Secure Remote Reconfiguration of FPGAs

Nele Mentens[1,2], Jo Vliegen[1,2], An Braeken[3], Abdellah Touhafi[3,4], Karel Wouters[1] and Ingrid Verbauwhede[1]

[1] Katholieke Universiteit Leuven, ESAT, SCD/COSIC
Kasteelpark Arenberg 10, bus 2446, 3001 Leuven, Belgium
{nele.mentens,jo.vliegen,karel.wouters,ingrid.verbauwhede}
@esat.kuleuven.be
[2] Katholieke Hogeschool Limburg, IWT, ACRO/ES&S
Agoralaan Gebouw B bus 3, 3590 Diepenbeek, Belgium
{nele.mentens,jo.vliegen}@iwt.khlim.be
[3] Erasmushogeschool Brussel, IWT
Nijverheidskaai 170, 1070 Anderlecht, Belgium
{an.braeken,abdellah.touhafi}@ehb.be
[4] Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussel, Belgium

Abstract. This paper presents a solution for secure remote reconfiguration of FPGAs. Communicating the bitstream has to be done in a secure manner to prevent an attacker from reading or altering the bitstream. We propose a setup in which the FPGA is the single device in the system's zone-of-trust. The result is an FPGA architecture that is divided into a static and a dynamic region. The static region holds the communication, security and reconfiguration facilities, while the dynamic region contains the targeted application.

Keywords. FPGA, cryptography, security, remote configuration

1 Introduction

Remote reconfigurability is a feature from which FPGAs can benefit through remote upgrades, conditional functionality and a reduction of the time-to-market by selling products with limited functionality.

In this paper we propose an architecture that allows secure remote reconfigurability with a generic architecture based on a low-cost soft-core microprocessor and cryptographic IP cores.

The paper is organized as follows. Section 2 describes the most relevant related work. In Sect. 3, our solution is explained in terms of assumption and goals, protocol and algorithms and the hardware architecture on the FPGA.

2 Related Work

The most recent and most efficient architecture for secure remote reconfiguration of FPGAs, was proposed by Drimer in [1]. His solution consists of a non-volatile

Dagstuhl Seminar Proceedings 10281
Dynamically Reconfigurable Architectures
http://drops.dagstuhl.de/opus/volltexte/2010/2839

memory that stores the received bitstream after the user logic in the FPGA has performed security operations on the bitstream. The result is a solution that ensures bitstream confidentiality and authenticity. It also prevents denial-of-service attacks and guarantees the freshness of the bitstream. Most other proposed solutions in literature show cryptographic weaknesses, as also described in [1].

The difference with our solution is that we assume the FPGA to be the only device inside the system's zone-of-trust. Further, our architecture allows an easy expansion of the cryptographic functionality.

3 Our solution

In this section, we describe our architecture for secure remote reconfiguration of FPGAs. In the first paragraph we list the assumptions and goals that we started from. The second paragraph describes the protocol that meets these goals and the algorithms needed to execute the protocol. The last paragraph of this section describes the implemented FPGA architecture.

Assumptions and Goals We aim at a solution in which the FPGA is the only device inside the system's zone-of-trust. As a consequence, we do not require any secure external memory. We also want our solution to be suitable for many FPGAs in the field being reconfigured by one central reconfiguration unit. Moreover, we do not want to store a list of secret keys, which brings up the need for public key cryptography. Further, we do not want to rely on any trusted third party. Finally, we assume that the updates of the reconfiguration occur only once in a while, which causes the reconfiguration speed to be of minor importance.

Under these assumptions, we want to achieve the following security goals:

- confidentiality of the bitstream
- authenticity of the bitstream
- authenticity of the central reconfiguration unit and the FPGA
- extendability of the cryptographic functionalities

Protocol and Algorithms A standardized protocol that suits all our assumptions and goals, is the Station-To-Station (STS) protocol [2]. This protocol involves hash functions, symmetric key encryption and decryption and the signing and verification of digital signatures. The hash function we implemented is SHA-256 and for symmetric key encryption/decryption we use AES-128. Because we focus on the compactness of the implementation, we use elliptic curve cryptography for the public key operations. This is shown in Table 1.

Architecture Fig. 1 shows the architecture that we implemented on a Virtex-II Pro FPGA. The PicoBlaze is an 8-bit soft-core microprocessor. It is connected through a bus with a communication block that takes care of the internet communication. Although the PicoBlaze consumes very little FPGA resources, the

Table 1. STS protocol using SHA-256 as a hash function and AES-128 for symmetric key encryption and decryption. Signing and verifying is done using elliptic curve cryptography.

CRU		**ES**
key pair (a, A)		key pair (b, B)
generator P		generator P
Cert_A		Cert_B, A
choose k_1		
$Q_1 = k_1 * P$		
	Q_1 $\longrightarrow$	
		choose k_2
		$Q_2 = k_2 * P$
		$K_1 \parallel K_2 = \text{HASH}(k_2 * Q_1)$
		$C = E_{K_1}(S_b(Q_1 \parallel Q_2))$
	$Q_2 \parallel C \parallel B \parallel \text{id} \parallel \text{Cert}_B$ $\longleftarrow$	
$K_1 \parallel K_2 = \text{HASH}(k_1 * Q_2)$		
$D_{K_1}(C)$		
verify $S_b(Q_1 \parallel Q_2)$ using B		
verify Cert_B using A		
$F = E_{K_1}(S_a(Q_1 \parallel Q_2))$		
	$\text{Cert}_A \parallel F$ $\longrightarrow$	
		$D_{K_1}(F)$
		verify $S_a(Q_1 \parallel Q_2)$ using A
	$E_{K_1}(\text{HMAC}(K_2,\text{msg}) \parallel \text{msg})$ $\longleftrightarrow$	

architecture could be optimized by replacing it by an FSM. The block RAM is used to store data and instructions and the cryptographic block incorporates the implementation of the algorithms described in the previous paragraph. The ICAP (Internal Configuration Access Port) is a component that handles the reconfiguration of the FPGA.

The cryptographic block represents a total of 4760 slices, 19 (16 kbit) BRAM blocks and 7 (18x18) multipliers. More details on the cryptographic part of our solution can be found in [3].

4 Conclusions and Open Problems

We presented an generic architecture for secure remote reconfiguration of FPGAs. We implemented the standardized Station-To-Station (STS) protocol with a focus on compactness.

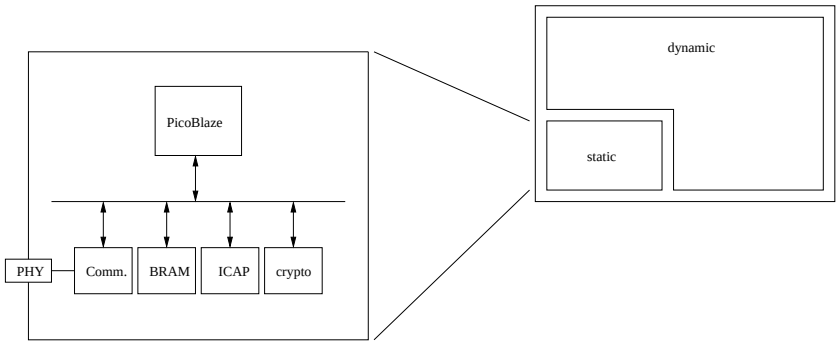

Fig. 1. Architecture in which the targeted application is implemented in the dynamic region.

The problems to be solved in our architecture are the fact that the bitstreams are too large to be stored in the clear inside the FPGA, which could (partially) be solved by using bitstream compression or a change of the protocol.

Further, to counter the fact that most standardized cryptographic components are not provably secure, a good solution would be to make these components remotely reconfigurable too. This would, however, require a change of the entire architecture.

Acknowledgements

This work was funded by the IWT-Tetra project "STRES: Secure Techniques for Remote configuration of Embedded Systems".

References

1. Drimer, S.: Security for Volatile FPGAs. PhD thesis, University of Cambridge, Computer Laboratory (2009)
2. Diffie, W., van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Design Codes Cryptography **2** (1992) 107–125
3. Vliegen, J., Mentens, N., Genoe, J., Braeken, A., Kubera, S., Touhafi, A., Verbauwhede, I.: A compact FPGA-based architecture for elliptic curve cryptography over prime fields. In: ASAP. (2010) 313–316