

Compact and On-the-Fly Secure Dynamic Reconfiguration for Volatile FPGAs

HIRAK KASHYAP and RICARDO CHAVES, INESC-ID, IST, Universidade de Lisboa

The dynamic partial reconfiguration functionality of FPGAs can be attacked, particularly when the FPGA is remotely located or the configuration bitstreams are sent through insecure networks. The existing FPGA technologies provide some built-in security mechanisms; however, these are often inadequate. The existing solutions still impose a significant impact on the reconfiguration process and on the available resources.

This article proposes a solution to improve the security of dynamic partial reconfiguration of FPGAs, without significantly affecting the reconfiguration performance. The proposed solution changes the encryption key of the remotely received bitstream by a randomly generated key, unique for each configuration, when storing them in the external unsecured memory. The native frame-wise error detection mechanism combined with an additional CBC-MAC authentication mechanism, allows for an improved countermeasure against replay attack and wrongful bitstream usage. The proposed solution introduces an overhead of 1% of the available resources on the target FPGA and provides the lowest impact on the reconfiguration process when compared to the state of the art, achieving a reconfiguration throughput of 2.5Gbps. Regarding the built-in security mechanism provided by the Xilinx FPGAs, the solution herein proposed provides better security and improves the reconfiguration performance by more than 3 times.

Categories and Subject Descriptors: B.6.2 [Hardware]: Reconfigurable Logic and FPGAs; K.4.4 [Security and Privacy]: Hardware Security Implementation; K.4.1 [Security and Privacy]: Embedded Systems Security; K.4.1 [Security and Privacy]: Malicious Design Modifications; K.4.4 [Security and Privacy]: Security Services

General Terms: Security, Design, and Performance

Additional Key Words and Phrases: Bitstream encryption, bitstream security, reconfigurable architectures, secure dynamic partial reconfiguration, and system downgrade prevention

ACM Reference Format:

Hirak Kashyap and Ricardo Chaves. 2016. Compact and on-the-fly secure dynamic reconfiguration for volatile FPGAs. *ACM Trans. Reconfigurable Technol. Syst.* 9, 2, Article 11 (January 2016), 22 pages.

DOI: <http://dx.doi.org/10.1145/2816822>

1. INTRODUCTION

Reconfigurability has emerged as a key feature in the dedicated, embedded, and high-performance computing systems. Apart from providing high adaptability to dynamic computing requirements, it also allows to use configurable accelerators and to reduce the overall cost of the system. This feature is very useful in systems with high processing requirements that need to be remotely configured/adapted [Hauck and DeHon 2010]. Reconfigurable systems use highly flexible computing fabric, usually

This work was partially supported by the ARTEMIS Joint Undertaking under grant agreement n° 621429 and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

Authors' addresses: H. Kashyap and R. Chaves, INESC-ID, IST, Universidade de Lisboa, Rua Alves Redol 9, 1000-029 Lisbon, Portugal; emails: kashyap@sips.inesc-id.pt; ricardo.chaves@inesc-id.pt.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2016 ACM 1936-7406/2016/01-ART11 \$15.00

DOI: <http://dx.doi.org/10.1145/2816822>

Field Programmable Gate Arrays (FPGAs). The partial dynamic reconfiguration functionality of FPGAs enables the device to repeatedly change its configuration while operating. The FPGA devices are today used in a wide range of applications, from consumer electronics, such as set-top boxes, to mission-critical systems such as vehicular automation systems or outerspace vehicles and robots. El-Ghazawi et al. [2008] showed that using FPGAs in the nodes of heterogeneous clusters, significant speedup and energy reduction can be achieved for various high-performance computing problems, in comparison to the traditional clusters.

While FPGAs are being increasingly used, security threats against FPGAs and their configuration operations are also increasing. Such threats arise mainly due to the fact that most FPGAs are volatile and do not contain internal Non-Volatile Memory (NVM) to store the configuration data, known as bitstreams. As such, the configuration data is stored unprotected and externally to the FPGA itself. At FPGA power up, the configuration data, externally stored, is sent to the FPGA in order to configure it. Identically, in case of partial reconfiguration, the partial configuration bitstreams are stored in external NVM and later loaded into the FPGA, according to the dynamic requirements.

Wollinger et al. [2004] and Drimer [2009] detail several attacks against FPGAs and their configurations, particularly against SRAM-based FPGAs. In this context, an attacker is an agent who tampers or replays a bitstream in order to break or downgrade the configuration or to perform unintended functions. Passive attacks, such as reverse engineering and cloning of bitstreams, are also possible. In case of dynamic reconfiguration, due to frequent updates of volatile FPGA fabric using externally stored bitstreams, an attacker can easily eavesdrop on the transmission to obtain or tamper with the configuration file. In applications where the hardware constantly reconfigures itself, multiple configuration bitstreams need to be stored in external high-throughput memories. Given this, secure external storage of configuration bitstreams is required, in particular when the reconfigurable systems are deployed in hostile environments.

To cope with this, several solutions have been proposed in the literature toward securing the process of bitstream upload and reconfiguration of FPGAs, considering the bitstream storage on external memories. Maes et al. [2012] and Vliegen et al. [2015] use a trusted third party to secure the bitstream upload process and to provide a pay-per-use licensing policy. Other approaches proposed by Braeken et al. [2011] and Drimer and Kuhn [2009] assume a secure external memory outside the FPGA to store partial configuration bitstreams and other control data. Vliegen et al. [2013] propose a rather secure solution but restrict the bitstreams storage to the FPGA's internal logic. Unfortunately, this solution lacks scalability and imposes a significant area cost on the device, leaving less configurable resources available for applications.

The solution herein proposed allows to improve the security of the dynamic partial reconfiguration process in regard to the existing state of the art [Braeken et al. 2011; Devic et al. 2012; Hori et al. 2013; Vliegen et al. 2014], while imposing a low footprint on the available reconfiguration logic and a minimum impact on the reconfiguration process itself. The reduced impact is particularly useful in systems that require frequent reconfiguration of the hardware. Security-wise, the proposed solution considers confidentiality, integrity, authentication, and freshness of the bitstreams during storage on the unsecured external memory and during the reconfiguration process itself.

These properties are achieved mainly by re-encrypting each received bitstream with a unique randomly generated key before storing it on the external memory and an additional CBC-MAC authentication. Associated with the frame-wise error detection mechanism of the newer FPGA technologies, near immediate partial configuration bitstream replay and tamper attacks can be detected and prevented.

To implement the proposed solution, the architecture considers high-performance AES engines to perform the symmetrical encryption/decryption processes and Message

Authentication Code (MAC) calculation, a True Random Number Generator (TRNG) for the key generation, and the remaining logic for the reconfiguration processes.

The considered security solution occupies 45% less Slice resources regarding the most relevant state of the art by Devic et al. [2012], while achieving a reconfiguration throughput 93 times higher. The solution proposed by Hori et al. [2013] with on-the-fly bitstream validation does not provide replay attack protection and still requires more Slice resources and achieves a reconfiguration performance 51% slower than the solution herein proposed. On the target device (a Xilinx Virtex-7-XC7VX485t FPGA), the proposed solution introduces a resource usage increase of 1.1%, achieving a reconfiguration throughput of 2.5Gbps. Moreover, the proposed solution allows to achieve 3 times faster reconfiguration throughput than the built-in secure reconfiguration mechanism of modern Xilinx FPGAs. The Xilinx FPGA technology is herein considered because it is the one most used in the industry and in the state of the art. This is mostly due to the fact that the other major manufacturers supporting partial dynamic reconfigurable FPGAs, such as Altera, do not provide enough information on their bitstream structure and security mechanisms [Swierczynski et al. 2015].

1.1. Contributions

The following details the main contributions of this article:

- (1) Herein, a novel solution to securely store the partial configuration bitstreams on unsecured external memory and to validate them on the fly during the dynamic reconfiguration of the FPGA is presented. Regarding the state of the art, the proposed solution is more robust and provides improved bitstream freshness, authentication, and the ability to securely store the bitstreams on external memory.
- (2) This work extends the solution originally proposed in Kashyap and Chaves [2014], presenting two improved implementations, considering a single AES core, instead of the original solution with three AES cores. Regarding the state of the art, both of these implementations are the most compact and provide the highest configuration throughput, imposing the least overhead on the reconfiguration process.

1.2. Organization

This article is organized as follows. Section 2 presents an overview of the existing technology and the state of the art regarding the security of partial configuration bitstreams and the dynamic partial reconfiguration process. The proposed architecture for the secure partial dynamic reconfiguration process is presented in Section 3. Section 4 details the implementation of the proposed solution, and Section 5 presents the obtained results and comparison with the related state of the art. Concluding remarks are presented in Section 6.

2. STATE OF THE ART IN SECURE FPGA RECONFIGURATION

Recent FPGAs have the capability to adjust the entire or part of their computing fabric to the desired computation. This is achieved with the support for partial dynamic reconfiguration. Typically in SRAM FPGAs, bitstreams are stored on an external nonvolatile memory or retrieved from a remote source.

Given the physical separation between the FPGA and the bitstreams storage, several malicious attacks may be performed against the FPGA configuration [Wollinger et al. 2004; Trimberger and Moore 2014], such as readback attack, cloning of the FPGA content, reverse engineering of the bitstreams, and replay/downgrade attacks. The following describes the existing security mechanisms in current volatile FPGA technologies and the solutions proposed in the related state of the art to mitigate such attacks.

It is to be noted that the existing nonvolatile FPGAs do not support partial reconfiguration.

2.1. Native Technology Support for Secure Dynamic Reconfiguration

This article considers FPGA devices that allow for partial dynamic reconfiguration. Among the FPGA technologies that allow partial reconfiguration, the Xilinx FPGA devices provide the more elaborate built-in security features for partial configuration bitstreams, such as frame-wise CRC in the recent 7 series FPGA devices, discussed later. Notwithstanding, the state-of-the-art Xilinx security features for partial configuration bitstream are insufficient against many types of attacks against FPGAs. This article considers the problem of developing a more robust security mechanism for partial configuration bitstreams than the existing state of the art.

2.1.1. Bitstream Encryption and Authentication. Modern FPGAs provide bitstream encryption and authentication mechanisms. The encryption of the bitstream is performed in software when generating the bitstream itself, while the decryption is performed within the device by a dedicated hardware circuit. Particularly, recent Xilinx Virtex and Spartan devices include an on-chip AES decryption engine, using an internal key stored either on a dedicated battery-backed RAM (BBRAM) or on an internal ROM memory called eFUSE. The encryption key cannot be reprogrammed without erasing the whole configuration. Since the Virtex-6 devices, the SHA-256-based keyed-Hash Message Authentication Code (HMAC) algorithm is also included in the hardware. The authentication key and the MAC are transmitted as part of the encrypted bitstream. It should be noted that bitstream authentication is only available when bitstream encryption is used [Xilinx 2013].

2.1.2. Bitstream Integrity. Regardless of the authentication, a noncryptographic validation is performed by a Cyclic Redundancy Check (CRC) code added to the bitstream, covering the entire bitstream. However, the verification of the CRC code is only performed at the end of the bitstream upload. If partial reconfiguration is considered and an error exists in the address region of the configuration bitstream, the corruption may modify undesired or restricted regions of the device, irreversibly compromising the overall configuration of the FPGA.

To solve this problem, the newer Xilinx 7 series FPGAs include a frame-by-frame CRC. If the CRC for a configuration frame fails, the configuration engine does not load that frame into the configuration memory [Xilinx 2012]. This ensures that the remainder of the device stays operational while the system is allowed to recover from this error. It also ensures that the partial configuration bitstreams do not overwrite unintended regions within the device.

2.1.3. Vulnerabilities in the Built-in Security Mechanisms. The built-in security features of recent Xilinx FPGAs, such as bitstream encryption and authentication, are inadequate to prevent the main attacks against FPGAs. These attacks are possible mainly due to the following reasons.

- (1) **Single key for all bitstreams:** The built-in security system uses the same encryption key for all bitstreams. Therefore, replay attacks using older or unintended encrypted partial configuration bitstreams are possible. The use of a single encryption key also simplifies the passive attacks such as cryptanalysis. Moreover, both the authentication key and the authentication tag are sent as a part of the encrypted bitstream. As a result, the authentication performed in the hardware is incapable of detecting a replayed or unintended bitstream.
- (2) **The MAC is verified only at the end of the bitstream:** The MAC is calculated from the decrypted bitstream in hardware as the configuration data are sent to

the configuration port. In case of full bitstream, the configuration will not be active when a mismatch is detected (between the calculated and the received MAC). However, when performing partial reconfiguration, the design is already in the active state before the mismatch is detected. Therefore, an attacker can overwrite an unintended or static partition of the active design by manipulating the address field of the bitstream.

- (3) **CRC field can be attacked:** The novel frame-wise CRC feature, available since Virtex-7, ensures that the partial configuration bitstreams never overwrite unintended partitions. However, the CRC value can be deterministically recalculated by a malicious attacker [Geier 2002], even when the bitstream is encrypted using a stream cipher. Nevertheless, when using block cipher encryption, the CRC validation becomes significantly more robust against attacks.

Despite the existing mechanisms on the more recent FPGA devices, the use of a single encryption key and the lack of freshness mechanisms make the encryption key susceptible to overexposure and the bitstream to replay attacks [Drimer 2008]. Replay attacks, also termed as system downgrade attacks [Badrignans et al. 2008], are achieved when an attacker replaces the correct bitstream by an outdated or currently undesired bitstream. Moreover, since the authentication key and the verification MAC are within the encrypted bitstream, the calculated MAC will always match.

2.1.4. Configuration Limitations Imposed by Bitstream Encryption. In Xilinx 7 series FPGAs, an encrypted bitstream can be delivered through several configuration interfaces, namely JTAG, SelectMAP, SPI, BPI, and ICAP. However, encrypted bitstream imposes additional limitations and configuration delays. In case of the SelectMAP, JTAG, SPI, and BPI interfaces, the encrypted bitstream must configure the entire device, since partial reconfiguration through the external configuration interfaces is not permitted for encrypted bitstreams [Xilinx 2013].

In case of the internal ICAP interface, the aforementioned restriction of the external interfaces does not apply. Encrypted or unencrypted partial configuration bitstreams can be sent through the ICAP interface even after a full device configuration using an encrypted bitstream.

However, the ICAP interface only accepts encrypted bitstreams through the 8-bit bus at a frequency of 100MHz. The same interface accepts unencrypted bitstreams through the 32-bit bus. Therefore, the use of the built-in security mechanisms effectively reduces the maximum reconfiguration throughput of the ICAP interface by 4 times, from 3,200Mbps to 800Mbps.

2.1.5. Comparison of FPGAs in Terms of Security Features. Table I presents a comparative study among the different generations of the Xilinx FPGA devices in terms of bitstream security.

The Xilinx Virtex-4 series introduced the use of AES 256-bit cipher, the possibility to perform partial reconfiguration after an initial configuration with an encrypted bitstream (not allowed in the Virtex-5 series), and the use of a 32-bit ICAP bus capable of achieving a configuration throughput of 3.2Gbps, for nonencrypted bitstreams. With the Virtex-6 series, additional security features were introduced, namely the introduction of the eFUSE NVM to store the bitstream decryption key and the bitstream authentication with keyed HMAC (using SHA256), as a complement to the full bitstream CRC. The most recent Virtex-7 series added the frame-wise CRC, allowing the individual verification of each configuration frame. The Spartan-6 series allows for encrypted bitstreams using AES 256 and the use of eFUSE memory, but it does not provide bitstream authentication.

Table I. Built-in Security Mechanisms in the Xilinx FPGA Devices

Technology	Virtex-II	Virtex-4	Virtex-5	Virtex-6	Virtex-7	Spartan-6
Bitstream Decryption	HW decryption	HW decryption	HW decryption	HW decryption	HW decryption	HW decryption
Encryption Algorithm	(Triple) DES	AES, 256 bit key	AES, 256 bit key	AES, 256 bit key	AES, 256 bit key	AES, 256 bit key
Encryption key store	BBRAM	BBRAM	BBRAM	BBRAM/eFUSE	BBRAM/eFUSE	BBRAM/eFUSE
PR after encrypted configuration	Not allowed	Through ICAP	Not Allowed	Through ICAP	Through ICAP	Not Allowed
ICAP bus width w/o vs w/ encryption	8 bits/8 bits	32 bits/8 bits	32 bits/8 bits	32 bits/8 bits	32 bits/8 bits	16 bits/ 8 bits
Bitstream authentication	No	No	No	SHA 256	SHA 256	No
Frame-wise CRC	No	No	No	No	Yes	No

Note that none of these FPGA families allow for configuration readback after the initial configuration with an encrypted bitstream.

2.2. Related State of the Art

The earliest works on secure configuration of FPGAs were done by Kean [2001] and Castillo et al. [2005]. These works raise the concern over bitstream security and introduce secure storage of encrypted bitstreams on external memory. Given the current FPGA security features, their works are now obsolete.

Zeineddini and Gaj [2005] consider the use of an embedded microprocessor within the FPGA as the configuration controller to achieve secure partial configuration. However, it considers the entire board as a secure platform, not just the FPGA device, being susceptible to physical tampering between the FPGA and the on-board memory.

Chaves et al. [2008] propose the use of a dedicated internal hardware module to perform attestation of incoming bitstreams and to enforce region delimitation, where only the regions intended to be reconfigured can be modified. This solution validates the bitstreams without affecting the reconfiguration performance and while also filtering out the commands that are not allowed to be executed. This approach does not assure confidentiality and only verifies the integrity after the bitstream is completely loaded.

Glas et al. [2008] propose a programming port called *user-JTAG*. Their solution first verifies the integrity of the bitstream in user logic and then feeds the bitstream back to a modified JTAG interface for configuration. However, this solution requires a trusted block outside the FPGA that needs to be secure and tamper-proof; otherwise, the bitstream can be tampered after the integrity check.

Hori et al. [2012] propose a secure dynamic reconfigurable system using AES in Galois/Counter Mode (AES-GCM). This solution is improved in Hori et al. [2013] by splitting the encrypted bitstream into several blocks and validating each block individually. Similar block-wise integrity checking is now provided in the more recent FPGAs. This solution also integrates a Physically Unclonable Function (PUF) for device-dependent key generation.

Abdellatif et al. [2013] propose an architecture that utilizes a single AES core to perform bitstream encryption and authentication in order to minimize area cost. They use a compact 32-bit AES core. The proposed method achieves a maximum throughput of 407Mbps, half of the throughput provided by the built-in security mechanism. Moreover, the solution does not consider replay attack prevention and on-the-fly bitstream validation.

Parelkar and Gaj [2005] also give a solution for authenticated encryption using EAX mode with AES. This work is targeted at older FPGAs without any built-in authentication mechanism. Like the previous work, a robust solution for secure dynamic reconfiguration of FPGAs is not provided.

More elaborate solutions have been proposed for secure download of bitstreams from external intellectual property (IP) providers.

Drimer and Kuhn [2009] propose a secure remote update protocol with a common key for encryption and authentication. This proposal assumes a secure external nonvolatile memory to store the key and other secret information.

Gaspar et al. [2012] use AES in CBC mode to provide confidentiality and authentication to partial configuration bitstreams. However, the solution requires two secret keys to be preinstalled on the FPGA, which cannot be changed later. This makes the FPGA prone to replay attacks with outdated bitstreams. Zhang et al. [2015] use a reconfigurable PUF-FSM binding method to prevent replay attacks against the FPGA.

Kepa et al. [2010] propose a mechanism with encryption, key sharing, and signature generation; all implemented in software based on a Root of Trust approach. This solution requires an additional trusted party and imposes significant overhead to the system. The overhead caused by this software solution is not presented.

The solution proposed by Guneyasu et al. [2007] uses both public key and symmetric key cryptography, as well as the Diffie-Hellman (DH) key exchange protocol. Their solution uses the public key cryptography only once for authenticating the DH half keys. The public-key functionality is implemented using a temporary configuration bitstream for a one-time procedure. However, their work does not consider unique half-key generation and their depreciation, resulting in possibility of replay attacks with outdated configuration bitstreams.

Other than the following two approaches, none of the above consider replay attacks.

Braeken et al. [2011] propose a solution to securely download bitstreams from an external bitstream source similarly to Drimer and Kuhn [2009]. The solution does not consider partial reconfiguration and focuses on reducing resource overhead over configuration performance. Moreover, the solution requires a secure and tamper-proof external memory. Although it uses the Station-To-Station protocol for key and entity authentication, the solution prevents man-in-the-middle attacks and replay attacks. Vliegen et al. [2013] extended this solution by considering partial reconfiguration and without an external memory. The partial configuration bitstream being received, from an external source, is decrypted and stored in BRAMs inside the FPGA, while the authentication tag is computed. Although they consider dedicated bitstream compression, the overhead caused by the local bitstream storage is significant, leaving less resources for the user logic. The authors added an external memory to store the encrypted bitstream in their recent work [Vliegen et al. 2014]. However, this solution discards the decrypted blocks after validation and fetches them again for reconfiguration, allowing the bitstream to be tampered between the validation and the reconfiguration processes during its external storage and transmission. In any of its versions, the solution becomes less suitable for applications requiring frequent HW reconfiguration.

To solve this problem, Devic et al. [2012] consider the external bitstream storage of multiple partial configuration bitstreams simultaneously and propose a system to prevent system downgrade, assuming two possible scenarios with either a fully secure board or only a secure FPGA device. The solution uses AES-CBC and HMAC for bitstream security. They consider the use of a partial configuration bitstream tag as the countermeasure against replay attacks, which is embedded as a prefix of the bitstream. However, the solution does not perform on-the-fly integrity verification of the bitstream during the partial reconfiguration process. This may result in the static section or an

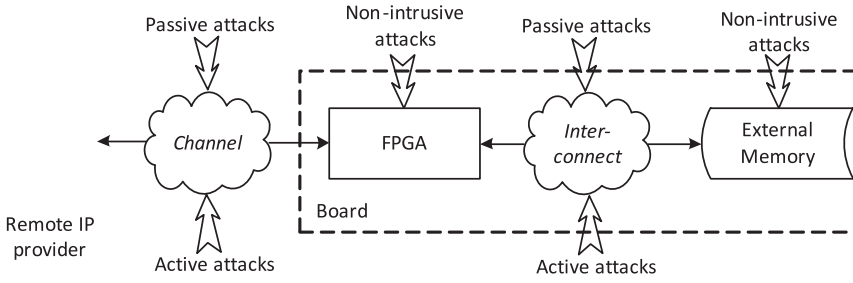


Fig. 1. Threat model for the proposed solution.

unwanted partition being overwritten when the bitstream is tampered between the FPGA and the external memory.

3. THE PROPOSED SOLUTION

Given the existing technology support and the proposed related art, herein a system capable of providing the required security for partial dynamic reconfiguration is proposed, in particular the protection against replay attacks, confidentiality of the bitstream, reliable and on-the-fly detection of bitstream tampering, and the ability to externally store and support multiple bitstream configurations. Additionally, and considering systems that fully explore the partial dynamic reconfiguration capabilities, the proposed solution also strives for the lowest impact in the reconfiguration performance. The main idea to achieve this goal is the re-encryption, with a unique random key, of the remotely received bitstream, before storing it on the external, potentially unsecured, memory.

3.1. Threat Model

The threat model considered for the proposed solution consists of five entities, which are shown in Figure 1.

3.1.1. The Channel. Partial configuration bitstreams are sent by the IP providers to the remote system containing the FPGA. The partial bitstreams are transmitted over conventional channels, such as the Internet, and are susceptible to both passive as well as active attacks. In passive attacks, the attacker can eavesdrop on the channel to retrieve data. In active attacks, the attacker may actively manipulate the messages transmitted over the channel. The communicating parties may be completely unaware of active attacks on the channel, such as in case of man-in-the-middle attacks. Both passive and active attacks on the channel are considered in the threat model.

3.1.2. The FPGA Device and the External Memory. It is assumed that the system consisting of the FPGA device and the external memory is remotely deployed and is not secured. Therefore, both intrusive as well as nonintrusive attacks are possible on the system. However, this threat model considers the FPGA as a secure device. Intrusive and non-intrusive, such as side channel analysis and fault injection attacks, are not considered in this threat model.

3.1.3. The Interconnect. The proposed solution stores the partial bitstreams on the external unsecured memory after being processed inside the FPGA. Later, the bitstreams are read from the external memory during each reconfiguration. Bitstreams may be subjected to both active and passive attacks during transmission between the FPGA and the external memory. These attacks are considered in the threat model.

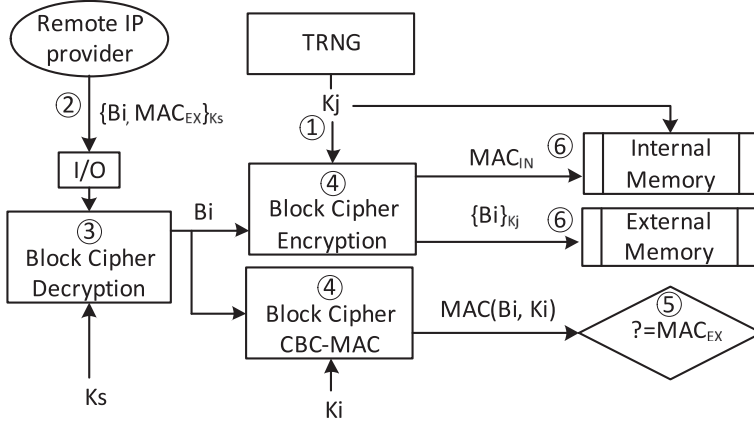


Fig. 2. Phase 1: Validation and storage of the received bitstream.

3.2. Methodology

The proposed reconfiguration process is divided into two phases, namely (i) phase 1, the reception of the bitstream and storage on the local external memory; and (ii) phase 2, the actual reconfiguration process using one of the stored bitstreams.

3.2.1. Phase 1: Validation and Storage of the Remotely Received Bitstream. In phase 1, the partial configuration bitstream (B_i) is received from a remote IP provider via a standard I/O interface (such as an Ethernet connection) and the bitstream is deciphered using the session key (K_s) shared with the remote IP provider, within the closed environment of the FPGA. The deciphered bitstream is then re-encrypted, still within the FPGA, using a unique random key (K_j), generated using a TRNG, and sent to the high-capacity, potentially unsecured, external memory. While the bitstream is being re-encrypted, the bitstream authentication MAC is computed using the authentication key (K_i) shared with the IP provider. After completing the download of the partial configuration bitstream, the calculated MAC value is compared with the one (MAC_{EX}) securely received from the remote IP provider, in order to validate the authenticity of the received bitstream. Both the re-encryption key (K_j) and the computed MAC value, which is the last encrypted block in CBC mode using K_j , are stored in the secure internal memory of the device. The last encrypted block, hereafter termed as MAC_{IN} , is used in phase 2 to validate the authenticity of the retrieved bitstream from the external memory. The considered bitstream processing in phase 1 is depicted in Figure 2.

The operations performed in phase 1 are as follows:

- (1) Generate a random key K_j using a true random number generator, which is unique for each of the received configuration bitstream data.
- (2) Receive $\{B_i, MAC_{EX}\}_{K_s}$ encrypted using K_s from the bitstream source.
- (3) Decrypt the received $\{B_i, MAC_{EX}\}_{K_s}$ in CBC mode.
- (4) Encrypt B_i using K_j , for storage in the external memory, and calculate the configuration bitstream MAC using the preshared authentication key K_i .
- (5) Compare the calculated MAC with the MAC_{EX} received from the bitstream source. If they do not match, abort the bitstream storage process.
- (6) Store the encrypted bitstream $\{B_i\}_{K_j}$ in the external memory and store the generated key (K_j) along with the last encrypted block internally in the FPGA. This last encrypted block is used in phase 2 as the CBC-MAC (MAC_{IN}) of the stored bitstream.

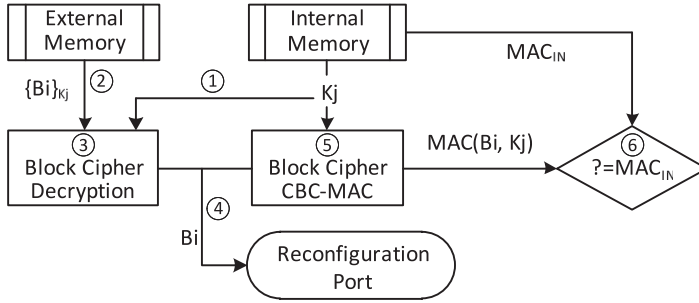


Fig. 3. Phase 2: Reconfiguration using the externally stored bitstream.

3.2.2. Phase 2: Reconfiguration Using the Stored Bitstream. Phase 2 of the proposed solution regards the actual partial dynamic reconfiguration of the device. In this phase, the selected partial configuration bitstream is downloaded from the external memory and sent to the reconfiguration port of the FPGA, using the flow depicted in Figure 3. Since the bitstream downloaded from the external memory is encrypted, it is first decrypted using the unique key (K_j) only known internally in the FPGA. The obtained bitstream is then sent to the reconfiguration port, and simultaneously its CBC-MAC value is computed to assure its authenticity. After termination of this process, the resulting MAC value is compared with the internally stored MAC (MAC_{IN}) obtained in phase 1.

The operations performed in phase 2 are as follows:

- (1) Obtain the unique random key K_j and the MAC_{IN} from the internal memory, corresponding to the configuration bitstream Bi .
- (2) Read the encrypted bitstream $\{Bi\}_{K_j}$ from the external memory.
- (3) Decrypt $\{Bi\}_{K_j}$ using the key K_j , in CBC mode.
- (4) Send the decrypted bitstream to the configuration port for dynamic reconfiguration.
- (5) Encrypt the decrypted bitstream using the same key K_j in CBC mode to calculate the last CBC block.
- (6) Compare the last CBC block with the corresponding MAC_{IN} . If they match and no error is reported by the reconfiguration port, the operation concludes with success. Otherwise, an error is outputted.

With the proposed solution, two additional bitstream validation mechanisms of the configuration port, present in the modern FPGA devices, are also considered, namely (i) the detection of improperly formatted/decrypted bitstreams and (ii) the frame-by-frame CRC. These two detection features complement the proposed solution in the on-the-fly detection of bitstream corruption.

3.3. Proof of Security

This section describes how the proposed method, combined with the built-in mechanism for detecting unformatted bitstreams and the frame-wise CRC, enables secure dynamic reconfiguration using externally stored partial bitstreams.

Bitstreams have a well-defined format, comprised of a header, the control and configuration data, and a footer consisting of NOOP words. A Sync word separates the header and the configuration data, and a Desync word separates the configuration data and the footer. The configuration interface ignores all data before the Sync word and all data after the Desync word. The format of the partial configuration bitstream for the Xilinx 7 series FPGA devices is illustrated in Figure 4.

The control and configuration data begins with the commands to set up control and status registers for configuration. The configuration frames appear next, each

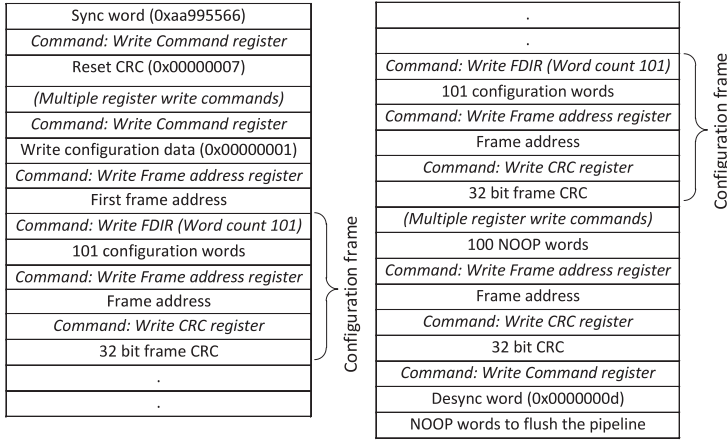


Fig. 4. The format of Xilinx 7 series FPGA XC7VX485T partial configuration bitstream.

containing the instructions and data for frame dimension, configuration data, frame address, and the corresponding CRC value calculated over the whole frame. Given this and the fact that in the proposed solution each partial configuration bitstream is encrypted with a unique key in CBC mode, the attempt to use a different bitstream (ciphered with a wrong key) or the manipulation of a ciphered data block will result in erroneous configuration commands that do not make sense. In this scenario, the ICAP interface will raise an error while processing the first erroneous command.

Moreover, the proposed solution encrypts the bitstream in CBC mode, where a single bit manipulation affects not only the current cipher block after decryption but also the following one. Encryption using a standard block cipher in CBC mode will ensure that the tampered configuration commands in the two cipher blocks are rejected by the configuration port.

As described in Section 2.1, the frame-by-frame CRC provided by the recent Xilinx 7 series FPGAs ensures the integrity of each frame detailed in the partial configuration bitstream. If the manipulation is performed only on the part of the bitstream detailing the content of the FPGA's memory, the configuration port will only generate an error when testing the frame CRC before configuring the fabric. Given the diffusion and confusion provided by block ciphers, the CRC field is less vulnerable, allowing a more secure CRC validation. Even if a single bit of the bitstream is changed outside the FPGA, the entire decrypted block will be affected, the frame CRC will fail, and the frame will not be configured. Subsequent correct frames will be configured if the configuration process is not aborted immediately. However, the FPGA will never be configured with a tampered or erroneous frame, and the static partition will never be overwritten. The ICAP status port raises the signal for CRC error, which can be read to take the necessary action for recovery, such as load an alternate partial module, reprogram the whole device, and report to the remote administrator.

Identically to the existing related state of the art, the proposed solution assumes that the FPGA is initially configured with a legitimate encrypted full bitstream. The built-in full bitstream decryption mechanism and the internal key ROM can be used for this purpose. Secret information such as a private key or a symmetrical shared key can be embedded in the initial bitstream. These secret values are then used to establish the authentication and session keys with a remote server, which provides the needed partial configuration bitstreams. Modules for communication with the remote server

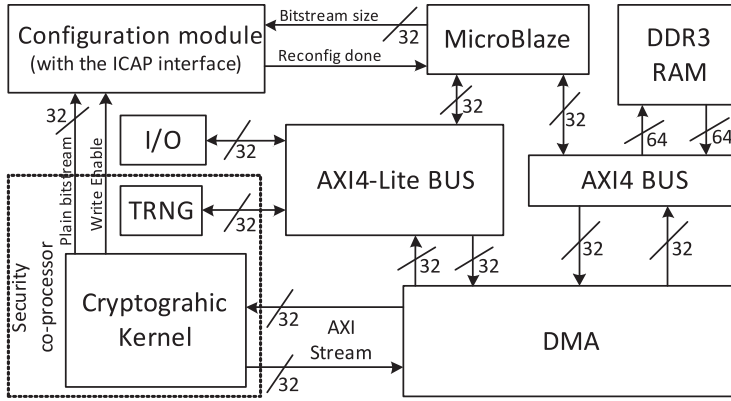


Fig. 5. Architecture of the reconfigurable system.

can be implemented in software, as it does not affect the reconfiguration performance (in phase 2) of the proposed solution.

4. ARCHITECTURE AND IMPLEMENTATION

In order to evaluate the proposed solution, a prototype was implemented on a Xilinx VC707 development board containing a Virtex-7 series FPGA device. The decryption, re-encryption, and CBC-MAC operations of the proposed solution are implemented in a single module, hereafter termed as the cryptographic kernel. Nevertheless, these operations are dependent on some other supporting peripherals, such as the I/O interface and the external memory for bitstream transfer, a random number generator for unique re-encryption keys, and the ICAP interface for the reconfiguration process in phase 2. It should be noted that in the Xilinx 7 series FPGAs, the internal configuration interface is designated as ICAPE2.

For the supporting peripherals, an embedded platform was deployed using the cryptographic kernel module, the peripherals themselves, and a central processor to control the operations and the communications among different peripherals. The resulting architecture is illustrated in Figure 5. The directional arrows represent the AXI buses and dedicated channels used for the transfer of bitstreams, keys, and control data.

The supporting architecture for the reconfiguration process includes a MicroBlaze processor, a DDR3 external memory, a DMA controller, an I/O interface, and the configuration module capable of handling errors derived from incorrect configuration data. The security components of the proposed solution, namely the cryptographic kernel and the TRNG, together are herein defined as the security co-processor, as depicted in Figure 5. It is connected to the external memory via the AXI Stream buses, the DMA interface, and the AXI4 buses, and to the central processor via the AXI4-Lite buses.

The proposed system was prototyped using Xilinx EDK 14.5 tools. Apart from the configuration module, all the other components of the system are connected using either a memory mapped or a stream interface of the AXI bus. Peripherals connected to this bus can run at different frequencies in a nonarbitrated mode.

To optimize the performance of the system, a DMA controller is used in all bitstream transfers to and from the external memory. The DMA interfaces with the cryptographic kernel using the two unidirectional AXI Stream channels. The DMA engine reads the encrypted bitstream from the external memory into the cryptographic kernel. Once decrypted, the bitstream is transferred to the ICAP via the configuration module.

The configuration module includes a BRAM-based FIFO used to temporarily store the bitstream while sending it to the ICAP. This FIFO has a different clock domain for the ICAP interface, operating at 100MHz, the maximum operating frequency of the ICAP. This allows the cryptographic kernel to operate at its maximum frequency, higher than 100MHz. This allows for a full streaming operation, maximizing the throughput of the co-processor and, consequently, the reconfiguration performance.

The output of the ICAP interface is fed to the control logic. During configuration, this output port gives the status of the ongoing configuration process [Xilinx 2012]. This signal is used by the control logic to maintain the correct state of the reconfiguration process. By default, in case of a CRC error, the ICAP interface will continue the configuration using the subsequent correct frames, if CSIB is held asserted. However, the configured partition will not have any valid logic in it. To prevent this, the implemented solution disables CSIB when a CRC mismatch is signaled through the ICAP out port.

Additionally, the configuration module receives the bitstream length parameter from the central processor. This parameter is used to detect a failure in the bitstream transfer performed via the DMA engine and the cryptographic kernel. After a successful reconfiguration, the configuration module informs the central processor, using the reconfiguration done signal.

While the session and authentication keys are established with the IP provider, the unique random re-encryption key (K_j) needs to be internally generated. For this purpose, an oscillator ring-based TRNG [Wold and Tan 2008] is included into the architecture and accessed by the MicroBlaze via the AXI4-Lite bus. This ensures that each bitstream can be encrypted with a unique key before it is stored in the external memory.

The cryptographic kernel performs decryption, re-encryption, and MAC calculation on, continuous as well as discontinuous, streams of data. The operation of the cryptographic kernel is defined by the first 32-bit packet sent by the main processor, specifying whether it is going to receive a new bitstream from the IP provider or a partial reconfiguration has been triggered. This 32-bit control word also specifies the bitstream length. Following the control word, the unique encryption key (K_j) and, in case of phase 1, also the session key (K_s) and the authentication key (K_i) are sent to the cryptographic kernel. After this initialization, the cryptographic kernel will either receive an external partial configuration bitstream (phase 1) or retrieve a stored partial configuration bitstream to upload to the ICAP interface (phase 2). The control and data packets are sent by the main reconfiguration processor (in this prototype a MicroBlaze processor) using DMA transfers. The MicroBlaze processor is only required for pre-setting the DMA buffer descriptor registers, thus removing the bottleneck imposed by the MicroBlaze processor on the data transfer speed.

In phase 1, the external encrypted bitstream is directly transferred from the I/O to the cryptographic kernel to be decrypted. As the blocks of the plaintext bitstream are obtained, they are re-encrypted and the CBC-MAC is calculated. The re-encrypted bitstream blocks are directly sent to the external memory through a different stream channel, thus avoiding bus collisions. Associated with the DMA controller, this makes the encryption cores the only bottleneck of the added cryptographic kernel. Once the entire bitstream is re-encrypted, the computed MAC is sent to the MicroBlaze processor to be compared with the expected value (MAC_{EX}).

Afterward, in phase 2, the bitstream is sent from the external memory to the configuration module via the cryptographic kernel, using a DMA transfer. In this phase, the cryptographic kernel decrypts the received data, using the unique encryption key (K_j), and sends it to the configuration module. In parallel, the MAC value is recomputed and sent to the MicroBlaze for a final bitstream validation before using the FPGA logic just reconfigured.

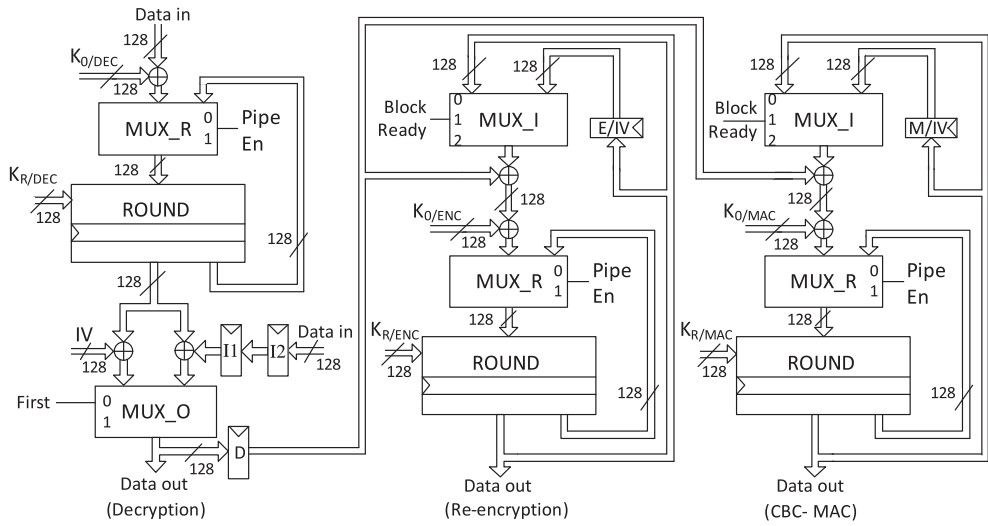


Fig. 6. Datapath of the 3-AES crypto kernel.

Note that the critical values, such as keys and MAC values, are stored internally in the FPGA, specifically in the MicroBlaze local memory, composed of BRAMs.

The cryptographic kernel performs the decryption, re-encryption, and CBC-MAC operations using AES in CBC mode. A folded AES-128 implementation [Chaves et al. 2006] is used as the base cipher engine. This AES implementation outputs a 128-bit block of data after every 10 cycles of execution and can perform both encryption as well as decryption.

Since three different cryptographic operations are performed in phase 1 of the considered reconfiguration process, three separate instances of the AES core can be used. This allows to perform the decryption, re-encryption, and MAC computation over consecutive 128-bit blocks of the partial configuration bitstream in a pipeline. Consequently, the cryptographic kernel is able to achieve throughput identical to the considered AES core in both phase 1 and phase 2 of the proposed solution. However, in terms of resource requirements, the cryptographic kernel is not optimized due to the use of the three separate instances of the AES core. In order to improve the resource uses, the cryptographic kernel can be designed using only one instance of the AES core to sequentially perform decryption, re-encryption, and CBC-MAC computation.

Herein, the cryptographic kernel is developed using both of the approaches, that is, using three AES core instances and using a single AES core instance. The resulting architectures of the cryptographic kernel are herein designated as (i) the 3-AES crypto kernel and (ii) the 1-AES crypto kernel, respectively. These two approaches of the cryptographic kernel design are discussed in the following.

4.1. The 3-AES Crypto Kernel

The architecture with the 3-AES crypto kernel considers the use of a dedicated AES core for each of the decryption, re-encryption, and MAC operations, as proposed in Kashyap and Chaves [2014]. In this kernel, the output of the decryption core is used by the cores for re-encryption and the CBC-MAC operations. The datapath of the 3-AES crypto kernel is illustrated in Figure 6.

Each 128-bit bitstream block is first sent to the decryption core. The resultant 128-bit decrypted bitstream block is stored in register D to be used as the input for the re-encryption and the CBC-MAC operations.

The considered AES core [Chaves et al. 2006] is folded, that is, the same core component, consisting of MUX_R and the ROUND component performs the 10 rounds of the AES cipher in 11 execution cycles. The MUX_R multiplexer selects the input in the first round and the feedback value from previous round in the later rounds of AES operation. The ROUND component implements an AES round with part of the operations implemented as T-boxes in a single BRAM. The output of the BRAM inside the ROUND component is registered. Therefore, during the 11th cycle of AES operation, the next data block can already be selected by MUX_R. This allows the AES core to process a new 128-bit input block in every 10 execution cycles.

The decryption results are generated during the 11th execution cycle. Therefore, the re-encryption and the CBC-MAC operations can only start during the 12th execution cycle, and these results are obtained with a latency of 21 execution cycles. However, the decryption, re-encryption, and CBC-MAC operations are pipelined. This allows the cryptographic kernel to process a 128-bit block in every 10 cycles.

When decrypting in CBC mode, the output data needs to be XORed with the previous input block or the initialization vector (IV). This selection is achieved by the MUX_O multiplexer, as depicted in Figure 6. The previous and the current input bitstream blocks are stored in registers I1 and I2, respectively, in order to be available for this XOR operation.

However, in case of CBC mode encryption, the input data is first XORed with the previous encrypted block or the IV, before being encrypted. In the considered datapath, this selection is achieved by the MUX_I multiplexer. The E/IV and the M/IV registers are initialized with the IV when processing the first block of the bitstream. While processing the later bitstream blocks, these registers store the results of the re-encryption/MAC operations. The MUX_I multiplexer selects the encryption output (Data out) for continuous stream of data.

4.2. The 1-AES Crypto Kernel

In this proposed approach to the cryptographic kernel, a single AES core is used to perform decryption, re-encryption, and MAC calculation operations. This resource reduction comes at the cost of the sequential execution of the cryptographic operations in phases 1 and 2.

The datapath of the 1-AES crypto kernel is depicted in Figure 7, excluding register D_R (in grey). Register D_R is used in a modified single AES-based crypto kernel discussed later. As depicted, the results of the decryption, re-encryption, and CBC-MAC operations are stored in the registers D, E, and M, respectively. During decryption, the MUX_I1 multiplexer selects 0's because the input is fed unaltered. For re-encryption and MAC computation, the multiplexer selects the IV for the first block and registers E and M for the following blocks. The MUX_I2 multiplexer selects either the input data block during decryption, or register D during re-encryption and MAC computation. Since the AES round 0 keys for the decryption, re-encryption, and MAC operations are stored in separate registers, they are selected using the MUX_K multiplexer. The MUX_R and MUX_O multiplexers and the ROUND component perform the same functions as discussed in Section 4.1.

Similar to the 3-AES crypto kernel, the decryption result is generated during the 11th cycle and stored in the register D. Therefore, the re-encryption and CBC-MAC operations can only begin in the 12th cycle, and the 1-AES crypto kernel will be idle during the 11th cycle, waiting for the decryption result to be available in register D. Consequently, the kernel provides a throughput of 31 cycles in phase 1 (for decryption,

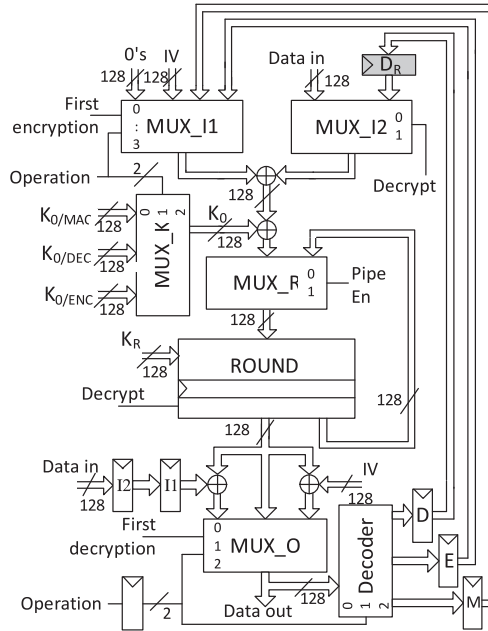


Fig. 7. Datapath of the single AES-based crypto kernels: (i) 1-AES crypto kernel (without D_R) and (ii) 1-AES crypto kernel 2 (with D_R).

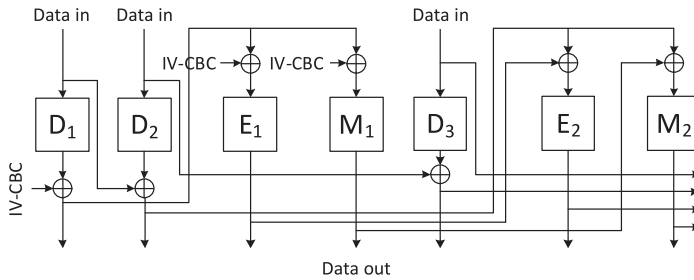


Fig. 8. Dataflow through the 1-AES crypto kernel 2 in phase 1.

re-encryption, and MAC) and 21 cycles in phase 2 (for decryption and MAC), for each 128-bit block of the bitstream.

4.3. The 1-AES Crypto Kernel 2

In the 1-AES crypto kernel discussed above, if the data dependency between consecutive decryption and re-encryption/MAC computation is removed, the throughput can be improved to 30 cycles in phase 1 and 20 cycles in phase 2, for each 128-bit block of bitstream data. In order to achieve this throughput, the 1-AES crypto kernel is further modified. The modified kernel is herein designated as the 1-AES crypto kernel 2.

The only difference of this kernel regarding the 1-AES crypto kernel is that the next data block is decrypted before the encryption of the current block. This eliminates the data dependency between consecutive decryption and re-encryption/MAC computation. The flow of bitstream data among the cryptographic operations in phase 1 is depicted in Figure 8. The suffix represents the block number; for example, D_n , E_n , and M_n

represent the decryption, re-encryption, and MAC operation of the n -th data block, respectively.

As illustrated in Figure 8, the D_{i+1} operation is performed before the E_i and M_i operations on the previous block. Therefore, the E_i operation in phase 1 (or the M_i operation in phase 2) can start during the 11th execution cycle of the D_{i+1} operation, since the data dependency no longer exists. As the modified kernel does not need to wait an extra cycle for the result of the decryption operation, before performing the re-encryption and CBC-MAC operations, it is able to achieve a throughput of 128 bits per 20 cycles of operation in phase 2.

However, with this approach, the results of two consecutive decryption operations need to be stored. To achieve this, register D_R , illustrated in grey in Figure 7, is used in order to store the result of the previous decryption operation.

In phase 2, the MAC verifies the integrity and authenticity of the partial configuration bitstream stored externally. However, with the built-in frame-wise CRC feature of the Xilinx 7 series FPGAs combined with the CBC mode encryption, the MAC only serves as an additional authentication mechanism. The MAC calculation is useful only when the proposed solution is implemented on older FPGAs without frame-wise CRC.

Therefore, when considering the Xilinx 7 series FPGA devices, the CBC-MAC verification in phase 2 is not necessary. Security-wise, the solution will still be uncompromised, since the authenticity and integrity of the bitstream is verified by the frame-wise CRC combined with the confusion and diffusion provided by the block cipher in feedback mode. In all kernel versions, replay attacks are not possible because a unique re-encryption key is used for each bitstream. Tampering attacks will be detected on the fly by the frame-wise CRC combined with the block cipher encryption.

Performance-wise, the omission of the CBC-MAC operation in phase 2 allows to significantly improve the throughput of the 1-AES crypto kernel. When performing only decryption in phase 2, the 1-AES crypto kernel is able to achieve a throughput of 128 bits per 10 cycles, identical to the 3-AES crypto kernel.

5. EXPERIMENTAL RESULTS

To evaluate the proposed secure dynamic reconfiguration system, experimental results were obtained for the resulting prototype on a Xilinx Virtex-7 FPGA device XC7VX485T-2, using Xilinx ISE 14.7 and EDK 14.7 tools. Additionally, experimental results were also obtained on older FPGAs devices, namely on Virtex-5 and Virtex-6 devices, in order to compare with the most relevant state of the art.

5.1. Resource Requirements and Performance Evaluation

To analyze the impact of the proposed solution, one base reconfigurable system is implemented without security, presenting the core architecture supporting the partial reconfiguration process. The resources used by the base reconfigurable system consist of the MicroBlaze processor, the buses, the DMA controller, the memory controller, the communication controller, and the configuration module. The obtained post Place&Route results for the base system on a Virtex-7 FPGA presented in Table II suggest that the base system requires 14,463 Slice registers, 14,608 Slice LUTs, 48 BRAMs, 3 DSP Slices, and an ICAP instance.

The resource requirements of the security co-processor implementations using the three cryptographic kernel architectures on a Virtex-7 FPGA are presented in Table III. The 3-AES crypto kernel requires more FPGA resources than the single AES-based cryptographic kernels due to the use of three instances of the fully folded AES.

Table IV presents the FPGA resources required by the implementations of the reconfigurable system with and without the security co-processors. The implemented base system occupies 6,984 Slices and 48 BRAMs, that is, 9.2% and 4.7% of the

Table II. Resource Requirements for the Individual Components of the Base System

Component	Slice Reg	Slice LUT	BRAM	DSP
MicroBlaze	2,368	2,395	38	3
AXI4	3,330	2,474	2	0
AXI4-Lite	366	483	0	0
DMA	3,202	2,715	6	0
Configuration module	92	127	2	0
DDR3	5,016	6,300	0	0
UART	89	114	0	0
Total	14,463	14,608	48	3

Table III. Resource Requirements of the Three Security Co-Processor Architectures

Component	Slice Reg	Slice LUT	BRAM
3-AES crypto kernel			
Crypto kernel	1,351	1,956	30
TRNG	83	74	0
Total	1,434	2,030	30
1-AES crypto kernel			
Crypto kernel	1,268	1,398	10
TRNG	83	74	0
Total	1,351	1,472	10
1-AES crypto kernel 2			
Crypto kernel	1,332	1,445	10
TRNG	83	74	0
Total	1,415	1,519	10

Table IV. Resource Requirements of the Base System and the Security Co-processor

Implementation	Slices	BRAM	DSP	ICAP
Base system	6,984 (9.20%)	48 (4.7%)	3 (0.1%)	1 (50%)
Base + 3-AES crypto kernel + TRNG	7,758 (10.22%)	78 (7.6%)	3 (0.1%)	1 (50%)
Base + 1-AES crypto kernel+ TRNG	7,563 (9.96%)	58 (5.6%)	3 (0.1%)	1 (50%)
Base + 1-AES crypto kernel 2 + TRNG	7,579 (9.99%)	58 (5.6%)	3 (0.1%)	1 (50%)
Base + 1-AES kernel w/o MAC + TRNG	7,567 (9.97%)	58 (5.6%)	3 (0.1%)	1 (50%)

available resources on the target FPGA, respectively. Performance-wise, the base system operates at 100MHz. Since the transfer of data is performed via DMA, a maximum throughput of 3.2Gbps (32-bit words at 100MHz) can be achieved, which is the limit of the ICAP interface. These results also suggest that the complete system with the proposed security co-processor implemented using the 3-AES crypto kernel requires 7,758 Slices and 78 BRAMs, that is, the co-processor requires an additional 774 Slices and 30 BRAMs. These results allow to conclude that the proposed security co-processor imposes an overall resource increase of 1.02% regarding the base reconfiguration solution. Performance-wise, the kernel is able to operate at maximum frequency of 196MHz. At this frequency, the kernel achieves throughput of 2.5Gbps, 22% lower than the maximum reconfiguration throughput of the ICAP interface. This limit is imposed by the AES cores. If faster cryptographic implementations are considered, lower reconfiguration times can be achieved. Note that the cryptographic kernel and DMA use a different clock, being able to operate at a different frequency than the base system.

The security co-processor implementation using the 1-AES crypto kernel requires additional 579 Slices and 10 BRAMs, in regard to the base reconfiguration system, which is 0.76% and 0.97% of the available resources, respectively, whereas the implementation using the single 1-AES crypto kernel 2 requires 595 additional Slices and 10 additional BRAMs.

Each of the single AES-based cryptographic kernels is able to operate at a maximum frequency of 196MHz, due to the identical critical path as the 3-AES crypto kernel. If the MAC operation is considered in phase 2, the 1-AES crypto kernel requires 21 cycles for each 128-bit block of bitstream data in phase 2, achieving a reconfiguration throughput of 1,194Mbps, whereas the 1-AES crypto kernel 2 requires 20 cycles for each 128-bit block of the bitstream in phase 2, thereby achieving a reconfiguration

Table V. Comparison of the State of the Art in Terms of Configuration Bitstream Security

Approach	[Braeken et al. 2011]	[Vliegen et al. 2013]	[Hori et al. 2013]	[Devic et al. 2012] (FiT)	Ours
Partial reconfiguration	No	Yes	Yes	Yes	Yes
Secure external bitstream storage	No	No	Yes	Yes	Yes
On-the-fly bitstream verification	No	No	Yes	No	Yes
Detection of tampered bitstream	No	Before configuration	On tampered block	After configuration	On tampered frame
Replay attack protection	No (attack on board)	Yes	No	Yes	Yes

throughput of 1254 Mbps. If the MAC operation is not considered in phase 2, the single AES-based cryptographic kernels require 10 cycles for each 128-bit block of bitstream data in phase 2, thereby achieving a reconfiguration throughput of 2.5Gbps, identical to the maximum throughput of the 3-AES crypto kernel.

The built-in security mechanism of the Xilinx FPGAs significantly reduces the maximum reconfiguration speed of the ICAP interface. The ICAP interface can receive encrypted bitstream only through the 8-bit input port. Therefore, the built-in security mechanism provides a reconfiguration throughput of 800Mbps. The proposed solution achieves a maximum throughput of 2.5Gbps, 3 times higher than the maximum throughput of the built-in security mechanism.

5.2. Comparison with the Related State of the Art

The proposed solution is compared with the related state of the art in terms of the security features, resource requirements, and performance. Table V presents a qualitative analysis of the state of the art in terms of configuration bitstream security.

Toward a quantitative analysis of the proposed solution with the related state of the art, the proposed security co-processor was also mapped to other FPGA technologies. The post Place&Route results, obtained using the Xilinx tools, are presented in Table VI. Note that in the Virtex-5 devices, AXI peripherals are not available. Nevertheless, the results for the proposed solution still consider an AXI interface rather than a Processor Local Bus (PLB) interface. An AXI to PLB bridge can be considered to interface the security co-processor to the other peripherals.

Braeken et al. [2011] propose a solution providing bitstream confidentiality, authentication, and freshness. However, the system is susceptible to on-board attacks, as the solution does not provide secure external bitstream storage. Vliegen et al. [2013] do not provide on-the-fly bitstream verification. The system first stores the entire bitstream internally and verifies its authenticity before starting the reconfiguration process. This limits the scalability and performance of the solution. Resource-wise, this solution cannot be fairly compared because the solution in Vliegen et al. [2013] also includes dedicated hardware for asymmetrical cryptography. Moreover, the additional BRAM resources for the internal bitstream storage have also to be considered. No reconfiguration performance metrics are presented by the authors. In their recent work, Vliegen et al. [2014] consider the use of an external memory to store the encrypted bitstream. However, the bitstream can be tampered between the two fetches from the external memory for validation and reconfiguration. The solution in Vliegen et al. [2014] is 31 times slower than the solution herein proposed, while still requiring 2.5 times more Slice resources.

Table VI. Performance Analysis and Comparison with the Related State of the Art

Approach	Area (Slices)	Slice registers	Slice LUTs	BRAMs	DSP	Throughput
Xilinx XC7VX485T-2 FPGA device						
3-AES crypto kernel	774	1,434	2,030	30	0	2.5Gbps
1-AES crypto kernel	579	1,351	1,472	10	0	1,194Mbps
1-AES crypto kernel 2	595	1,415	1,519	10	0	1,254Mbps
1-AES crypto kernel w/o MAC	583	1,288	1,475	10	0	2.5Gbps
Xilinx XC5VFX70T-1 FPGA device						
[Braeken et al. 2011]	3,871	NA	NA	21	0	NA
[Vliegen et al. 2013]	2,985	NA	NA	15	2	NA
[Vliegen et al. 2014]	2,378	NA	NA	21	7	67Mbps
Ours (3-AES crypto kernel)	919	2,078	2,046	30	0	2,077Mbps
Ours (1-AES crypto kernel 2)	700	1,430	1,232	10	0	1,036Mbps
Xilinx XC5VLX50T-1 FPGA device						
[Hori et al. 2013]	1,074	1,756	2,450	NA	0	913Mbps
Ours (3-AES crypto kernel)	1,031	2,078	2,046	30	0	1,881Mbps
Ours (1-AES crypto kernel 2)	903	1,430	1,232	10	0	940Mbps
Xilinx XC6VLX240T-1 FPGA device						
[Devic et al. 2012]	NA	2,619	5,116	24	0	23Mbps
Ours (3-AES crypto kernel)	768	1,432	2,022	30	0	2,150Mbps
Ours (1-AES crypto kernel 2)	666	1,452	1,379	10	0	972Mbps

The solution proposed by Hori et al. [2013] does not provide replay attack protection. Nevertheless, it still requires more Slices and achieves a reconfiguration performance 51% slower than the implementation using the 3-AES crypto kernel herein proposed.

The solution proposed by Devic et al. [2012] provides both secure external bitstream storage and replay attack protection. It is the most relevant existing approach in regard to the solution herein proposed. However, in their solution, bitstream tampering is detected only after reconfiguration, meaning that the static partition can be overwritten before any tamper is detected.

Quantitatively, the solution herein proposed is able to achieve a reconfiguration time 93 times faster while requiring 45% less Slice registers and 60% less Slice LUTs using the 3-AES crypto kernel. BRAM usage-wise, the 3-AES crypto kernel herein proposed requires more (1.44%). Nevertheless, the overall BRAM usage (7.6%) of the implementation is still lower than the overall Slice usage (10.2%) on the target device. The single AES-based cryptographic kernels herein proposed require 44% less Slice registers, 73% less Slice LUTs, and 58% less BRAMs than the solution in Devic et al. [2012], while achieving 42 times faster reconfiguration performance.

The sources of the solution developed for secure dynamic reconfiguration of FPGAs are available at <http://sips.inesc-id.pt/~rjfc/cores/SecDR/>.

6. CONCLUSION

In this article, a novel approach is proposed to secure the partial dynamic reconfiguration process, in particular when considering an external unsecured memory to store the received bitstreams. The proposed solution considers the security threats against remotely deployed FPGAs and the inadequacy of the built-in security mechanisms of modern SRAM-based FPGAs, such as the lack of freshness assurance, to prevent system downgrade or replay attacks, and the protection of already configured regions using on-the-fly bitstream verification. Particular emphasis has also been given to

reduce the impact of the added security on the FPGA reconfiguration performance as well as the device occupation.

In order to analyze the trade-off between compactness and performance, the proposed solution was implemented with several approaches. In the performance-oriented approach, the configuration data are processed in pipeline, still imposing minimal overhead on the FPGA resources in regard to the state of the art, whereas the resource-optimized approaches use the same design components more efficiently for all the cryptographic operations.

Experimental results suggest that the proposed solution imposes a low footprint, requiring 0.76% to 1.02% of the total available Slice resources on a Virtex-7 FPGA. Performance-wise, a reconfiguration throughput of 2.5Gbps can be achieved, which is 3 times higher than the throughput of the Xilinx built-in security solution. In regard to the most complete solution in the related state of the art, the proposed approach provides a more complete security solution while requiring 45% less Slice resources and being 93 times faster.

In conclusion, the proposed approach provides a complete and on-the-fly security solution for partial configuration bitstreams with less resources and lower performance impact than the state of the art while efficiently allowing for multiple and frequent partial dynamic reconfigurations.

REFERENCES

- Karim M. Abdellatif, R. Chotin-Avot, and H. Mehrez. 2013. Protecting FPGA bitstreams using authenticated encryption. In *Proceedings of the 2013 IEEE 11th International New Circuits and Systems Conference (NEWCAS'13)*. IEEE, 1–4.
- Benoit Badrignans, Reouven Elbaz, and Lionel Torres. 2008. Secure FPGA configuration architecture preventing system downgrade. In *Proceedings of the 2008 International Conference on Field Programmable Logic and Applications (FPL'08)*. IEEE, 317–322.
- An Braeken, Jan Genoe, Serge Kubera, Nele Mentens, Abdellah Touhafi, Ingrid Verbauwhede, Yannick Verbelen, Jo Vliegen, and Karel Wouters. 2011. Secure remote reconfiguration of an FPGA-based embedded system. In *Proceedings of the 2011 6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC'11)*. IEEE, 1–6.
- Javier Castillo, Pablo Huerta, Victor López, and José Ignacio Martínez. 2005. A secure self-reconfiguring architecture based on open-source hardware. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs, (ReConFig'05)*. IEEE, 224–230.
- Ricardo Chaves, Georgi Kuzmanov, and Leonel Sousa. 2008. On-the-fly attestation of reconfigurable hardware. In *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL'08)*. IEEE, 71–76.
- Ricardo Chaves, Georgi Kuzmanov, Stamatis Vassiliadis, and Leonel Sousa. 2006. Reconfigurable memory based AES co-processor. In *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06)*. IEEE, 8–15.
- Florian Devic, Lionel Torres, Jeremie Crenne, Benoit Badrignans, and Pascal Benoit. 2012. SecURe DPR: Secure update preventing replay attacks for dynamic partial reconfiguration. In *Proceedings of the 2012 22nd International Conference on Field Programmable Logic and Applications (FPL'12)*. IEEE, 57–62.
- Saar Drimer. 2008. Volatile FPGA design security—a survey. *IEEE Computer Society Annual Volume* (April 2008), 292–297.
- Saar Drimer. 2009. Security for volatile FPGAs. *Rapport Technique UCAM-CLTR-763, University of Cambridge, Computer Laboratory* (November 2009).
- Saar Drimer and Markus G. Kuhn. 2009. A protocol for secure remote updates of FPGA configurations. In *Proceedings of the 5th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications*. Springer-Verlag, 50–61.
- Tarek El-Ghazawi, Esam El-Araby, Miaoqing Huang, Kris Gaj, Volodymyr Kindratenko, and Duncan Buell. 2008. The promise of high-performance reconfigurable computing. *IEEE Computer* 41, 2 (February 2008), 69–76.
- Lubos Gaspar, Viktor Fischer, Tim Guneyusu, and Zouha Cherif Jouini. 2012. Two IP protection schemes for multi-FPGA systems. In *Proceedings of the 2012 International Conference on Reconfigurable Computing and FPGAs (ReConFig'12)*. IEEE, 1–6.

- Jim Geier. 2002. 802.11 WEP: Concepts and vulnerability. *Wi-Fi Planet 20* (June 2002).
- Benjamin Glas, Alexander Klimm, Oliver Sander, Klaus Müller-Glaser, and Jürgen Becker. 2008. A system architecture for reconfigurable trusted platforms. In *Proceedings of the Conference on Design, Automation and Test in Europe*. ACM, 541–544.
- Tim Guneyssu, Bodo Moller, and Christof Paar. 2007. Dynamic intellectual property protection for reconfigurable devices. In *Proceedings of the International Conference on Field-Programmable Technology (ICFPT'07)*. IEEE, 169–176.
- Scott Hauck and Andre DeHon. 2010. *Reconfigurable Computing: The Theory and Practice of FPGA-based Computation*. Morgan Kaufmann.
- Yohei Hori, Toshihiro Katashita, Hirofumi Sakane, Toda Kenji, and Akashi Satoh. 2013. Bitstream protection in dynamic partial reconfiguration systems using authenticated encryption. *IEICE Transactions on Information and Systems* 96, 11 (November 2013), 2333–2343.
- Yohei Hori, Toshihiro Katashita, and Akashi Satoh. 2012. Tackling the security issues of FPGA partial reconfiguration with physical unclonable functions. In *Proceedings of Engineering of Reconfigurable Systems and Algorithms (ERSA'12)*. 79–90.
- Hirak Kashyap and Ricardo Chaves. 2014. Secure partial dynamic reconfiguration with unsecured external memory. In *Proceedings of the 2014 24th International Conference on Field Programmable Logic and Applications (FPL'14)*. IEEE, 1–7.
- Tom Kean. 2001. Secure configuration of field programmable gate arrays. In *Field-Programmable Logic and Applications*. Springer, 142–151.
- Krzysztof Kepa, Fearghal Morgan, Krzysztof Kosciuszkiwicz, and Tomasz Surmacz. 2010. SeReCon: A secure reconfiguration controller for self-reconfigurable systems. *International Journal of Critical Computer-Based Systems* 1, 1 (February 2010), 86–103.
- Roel Maes, Dries Schellekens, and Ingrid Verbauwhede. 2012. A pay-per-use licensing scheme for hardware IP cores in recent SRAM-based FPGAs. *IEEE Transactions on Information Forensics and Security* 7, 1 (February 2012), 98–108.
- Milind M. Parelkar and Kris Gaj. 2005. Implementation of EAX mode of operation for FPGA bitstream encryption and authentication. In *Proceedings 2005 IEEE International Conference on Field-Programmable Technology*. IEEE, 335–336.
- Pawel Swierczynski, Amir Moradi, David Oswald, and Christof Paar. 2015. Physical security evaluation of the bitstream encryption mechanism of Altera Stratix II and Stratix III FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 7, 4 (January 2015), 34:1–34:23.
- Stephen M. Trimberger and Jason J. Moore. 2014. FPGA security: Motivations, features, and applications. In *Proceedings of the IEEE*, Vol. 102. IEEE, 1248–1265.
- Jo Vliegen, Nele Mentens, Dirk Koch, Dries Schellekens, and Ingrid Verbauwhede. 2015. Practical feasibility evaluation and improvement of a pay-per-use licensing scheme for hardware IP cores in Xilinx FPGAs. *Journal of Cryptographic Engineering* 5, 2 (June 2015), 113–122.
- Jo Vliegen, Nele Mentens, and Ingrid Verbauwhede. 2013. A single-chip solution for the secure remote configuration of FPGAs using bitstream compression. In *Proceedings of the 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig'13)*.
- Jo Vliegen, Nele Mentens, and Ingrid Verbauwhede. 2014. Secure, remote, dynamic reconfiguration of FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 7, 4, Article 35 (December 2014).
- Knut Wold and Chik How Tan. 2008. Analysis and enhancement of random number generator in FPGA based on oscillator rings. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs*. 385–390.
- Thomas Wollinger, Jorge Guajardo, and Christof Paar. 2004. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Transactions on Embedded Computing Systems (TECS)* 3, 3 (August 2004), 534–574.
- Xilinx. 2012. Partial Reconfiguration User Guide - UG702. (July 2012).
- Xilinx. 2013. 7 Series FPGAs Configuration User Guide - UG470. (October 2013).
- Amir Sheikh Zeineddini and Kris Gaj. 2005. Secure partial reconfiguration of FPGAs. In *Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology*. IEEE, 155–162.
- Jiliang Zhang, Yaping Lin, and Gang Qu. 2015. Reconfigurable binding against FPGA replay attacks. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 20, 2 (February 2015), 33.

Received March 2015; revised July 2015; accepted August 2015