential bitstream bought for that specific device can be publicly disclosed.

VII. Conclusion

This work addresses the protection of hardware IP cores to be exchanged in the context of mobile heterogeneous systems.

We provide an architecture for a secure transfer of a IP core bitstream from the developer to the end user device equipped with reconfigurable logic. In the common scenario of an evolving mobile application market, only users purchasing a legitimate copy of an application must be able to use it. In this perspective, the confidentiality of the intellectual property must be maintained. Finally, the integrity of the data exchange is preserved to guarantee that no alteration has been performed during the transfer. Compliance to these requirements protects from MITM attackers. We considered the threat of MATE attackers as well. Finally we also provided a prototype implementation of the whole architecture, employing a system-on-chip as heterogeneous system.

The architecture here provided, however, requires agreements between the Software Provider and Hardware Vendor that must be able to access the bitstream in plaintex. This assumes a trust relationship between the two parties that has to be regulated. On-going work is underway to enhance the proposed protocol avoiding this requirement.

References

[1] P. Albright. Become a Mobile Apps Innovator: Picking an OS and learning to monetize are key. [Online]. Available: http://www.computer.org/portal/web/buildyourcareer/HS26

[2] S. Ke-fei, "Application of FPGA in Aerospace Remote Sensing Systems," *OME Information*, 2010.

[3] M. Surratt, H. Loomis, A. Ross, and R. Duren, "Challenges of remote FPGA configuration for space applications," in *Aerospace Conference, 2005 IEEE*. Ieee, 2005, pp. 1–9.

[4] A. Ahmad, B. Krill, A. Amira, and H. Rabah, "3d haar wavelet transform with dynamic partial reconfiguration for 3d medical image compression," in *Proc. IEEE Biomedical Circuits and Systems Conf. BioCAS 2009*, 2009, pp. 137–140.

[5] A. Tumeo, M. Monchiero, G. Palermo, F. Ferrandi, and D. Sciuto, "An internal partial dynamic reconfiguration implementation of the jpeg encoder for low-cost fpgasb," in *Proc. IEEE Computer Society Annual Symp. VLSI ISVLSI '07*, 2007, pp. 449–450.

[6] M. E. Dunham, Z. Baker, M. Stettler, M. Pigue, P. Graham, E. N. Schmierer, and J. Power, "High efficiency space-based software radio architectures: A minimum size, weight, and power teraops processor," in *Proc. Int. Conf. Reconfigurable Computing and FPGAs ReConFig '09*, 2009, pp. 326–331.

[7] X. Li, Q. Yuan, W. Wu, X. Peng, and L. Hou, "Implementation of GSM SMS remote control system based on FPGA," in *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, dec. 2010, pp. 2132–2135.

[8] H. Nakajo, K. Koike, A. Ohta, K. Ohshima, K. i. o. F.-b. R. p.-k. a. Fujinami, and i. a. i. m.-p. S. encryption system, "Reconfigurable Android with an FPGA Accelerator for the Future Embedded Devices," in *Networking and Computing (ICNC), 2011 Second International Conference on*, 30 2011-dec. 2 2011, pp. 173–178.

[9] N. Qi, J. Pan, and Q. Ding, "The Implementation of FPGA-based RSA Public-key Algorithm and its Application in Mobile-phone SMS Encryption System," in *Instrumentation, Measurement, Computer, Communication and Control, 2011 First International Conference on*, oct. 2011, pp. 700–703.

[10] Actel. Actel proasic3 handbook. http://www.actel.com/documents/PA3_HB.pdf.

[11] Altera. Design security in stratix iii devices. www.altera.com/literature/wp/wp-01010.pdf.

[12] Lattice. Xp2 family handbook. http://www.latticesemi.com/documents/HB1004.pdf.

[13] Xilinx. Lock your designs with the virtex-4 security solution. Xilinx commercial brochure. www.xilinx.com/publications/xcellonline/xcell_52/xc_pdf/xc_v4security52.pdf.

[14] B. Badrignans, D. Champagne, R. Elbaz, C. Gebotys, and L. Torres, "SARFUM: Security Architecture for Remote FPGA Update and Monitoring," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 3, no. 2, pp. 8:1–8:29, May 2010.

[15] S. Drimer, "Authentication of FPGA bitstreams: Why and how," *Reconfigurable Computing: Architectures, Tools and Applications*, pp. 73–84, 2007.

[16] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of-the-art implementations and attacks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534–574, 2004.

[17] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," in *Proc. 18th Int. Parallel and Distributed Processing Symp*, 2004.

[18] A. Lesea, *IP security in FPGAs*, Xilinx Inc., February 2007.

[19] J.-B. Note and É. Rannaud, "From the bitstream to the netlist," in *ACM/SIGDA Symposium on Field Programmable Gate Arrays*. ACM New York, NY, USA, February 2008, pp. 264–264.

[20] Altera Corp. AN357: Error detection using CRC in Altera FPGA devices.

[21] Xilinx Inc. UG191: Virtex-5 configuration user guide.

[22] M. Stigge, H. Platz, W. Muller, and J.-P. Redlich, "Reversing CRC theory and practice," Humboldt University Berlin, Technical Report SAR-PR-2006-05, 2006.

[23] C. Basile, S. Di Carlo, and A. Scionti, "FPGA based remote code integrity verification of programs in distributed embedded systems," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. PART C, APPLICATIONS AND REVIEWS*, vol. 42, no. 2, pp. 187–200, 2012.

[24] M. M. Parelkar and K. Gaj, "Implementation of EAX mode of operation for FPGA bitstream encryption and authentication," in *Proc. IEEE Int Field-Programmable Technology Conf*, 2005, pp. 335–336.

[25] M. M. Parelkar, "Authenticated encryption in hardware," Master's thesis, George Mason University, 2005.

[26] S. Drimer, "Authenticated of FPGA bitstreams: why and how," *In Applied Reconfigurable Computing*, vol. 4419, pp. 73–84, 2007.

[27] *SEcube(tm) Data Sheet*, Blu5 View, 8 2015, rev. 7.

[28] P. Council, "PCI DSS Requirements and Security Assessment Procedures, version 3.2.(2016)," 2016.

[29] H. B. M. Krawczyk and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," Internet Requests for Comments, RFC 2104.

[30] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Requests for Comments, RFC 5246.

[31] R. Atkinson and S. Kent, "Security architecture for the internet protocol," 1998.

[32] M. Bollo, A. Carelli, S. D. Carlo, and P. Prinetto, "Side-channel analysis of SEcube(tm); platform," in *2017 IEEE East-West Design Test Symposium (EWDTS)*, Sept 2017, pp. 1–5.