A NOVEL METHOD FOR SECURE INTELLECTUAL PROPERTY DEPLOYMENT IN EMBEDDED SYSTEMS

Sunil Malipatlolla and Sorin A. Huss

Center for Advanced Security Research Darmstadt
Technische Universität Darmstadt
Darmstadt, Germany
email: sunil.malipatlolla@cased.de, sorin.huss@cased.de

ABSTRACT

The configuration data sequence of a Field Programmable Gate Array (FPGA) is an Intellectual Property (IP) of the original designer. With the increase in deployment of FPGAs in modern embedded systems, the IP protection of FPGA has become a necessary requirement for many IP vendors. There have been already many proposals to overcome this problem using symmetric encryption techniques but these methods need a cryptographic key to be stored in a non-volatile memory located on FPGA or in a battery-backed RAM as done in some of the current FPGAs. The expenses with the proposed methods are, occupation of larger area on FPGA in the former case and limited lifetime of the device in the latter. In contrast, we propose a novel method which combines the Dynamic Partial Reconfiguration (Dynamic PR) feature of an SRAM-based FPGA with the Public Key Cryptography (PKC) to protect the FPGA configuration files without the need of fixed key storage on FPGA or external to FPGA. The proposed method, is secure against the known attacks such as the Man-In-The-Middle (MITM) attack and replay attack. Therefore, the method can be used for secure deploying of IPs from local and remote vendors. Also, using this novel method not only high-end FPGAs but also low-end FPGAs with PR capabilities are secured.

1. INTRODUCTION

Currently SRAM-based FPGAs are becoming increasingly popular as building blocks of electronic systems because of advantages such as easy design modification (reconfigurability), rapid prototyping, economical cost for low volume production, and availability of sophisticated design and debugging tools. Applications of FPGAs in the area of consumer electronics include, for example, television circuits, communication and video processing devices, and software-defined radios.

This work was supported by CASED (www.cased.de)

Since FPGAs are becoming so important for the electronic industry, it is necessary to think about the security of FPGA-based systems. Two possible security measures include are, protecting the FPGA data and the FPGA design itself. In the former case it is necessary to protect the FPGA application i.e., the data inside the circuit and the data transferred to/from the peripheral circuits during the communication. Whereas in the latter, the concerns are against cloning and reverse engineering which is the IP protection problem. Concerning SRAM-based FPGAs it corresponds to the way to protect the bitstream so the FPGA configuration. In essence, the problem of design security is simple, the designer doesn't want that a competitor could be able to pirate his design.

There are two types of piracy: cloning and reverse engineering. Cloning is when a competitor makes a copy of the design, and when he is able to make a copy of the pirated system. With FPGAs it is very simple to clone an unprotected design as the bitstream can be copied to another FPGA's configuration memory. In the case of reverse engineering the design is copied by reconstructing a schematic or netlist level representation. As demonstrated in [1], methods intended to convert FPGA bitstreams into netlists are relatively easy to apply and may soon deliver widely usable results. These two correspond to different attacks, and the design security must protect the system against both these attacks. The papers [2] and [3] give some information about these different attacks.

The two types of attacks are: non-invasive and invasive.

- **The non-invasive attacks** gather all the methods that use external means. For example the attackers can use all the possibilities of the circuit inputs in order to obtain all the different outputs and draw the system truth table, this method is called "black box attack". In the case of an SRAM-based FPGA a simple attack method can be, intercepting the bitstream between the root ROM and the FPGA when the power is switched on. More complex attacks can use power and electromagnetic changes and measures like the simple or differential power analysis [4].

- **The invasive attacks** (- or **physical attacks**) are characterized by the necessity to destroy the integrated circuit (component package) to study the chip (design inside the component) using some complex methods. For example, it is possible to use a laser cutter microscope in order to split the chip in several slices and understand the chip layout [5].

In this paper we consider the protection of FPGA configuration files against the non-invasive attacks only. Especially, the proposed method protects the bitstream against interception by an attacker on the communication channel (Internet/Non-Internet based) between IP vendor (local or remote) and system developer (local or remote). One application scenario for the proposed method can be a secure IP deployment in an embedded system design in the automotive electronics. The rest of this paper is organized as follows. Section 2 gives an overview about the related work done to address the problem of FPGA configuration file (bitstream or an IP) protection. Section 3 briefly explains the PKC and the Dynamic PR feature supported by the state of the art FPGAs. Section 4 compares the conventional scheme with our own proposed novel scheme for bitstream protection by considering some possible attack scenarios on the proposed scheme and giving some solutions against the attacks. In section 5, an analysis of the feasibility in implementation of the proposed method is given while section 6 concludes the paper and gives an outlook into future work.

2. RELATED WORK

There are generally two approaches possible to address the problem of FPGA IP protection. The first solution to protect the device against piracy is the legal solution. The definition of efficient laws, the regulation and the management of intellectual properties are parts of this solution. The second proposal to improve the security level of SRAM-based FPGAs is by bitstream encryption. In this section we mostly address the related work done using the second approach only. For example, Xilinx Virtex series devices support configuration with an encrypted bitstream. Virtex devices have a built-in bitstream decryption unit on them. Virtex-II and Virtex-II Pro support Triple-DES [6] with a 56-bit key, while Virtex-4 and Virtex-5 support AES [7] with a 256-bit key. The secret key for the bitstream decryption is stored in a dedicated volatile memory inside the FPGA which must always be supplied with power through an external battery, thus limiting the lifetime of the device. Additionally, the on-board decryption unit and the corresponding key occupy a considerable amount of space which is very crucial in resource constraint embedded systems.

In order to overcome the problem of an additional battery in Xilinx's solution, Tom Kean of the Algotronix society proposed ideas to store the cryptographic secret key on FPGA, such as using laser to program a set of links during manufacture [8]. However, in his method the encryption and decryption circuits are embedded inside the FPGA which causes less available silicon area for developed applications. Also, the encryption and decryption circuits are fixed, so it is not possible to upgrade them. In contrast, Kun-Wah Yip et al. proposed the IP protection scheme using partial-encryption (PE) technique [9]. Their method argues that the PE technique outperforms the full-encryption technique in terms of the reverse engineering cost. Whereas, Jorge Guajardo et al. proposed a different scheme, using FPGA's intrinsic physical unclonable functions (PUF) and PKC for IP protection. Though their method uses PKC-based authentication protocol which does not need the private key to be stored on the FPGA, they did not make use of the advantages provided by partial reconfiguration. In addition the PUF implementation and its analysis on an FPGA is in itself a challenging task [10].

There are other techniques proposed for FPGA IP protection like watermarking as in John Lach et al., where they apply a watermark to the physical layout of a digital circuit when it is mapped onto an FPGA which uniquely identifies the circuit origin and yet difficult to detect [11]. In contrast, Tim Güneysu et al. used both public key and symmetric key cryptography to dynamically protect the IP of circuits in configuration files. In their method the symmetric cryptography is hard-wired and the public-key functionality is moved into a temporary configuration bitstream for a one-time setup procedure [12]. Also, Bossuet et al. proposed a scheme where an embedded key is accessible to the user logic and uses partial reconfiguration to encrypt and decrypt the bitstream [13]. Here, an on-chip key is used to encrypt the main design's bitstream before storing it in a PROM where a decryption bitstream is also stored. In [14], Simpson et al. proposed an offline authentication scheme for secure delivery of IP modules in non-networked environments of FPGA design. Their scheme implements a mutual authentication of IP modules and the hardware platform, thereby enabling the authentication and integrity assurances to both system developer and IP provider.

As mentioned above, all these methods need a secret key to be stored on an FPGA which in itself is a challenge in SRAM-based FPGAs as the memory on these devices is volatile. In contrast, we may store the keys in a non-volatile memory placed on an FPGA, but this has two drawbacks: One being necessity for an extra space on the FPGA, which is crucial for embedded systems as they are area constrained, and the other being the possible extraction of stored cryptographic keys by an attacker which makes the device less secure. However, to the best of our knowledge, the idea of

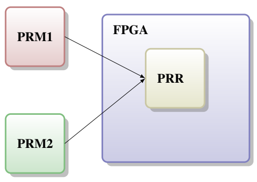

Fig. 1. Partial Reconfiguration in FPGAs

using Dynamic PR and PKC for FPGA bitstream protection has not yet been addressed.

Our method utilizes the special feature of SRAM-based FPGAs, the partial reconfiguration, and the well-known public key cryptography to protect the FPGA bitstreams. The novelty of our method is that it does not need any fixed key storage, either on the FPGA or outside of FPGA, for encryption and decryption of bitstreams as the keys are generated on the fly. There is no threat of the private key being stolen, as it is stored (temporarily) deep inside the memory blocks which are erased when the device is turned off. Also, as the keys for encryption and decryption of bitstreams are generated on the FPGA, unlike the single symmetric key in previously referenced papers, to be sent to the host for bitstream (IP) encryption, it is possible to address the problem of loading IPs from different vendors. So, different vendors can use on the fly generated keys to encrypt their IPs, before sending them to the FPGA for secure deployment in the field, where they will all be placed on a single System-on-Chip (SoC). In the following a brief introduction to Dynamic Partial Reconfiguration (Dynamic PR) and Public Key Cryptography (PKC), the techniques used in the proposed scheme, is given before delving into more details about the actual method for bitstream protection.

3. DYNAMIC PR AND PKC OVERVIEW

3.1. Partial Reconfiguration Overview

Some of the SRAM-based FPGAs support a special feature called Partial Reconfiguration (PR) in which a portion of the FPGA's fabric is reconfigured while the rest resumes its work. The portion being reconfigured is the dynamic part and the portion resuming the work is the static part of the FPGA. If the configuration of the FPGA is changed at run-time i.e., the system is neither stopped nor switched off, then its called as Dynamic PR. Additionally, if the system triggers the reconfiguration by itself then it is a self-reconfigurable system which does not require the use of internal FPGA infrastructure. The area of the FPGA that is reconfigured is called the Partially Reconfigurable Region (PRR). A PRR typically consists of a number of Configurable Logic Blocks (CLBs) and functional blocks. The module to be placed inside the PRR is called a Partially Reconfigurable Module (PRM), which is the specific configuration of the PRR and at least two PRMs are needed per PRR. In many cases the assignment of PRMs to PRR is fixed (non-relocatable) though in principle, a PRM may be configured to different PRRs.

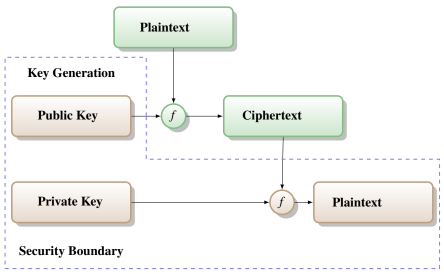

Fig. 2. Asymmetric Key Cryptography

In figure 1, we see that two PRMs which are mutually exclusive in time will be placed in the PRR inside the FPGA i.e., only one PRM can be assigned to a given PRR at a given time. The remaining region in the FPGA which is outside the PRR is the static region, where the application which needs to be run uninterruptedly, is placed. The configuration files placed in the PRR are called as partial bitfiles. The partial reconfiguration of an FPGA is done through the Internal Configuration Access Port (ICAP), a built-in hard core IP module available on the FPGA. The ICAP module is controlled by the software driver for the processor (Xilinx$'s$ Microblaze or IBM$'s$ PowerPC) on the FPGA. In our scenario, the static logic contains the asymmetric algorithm (RSA or ECC), which generates the public-private key pair. The reconfigurable (dynamic) logic is populated with the partial bitstreams through ICAP, which are the actual FPGA applications to be implemented, after an on-board decryption process.

3.2. Public Key Cryptography Overview

Public (or Asymmetric) key cryptography (PKC) uses asymmetric key algorithms like RSA and ECC. Unlike symmetric key algorithms, they do not require a secure initial exchange of key between the sender and the receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key. Messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key, which is known only to the receiver. In figure 2, we see that all of the functions in the "dashed box" (named as security boundary) can be implemented within the physical package of the FPGA. The plaintext and the private key information never leave a well-protected container i.e., the security boundary.

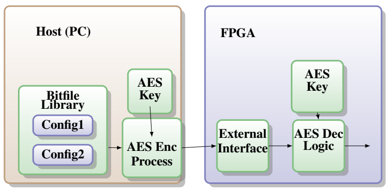

Fig. 3. Conventional Scheme for Bitstream Protection

4. FPGA BITSTREAM PROTECTION

This section compares the conventional scheme with the proposed novel scheme for bitstream protection in FPGAs.

4.1. Conventional Scheme

The conventional scheme uses the bitstream encryption to protect the bitstream from potential threats as shown in figure 3.. The Host (PC) first encrypts the bitstream with a symmetric encryption algorithm such as AES using a secret key before sending it to the FPGA. On the FPGA side the bitstream is decrypted using the same secret key, stored on FPGA, and the corresponding decryption unit. For example, the Xilinx Virtex-5 device consists of a software-based bitstream encryption (on the PC side) and an on-chip bitstream decryption with dedicated memory for storing the symmetric key (on the FPGA side). The drawback with the conventional scheme is the additional key storage space on the FPGA and its possible extraction.

4.2. Proposed Novel Scheme

A very secure method for protecting the FPGA configuration files can be built when Dynamic PR and PKC are combined. By using these two basic methods we propose a novel technique to protect FPGA IP without the need to store any cryptographic keys in a dedicated storage. The architecture of the proposed method is outlined in figure 4 and the corresponding communication protocol is given in figure 5. The considered FPGA supports the dynamic PR, and we divide the FPGA's logic into two parts: static area and dynamic area. The initial bitfile (full bitstream) to be loaded onto the FPGA in the static area is an unencrypted design that does not feature any proprietary information. It only contains the algorithm to generate the public-private key pair and the interface between the Host, FPGA, and ICAP. The communication medium between the Host and the FPGA uses the UART protocol. The partial bitstreams are the configurations that are loaded into the dynamic part of the FPGA and are secured against the attacks with our scheme as shown in figure 5.

Following are the advantages of the proposed scheme:

- The public-private key pair may be regenerated at any

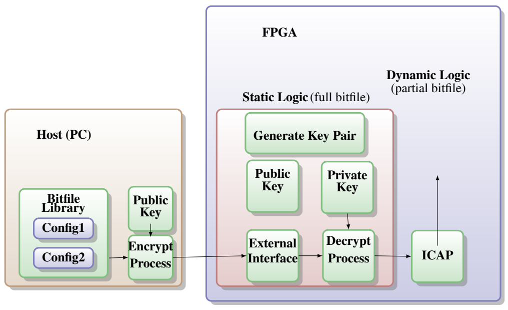

Fig. 4. Proposed Scheme for Bitstream Protection

time. If a new configuration is downloaded from the host it may be encrypted with a different public key and decrypted with the corresponding new private key. Even if the FPGA is configured with the same partial bitfile later, such as after a power-on-reset, a different public/private key pair is used even though it is the same bitfile.

- There is no need of any non-volatile memory to store the key (as for symmetric key in previously mentioned scheme) for decrypting the bitfiles as the private key is generated on the fly. In addition, the private key generated by the asymmetric algorithm running on the FPGA is stored in the SRAM and if the FPGA loses power then the private key no longer exists as the SRAM memory is volatile.

- In general, the partial bitfile contains the vast majority of the FPGA design with the logic in the static design consuming a very small percentage of the overall FPGA resources. So, most of the FPGA resources are allocated to the actual applications contained in the partial bitstreams.

- Even if at some point of time it is found that the asymmetric algorithm being used is no more secure, one can replace it with a new algorithm as it just requires loading a new full bitstream into the static region of the FPGA.

- Even if the system is stolen and the FPGA remains powered it is extremely difficult to find the private key, because it is stored in the general purpose FPGA fabric, but not in a special purpose register.

- The issue of loading IPs from different vendors onto a single SoC is addressed.

- This scheme can be used to protect the IPs of low-end FPGAs too, which support the partial reconfiguration feature.

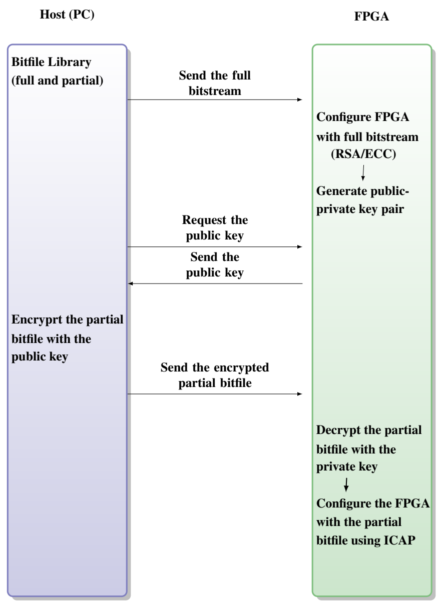
Host (PC)
FPGA
Bitfile Library (full and partial)
Send the full bitstream
Configure FPGA with full bitstream (RSA/ECC)
Generate public-private key pair
Request the public key
Send the public key
Encryprt the partial bitfile with the public key
Send the encrypted partial bitfile
Decrypt the partial bitfile with the private key
Configure the FPGA with the partial bitfile using ICAP

Fig. 5. Protocol for secure FPGA Configuration

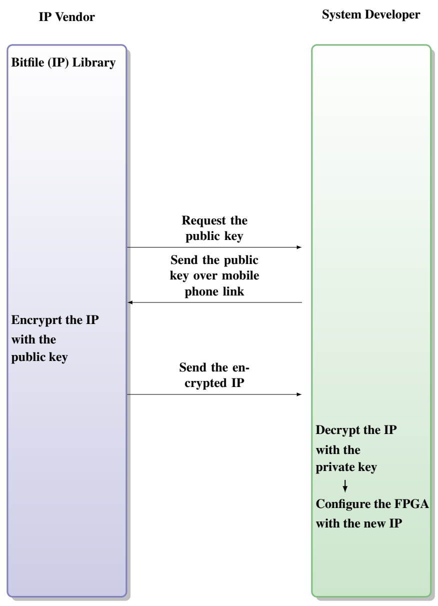
IP Vendor
System Developer
Bitfile (IP) Library
Request the public key
Send the public key over mobile phone link
Encryprt the IP with the public key
Send the en-crypted IP
Decrypt the IP with the private key
Configure the FPGA with the new IP

Fig. 6. Security Protocol against MITM Attack

There are also certain disadvantages with this scheme, like the implementation of asymmetric key algorithm (RSA) on an FPGA consumes a lot of resources which can be avoided by considering efficient asymmetric key algorithms like ECC instead of RSA. Although the generation of partial and full bitstreams for the FPGA is cost expensive, time expensive, and exhausting at the moment, the FPGA vendors claim that it will be much easier on top of their newer versions of tools. In the following we consider a few attack scenarios against the proposed method and define the corresponding solutions.

4.3. Possible Attack Scenario and Solution

The main objective of the proposed scheme is to protect the FPGA configuration files without the fixed storage of keys in/out of FPGA. However, we can use Flash-based FPGAs over SRAM-based FPGAs for key storage but the former are not in common use yet. The proposed system works well when we consider an in-house scenario i.e., the PC and the FPGA are directly connected to each other and the loading of IP is local i.e., from PC to FPGA over a JTAG/USB cable. But when the bitstream (IP) is to be loaded from a remote location or over Internet then there is a possibilty of an attack on the system such as the Man-In-The-Middle (MITM) attack.

In MITM attack, an adversary tries to intercept the communication channel between the IP provider (PC in our case) and the system developer (FPGA onto which the IP is to be deployed). Here the adversary can decieve the system in two possible ways. In one case, he can pose himself as the FPGA system (IP receiver) and send his own generated public key to the PC system (IP vendor) for encryption inorder to decrypt it later, with his private key, to know the application (or IP) to be loaded onto the FPGA, thereby compromising the privacy of the whole system. In the other case, he can pose himself as the IP provider to receive the public key from the FPGA system and send back encrypted malicious bitstreams to the FPGA which on after loading on to the system would destroy the device. These two attacks clearly show that there is a lack of mutual authentication for data origin in the system.

One solution to avoid the MITM attack is by providing an additional, non-Internet based channel for transmission of the public key from FPGA system to the IP vendor. The FPGA armed PC workstation (system developer) must exploit an UMTS stick for a mobile phone link to the configuration files server (IP vendor) as shown in figure 6. Thereby, only the IP vendor and the FPGA (IP receiver) would know the public key being transmitted over the channel, which guarantees the IP origin. However, there are other solutions available in the literature for thwarting MITM attack.

5. IMPLEMENTATION RESULTS

Algorithms ECC, AES, and RSA have been implemented on a Xilinx V5LX110T platform and their resource requirements are compared to show the feasibility of implementation of the proposed scheme. The resource consumptions for the algorithms and other modules needed by the scheme are summarised in table 1. Obviously, the number of resources (slice LUTs and slice registers) occupied by the public key

Table 1. Synthesis Results

Module	LUTs	Registers	BRAMs	Delay
ECC	2466	1207	2	5.142 ns
AES	853	536	5	3.870 ns
RSA	16319	12080	2	7.442 ns
PR ICAP	168	170	2	-
UART	753	442	-	-

algorithm (RSA) is much higher compared to the symmetric key algorithm (AES) but the ECC resource consumption is approximatable with AES. Also, the calculation time delay for each of the algorithms is measured at a speed grade of "-1" of the FPGA device. So, the use of ECC algorithm for decrypting the incoming encrypted partial bitstream instead of an on-board AES decryption unit is justified with reference to the overall advantages gained as mentioned in the previous section.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a novel design method to protect the FPGA configuration files which avoids the need to store the cryptographic keys in registers of the FPGA or in an external non-volatile memory. The proposed scheme uses the special feature of SRAM-based FPGAs, i.e., dynamic partial reconfiguration and the well-known public key cryptography scheme to secure the IP of the design. In addition, we considered how to avoid the MITM attack by proposing a security protocol against it thereby providing a complete solution for secure loading of the IPs supplied from different vendors onto a single SoC. The feasibility of the implementation of the proposed scheme on a Xilinx Virtex-5 FPGA platform was shown with resource consumption values of algorithms used in the scheme. As a part of future work there is a need to reduce the number of resources being consumed by public key algorithms through algorithm optimization and also to strengthen the system against other possible attacks.

7. REFERENCES

[1] J.-B. Note and E. Rannaud, "From the bitstream to the netlist," in *Proceedings of the 16th ACM/SIGDA nternational Symposium on Field Programmable Gate Arrays*, ser. FPGA '08. New York, NY, USA: ACM, 2008, pp. 264–264. [Online]. Available: http://doi.acm.org/10.1145/1344671.1344729

[2] R. Anderson and M. Kuhn, "Tamper resistance: a cautionary note," in *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, 1996, pp. 1–1.

[3] ——, "Low cost attacks on tamper resistant devices." Springer-Verlag, 1997, pp. 125–136.

[4] S. Mangard, "A simple power-analysis (SPA) attack on implementations of the AES key expansion," in *Proceedings of the 5th international conference on Information security and cryptology*. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 343–358.

[5] G. Canivet, R. Leveugle, J. Clediere, F. Valette, and M. Renaudin, "Characterization of Effective Laser Spots during Attacks in the Configuration of a Virtex-II FPGA," in *Proceedings of the VLSI Test Symposium, 2009. VTS'09. 27th IEEE*. IEEE, 2009, pp. 327–332.

[6] in *Xilinx Corporation. Virtex-II platform FPGA Handbook.*

[7] in *Xilinx Corporation. Virtex-5 FPGA configuration guide.*

[8] T. Kean, "Secure configuration of field programmable gate arrays," in *Proceedings of the International Conference on Field Programmable Logic and Applications*. Springer-Verlag, 2001, pp. 142–151.

[9] K. Yip and T. Ng, "Partial-encryption technique for intellectual property protection of FPGA-based products," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 1, pp. 183–190, Feb. 2000.

[10] J. Guajardo, S. S. Kumar, G. Schrijen, and P. Tuyls, "Physical unclonable functions and Public-Key crypto for FPGA IP protection," in *Proceedings of the International Conference on Field Programmable Logic and Applications*, 2007, pp. 189–195.

[11] J. Lach, W. H. Mangione-smith, and M. Potkonjak, "Signature hiding techniques for fpga intellectual property protection," in *Proceedings of the IEEE/ACM International Conference on Computer Aided design*, 1998.

[12] T. Guneysu, B. Moller, and C. Paar, "Dynamic intellectual property protection for reconfigurable devices," in *Proceedings of the International Conference on Field Programmable Technology*, 2007, pp. 169–176.

[13] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004*, Santa Fe, NM, USA, pp. 146–153.

[14] E. Simpson and P. Schaumont, "Offline Hardware/Software authentication for reconfigurable platforms," in *Proccedings of the Cryptographic Hardware and Embedded Systems CHES 2006*, 2006, pp. 311–323.