Forward-Secure Content Distribution to Reconfigurable Hardware

David Champagne[1], Reouven Elbaz[1,2], Catherine Gebotys[2], Lionel Torres[3] and Ruby B. Lee[1]
[1]*Princeton University USA, {dav, relbaz, rblee}@princeton.edu*
[2]*University of Waterloo, Canada, {reouven, cgebotys}@uwaterloo.ca*
[3]*LIRMM University of Montpellier 2/CNRS, France, lionel.torres@lirmm.fr*

Abstract

Confidentiality and integrity of bitstreams and authenticated update of FPGA configurations are fundamental to trusted computing on reconfigurable technology. In this paper, we propose to provide these security services for digital content broadcast to FPGA-based devices. To that end, we introduce a new property we call forward security, which ensures that broadcast content can only be accessed by FPGA chips configured with the latest bitstream version. We describe the hardware architecture and communication protocols supporting this security property, and we evaluate the associated cost.

1. Introduction

Field-Programmable Gate Array (FPGA) technology enables updates to computing hardware, to extend functionality or to fix design flaws such as security vulnerabilities. New hardware configurations are sent to platforms in the form of a configuration bitstream, often via a communication channel established with a remote server.

In addition to bitstream updates, certain FPGA-based devices we call *receivers* also receive digital content to be processed by the reconfigurable hardware. This content can consist of multimedia data (e.g., an audio-video stream for a PayTV decoder, video disc files for a DVD player) or of updates to software running on the reconfigurable hardware (e.g. code update for a car's Electronic Control Unit, ECU).

A desirable security goal in this setting is to ensure that content only be accessible to devices with the latest hardware configuration. This property we call *forward security*[1] guarantees that a hardware vulnerability allowing a violation of the system's security policy (e.g. leakage of session keys or confidential content) becomes benign as soon as the server sends out a bitstream update patching the vulnerability. This prevents devices that do not apply an earlier security patch from still being able to view content sent later. In addition, forward security precludes obsolete hardware from processing new digital content.

Achieving forward security requires the authentication and encryption of both the bitstream and the digital content. Existing techniques providing confidentiality and integrity for bitstream updates or content transmissions assign each device a private key from an asymmetric key pair or a symmetric key shared with the server [7, 9, 1, 4, 6, 14]. However, these techniques do not scale up to systems comprising millions of FPGAs since they require one secure communication channel per device. Moreover, they cannot provide forward security in such systems: the server either is unable to determine the version of a device's bitstream or it obtains version information by requesting an acknowledgment from each receiver, an approach that is inapplicable to a broadcast setup.

Broadcast Encryption (BE) is a cryptographic technique allowing a server to send confidential messages to a large number of devices—the broadcast group—over a single broadcast channel by assigning each device a small number of keys [8]. BE has been applied to receiver devices such as DVD players, PayTV systems and car ECUs, but the implementations did not account for the specificity of FPGA-based receivers—namely, their dynamic upgradeability.

For a given broadcast group, the secret key protecting data remains the same across transmissions. As a result, devices that do not apply a given bitstream update can still decrypt content that is broadcast after the update is sent—i.e. existing implementations of BE for receiver devices do not provide forward security.

Moreover, the static logic of several types of FPGAs has very limited non-volatile memory resources, typically a single register storing a secret key. Although BE uses only a few keys per device, it may still require more keys than can be stored at an acceptable cost in the static logic portion of these FPGA chips.

[1] Usage of the term forward security in past work differs; our definition follows the suggestions in [3].

978-0-7695-3474-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ReConFig.2008.51

In this paper, we present an architecture enabling forward-secure broadcast distribution of digital contents to FPGA-based receiver devices. We do so by introducing a *transmission key* in BE protocols in order to cryptographically bind broadcast content to the identity of the latest hardware configuration. We also extend FPGA hardware and adapt the BE protocols to ensure confidentiality and integrity of bitstream updates over a broadcast channel. Our solution is scalable and can be applied to most FPGAs since it maintains a small, constant non-volatile memory footprint by providing secure off-chip storage for BE keys.

Section 2 presents an overview of broadcast encryption as well as past work in FPGA security. Section 3 presents our threat model, while Section 4 introduces the architecture we propose to provide forward-secure content distribution to FPGA devices. Section 5 analyzes the security and implementation cost of our approach. Section 6 concludes.

2. Related Work

2.1. Broadcast Encryption

Introduced in [8], Broadcast Encryption (BE) is a cryptographic technique allowing a server we call the Broadcaster to send, over a broadcast channel, confidential messages to an arbitrary subset of privileged users *PS*, chosen from the set *U* containing all users. From one transmission to the next, the Broadcaster may change the composition of *PS*. For example, users may be removed from *PS* if they stop paying for their subscription to the broadcast service.

During an initialization phase, the Broadcaster assigns user devices a certain amount of secret key material—the *BE keys*—to be used in protecting the confidentiality of later broadcasts. There are many ways to generate the BE keys and use them during broadcasts, yielding a large solution space [8, 10].

In this paper, we use the Complete Subtree Method presented in [8], wherein the Broadcaster generates a balanced binary tree *T* of BE keys with each of its *n* leaf keys corresponding to a specific user in *U* (assuming $n = |U|$ is a power of 2). In general, each key k_i in *T* corresponds to the subset of users (leaves) spanned by the balanced subtree with root k_i. During initialization, a user device is assigned the $\log_2(n)+1$ keys that correspond to the subsets it is a member of, i.e. the keys on the path from the user's leaf to the root (e.g. see assignment of keys to u_3 in Fig. 1).

To transmit a confidential message *M* to the users in *PS*, the Broadcaster first encrypts *M* with a randomly generated session key *SK*. Let *R* be the set of revoked users ($R = U \backslash PS$) and *T'* be the graph formed by removing from *T* the keys assigned to users in *R*. *T'* is a collection of *m* subtrees of *T*; let k_{x1} *to* k_{xm} be the keys in *T* that are the roots of the subtrees in *T'* (k_3, k_4 and k_{11} in Fig. 1, where $|R| = 1$). By construction, each device in *PS* owns at least one of the k_{xi}'s, whereas none of the devices in *R* own any of the k_{xi}'s.

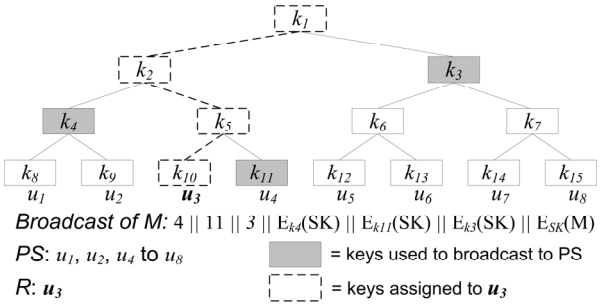

Fig. 1: Example BE key tree for users u_1 to u_8, keys k_1 to k_{15}

The Broadcaster then broadcasts $x_1 \| x_2 \| ... \| x_m \| b_1 \| b_2 \| ... \| b_m \| C$, where x_i is an index referring to k_{xi}, $C = E_{SK}(M)$ and b_i is *SK* encrypted under k_{xi}. $\|$ denotes concatenation and E_K encryption under K. A user receiving the broadcast can decrypt *M* only if it is in *PS*, i.e. if it owns a key in k_{x1} to k_{xm}. It then uses the key k_{xj} it owns to decrypt the corresponding b_j, thus obtaining the session key *SK* used to decrypt *C*.

2.2. Applications of Broadcast Encryption

The latest High-Definition Digital Video Disc formats, HD-DVD and Blu-Ray protect the confidentiality of media content with the Advanced Access Content System standard [11], which is based on the Subset Difference Tree BE scheme presented in [13]. Various PayTV schemes [15, 12] have been designed to leverage the properties of BE in order to broadcast media streams to the privileged set of paying subscribers. BE was also applied in the automotive industry to remotely update software on the multiple ECUs populating a car's computing system [2].

In all cases, however, the proposed implementations of broadcast encryption do not provide the forward security property we are trying to achieve in this paper. Indeed, these schemes do not allow binding content to the latest hardware version of an upgradeable receiver.

2.3. FPGA Security

FPGA security is an active field of research and many efforts are aimed at providing a trustworthy configuration process through bitstream authentication and encryption [1, 4, 6]. In all cases, each device is initialized with its own set of secure communication keys, so broadcasting an update is inefficient: it requires sending over the broadcast channel one encrypted bitstream and MAC per device since the

encryption and MAC keys are different for each device.

[4] has a versioning mechanism, but requires individual acknowledgments from each device to confirm a given update was successfully applied. Since two-way communication is not possible in a typical broadcast setting, this scheme cannot ensure that broadcast content is only accessed by updated devices.

This requirement for two-way communication is also found in approaches to FPGA security based on the Trusted Platform Module (TPM) [14, 7] and device authentication through public key cryptography [9], making those schemes inapplicable to forward-secure content distribution.

3. Threat Model

In this paper, the hardware and software on the Broadcaster's side are considered as trusted and so are the FPGA chips within the user devices. Side channel attacks are considered out-of-scope for this paper.

We consider users as potential adversaries that may attempt to extract secret keys handled by their devices, making it possible to produce clones. This key extraction can occur either through software or physical attacks against the receiver. In a physical attack, the user may, for example, replace a flash memory chip with a malicious one or enter in direct contact with the platform's external buses (i.e. buses outside the FPGA chip) in order to observe or tamper with bus data and control signals. This paper does not, however, consider invasive attacks—e.g. chip peeling—on the FPGA.

4. Architecture and Broadcast Protocols

This section details the approach we propose to provide forward-secure content distribution to FPGA-based receivers. We first give an overview of the proposed scheme, present the receiver's hardware and describe initialization procedures. We then present our secure bitstream loading procedure and the protocols at the core of our approach—i.e. the configuration update and content transmission protocols.

4.1. Overview

Our approach consists in modifying the Complete Subtree protocol for content transmission such that only devices with the latest hardware configuration can access contents being broadcast. The protocols proposed also guarantee that for bitstream updates, any device in the privileged set, regardless of its current configuration, can decrypt and apply a hardware update. At all times, we restrict access to BE keys to the (trusted) static FPGA logic to ensure configurations with security vulnerabilities cannot leak the keys before they are overwritten by an update.

For content transmissions, the Broadcaster encrypts the session key with new *transmission keys*, derived from a BE key and a Configuration IDentifier (CID) identifying the hardware configuration for which the content is intended. The static logic of privileged receivers uses BE keys and the CID of the current configuration to re-generate a transmission key, which it then makes available to the user logic. Hence, the user logic only has access to the correct transmission key (i.e. can decrypt content) when configured with the latest bitstream. Table 1 summarizes the usage of keys.

Table 1: Security Function of Keys

Type of Key	Security Function
BE Key	Retrieve session key Generate transmission key
Session	Protect bitstream confidentiality and integrity
Transmission	Protect digital content confidentiality and integrity

4.2. FPGA Hardware

Figure 2 presents our architecture: a flash memory chip, its controller and an FPGA chip. The three main new components are a register file containing the extra registers needed, control logic to run the protocols and a new User-to-Static-Logic (USL) interface allowing communications between the static logic and reconfigurable hardware. We assume the crypto engine contains hardware for encryption and MAC computation; it is already present in chips providing bitstream encryption and authentication (e.g. [1]).

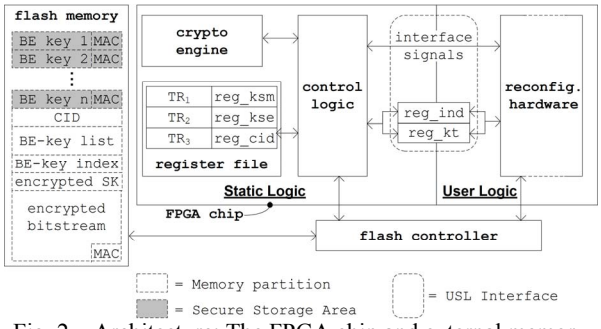

Fig. 2 – Architecture: The FPGA chip and external memory

Key Derivation. The protocols we present below derive cryptographic keys from a common master key. This derivation is done with a secure Key Derivation Function *KDF* based on the crypto engine. *KDF* could for example use the HMAC construction as in the Transport Layer Security (TLS) protocol [5]. In any case, it takes two arguments as its inputs, a public value and a secret master key, and outputs a key whose secrecy is based on the master key's entropy. Table 2 shows the constants and keys used in our protocols.

MAC_K denotes the function computing Message Authentication Codes, keyed with key K.

Table 2: Symbols used in our BE protocols

Symbol	Semantics	Value
CID	Configuration identifier	unique to each bitstream
C_E	Encryption key derivation constant	fixed pre-determined value
C_M	MAC key derivation constant	fixed pre-determined value
BK	A BE key from the BE key tree	generated by broadcaster during init
SK	Session key	generated on-the-fly by broadcaster
TK	Transmission key	TK = KDF(CID, BK)
K_E	Encryption key	K_E = KDF(C_E, SK)
K_M	MAC key	K_M = KDF(C_M, SK)
$^{MSG\_BIT\_}_{UPDATE}$	Bitstream update message header	fixed pre-determined value
$^{MSG\_}_{CONTENT}$	Content transmission header	fixed pre-determined value
K_{SE}	Secure storage encryption key	generated by broadcaster during init
K_{SM}	Secure storage MAC key	generated by broadcaster during init

FPGA-Rooted Secure Storage. With only two non-volatile registers (`reg_ksm` and `reg_kse`) containing an encryption key K_{SE} and a MAC key K_{SM}, the FPGA can protect the confidentiality and integrity of a large number of BE keys. Each of these keys is encrypted and MACed individually and both the ciphertext and the MAC are sent to the off-chip secure storage area (see Fig. 2). For clarity, our protocols only mention reading a BE key from flash; they do not describe its decryption and integrity checking. Note that the BE keys are stored once and never modified, hence they do not need to be protected against replay.

4.3. Initialization Phase

Broadcaster Initialization. Before it deploys devices in the field, the Broadcaster first generates the tree of $2N$–1 BE keys necessary to run a broadcast encryption scheme with its N devices. It then creates N sets of BE keys according to the method in Section 2.1. It also creates a revocation list RL—initially empty—which is to contain the list of devices excluded from the set PS of privileged users. It generates a session key SK which it encrypts with the root K_R of the BE key tree to produce ciphertext C_{SK}. The Broadcaster then encrypts and MACs the first valid bitstream (B_0, with configuration ID CID_0) with keys K_E and K_M derived from SK as in Table 2. Encryption yields ciphertext $C_0=E_{KE}(B_0)$ and MAC M_0 equals $MAC_{KM}(C_0, CID_0)$.

Device Initialization. To initialize a device, the Broadcaster injects it with CID_0, its secure storage keys and the set of BE keys corresponding to the device. We assume this one-time procedure is carried out in a trusted location. The Broadcaster first feeds the static logic of the FPGA K_{SE} and K_{SM} which the FPGA stores in `reg_kse` and `reg_ksm`. As the device is fed the broadcast keys, its static logic encrypts (with K_{SE}) and MACs (with K_{SM}) each of these keys individually and stores the keys and MACs in flash memory for future use. The Broadcaster then sends CID_0 which the device writes in `reg_cid`. Finally, the Broadcaster uploads CID_0, C_0, M_0 and C_{SK} in the device's flash, along with the list of keys owned by the device and the index of K_R (BE-key index list and BE-key index in Fig. 2). The FPGA determines which keys it owns by looking up the BE-key index list, while BE-key index indicates the specific key to use for decryption of C_{SK}. From this point on, the device can only boot up using bitstreams encrypted and MACed by the Broadcaster.

4.4. Broadcast Encryption Protocols

In this section, protocol steps are annotated with letters specifying the entity executing the step (B for Broadcaster, DS for Device's Static logic and DU for Device's User logic) and a sequence number.

Configuration Update Protocol. The Broadcaster initiates this protocol to send out a new version of the receiver hardware. The objective may be to patch security vulnerabilities in the previous version, modify the algorithms used for processing the broadcast data or simply to enhance receiver hardware with more features. In all cases, contents broadcast after the update must be accessible only to updated devices. Otherwise, the contents could be misinterpreted by legacy hardware or confidential information it contains could be leaked by vulnerable devices.

The Broadcaster proceeds as follows (where the new bitstream has configuration ID CID_X):

B1- From RL, determine the set PS of users that are to receive the update
B2- Create list L of n indices referring to the n BE keys $BK_1, BK_2, ..., BK_n$ covering PS
B3- Generate a session key SK
B4- From SK, compute an encryption key K_E and a MAC key K_M (see Table 2)
B5- Encrypt the plaintext P_X of the CID_X bitstream to obtain ciphertext $C_X=E_{KE}(P_X)$
B6- Compute MAC $M_X=MAC_{KM}(C_X, CID_X)$ binding the ciphertext to its CID
B7- Encrypt SK to obtain ciphertexts C_1 to C_n, where $C_i=E_{BKi}(SK)$
B8- Send message M over broadcast channel: $M=MSG\_BIT\_UPDATE \| L \| CID_X \| C_1 \| ... \| C_n \| C_X \| M_X$

Upon detecting an update message M, the user logic of a device carries out the following protocol steps:

DU1- Check if it owns a BE key in L by looking up BE key index list in flash. If yes, go to DU2; if no, ignore M.
DU2- Store in device flash CID_X, C_X, M_X, the index ind of the BE key owned by the device and C_{ind}, the corresponding SK ciphertext
DU3- Request reboot via USL to invoke bitstream loading procedure described next in order to authenticate & decrypt updated bitstream.

Bitstream Loading Procedure. Upon power-up, the static logic of the FPGA performs the following steps:

DS1- Read CID_X in reg_cid and *ind* in reg_ind
DS2- Read the BE key BK_{ind} in TR_1
DS3- Read C_{ind} into temporary register TR_2
DS4- Decrypt C_{ind} to TR_2 using BK_{ind} to get SK
DS5- Compute into TR_3 the encryption key $K_E'=KDF(C_E, [TR_2])$, where $[TR_X]$ means contents of TR_X
DS6- Compute into TR_1 $K_M'=KDF(C_M, [TR_2])$
DS7- Read in C_X and M_X and check the integrity of C_X by re-computing a MAC M_X' over C_X using K_M' and [reg_cid]
DS8- If $M_X'=M_X$, go to DS9; otherwise, raise an error and apply pre-determined policy
DS9- Decrypt C_X using $[TR_3]$ to obtain P_X and configure the user logic with P_X

Content Transmission Protocol. At the beginning of each transmission of plaintext content *V*, assuming the currently valid hardware has configuration ID CID_X, the Broadcaster performs following steps:

B9- Execute configuration update steps B1 to B6 with V rather than P_X
B10- For $BK_1, BK_2, \ldots, BK_n$, compute a transmission key K_{Ti} using CID_X: $K_{Ti} = KDF(CID_X, BK_i)$
B11- Encrypt the session key SK to obtain ciphertexts C_{T1} to C_{Tn}, where $C_{Ti} = E_{KTi}(SK)$
B12- Send message M over the broadcast channel: $M = MSG\_CONTENT \| L \| CID_X \| C_{T1} \| \ldots \| C_{Tn} \| C_X \| M_X$

Upon detecting a content message, the user logic i) checks that the platforms owns a BE-key in *L* by looking up its BE-key index list ii) requests a transmission key by feeding the static logic, via reg_ind, the index ind referencing the BE key owned by the device. The static logic then computes the new transmission key as follows:

DS10- Read into temporary register TR_1 the BE key BK_{ind} corresponding to index [reg_ind]
DS11- Compute KDF([reg_cid], $[TR_1]$) and store the resulting transmission key K_T in reg_kt
DS12- Notify user logic via USL interface that new transmission key is available

With K_T, the user logic can decrypt the session key and from it, derive the keys enabling decryption and authentication of the content message:

DU4- Decrypt C_{Tind} with [reg_kt] to get SK
DU5- Compute $K_E'=KDF(C_E,SK)$ and $K_M'=KDF(C_M,SK)$
DU6- Check the integrity of C_X by recomputing a MAC M_X' over C_X using K_M' and CID_X
DU7- If $M_X'=M_X$, go to DU8; otherwise, ignore M
DU8- Decrypt C_X using K_E' to obtain the P_X content

Note that the DU8 decryption yields unintelligible data when the device bitstream is outdated.

5. Evaluation

5.1. Cost Evaluation

Area. SRAM FPGAs providing bitstream confidentiality and integrity already have crypto engines in the static logic [16, 17, 18]. Hence, our architecture only requires extra control logic (including the new USL interface) and six new registers (two of them non-volatile) to support forward secure-content distribution. The area of these additions is likely to be small in comparison to that of the existing static logic.

Performance. Latencies related to encryption and authentication of bitstreams and content transmissions already apply to existing secure FPGA designs. The only performance hit incurred by our architecture consists in access latencies to the secure storage and computation of transmission keys by the static logic. Both operations are performed only once per broadcast so their relative performance cost should be acceptable when compared to the latencies associated with cryptographic processing of the broadcast content.

5.2. Security Evaluation

Resistance to Key Extraction. An important security objective for the FPGA-based receivers we propose is to prevent extraction of key material that could lead to cloning of devices. We have four types of secret keys.

The *Secure Storage Keys* (K_{SE} and K_{SM}) are unique to each FPGA; they are generated and stored on-chip by the Broadcaster during the initialization procedure. They are only used by the control logic to encrypt and decrypt BE keys and compute MACs: they never leave the static part of the chip. Since this static logic is within the trusted FPGA chip, an attacker cannot spy on or corrupt its operations in order to reveal the secure storage keys, i.e. K_{SE} and K_{SM} cannot be extracted.

The security of the *BE keys* depends on the security of K_{SE} and K_{SM}, as the former are encrypted and MACed by the latter when stored off-chip. Since K_{SE} and K_{SM} themselves cannot be extracted, the only way to obtain BE key bits is to infer information about a BE key by observing the value of a transmission key (e.g. if a vulnerable configuration leaks a transmission key). This is computationally infeasible, as we assume *KDF*, used to derive a transmission key from a BE key, is cryptographically secure—i.e. an attacker cannot obtain information about *KDF*'s pre-image by looking at its output. As a result, the BE keys are protected from key extraction attacks.

The leaking of a *Transmission Key* by vulnerable user logic can only lead to benign cloning of devices. Indeed, the value of all transmission keys changes as soon as bitstream update is sent out to fix the vulnerability. Any clone made prior to the update and containing solely a transmission key becomes obsolete as soon as the update is sent: without BE keys, it is unable to derive a new transmission key allowing decoding of post-update content transmissions.

The Broadcaster generates a new *Session Key* on every transmission. A clone created with a leaked session key is thus incapacitated as soon as that transmission is over.

USL Interface. The insertion of this interface between the user and static logic does not represent a new attack vector. It only allows the user logic to read a transmission key or request a bitstream update from the static logic. Leaking the key leads to benign device clones in the worst case, while a spurious bitstream update is detected when the static logic authenticates it.

Forward Security Property. In order to provide forward security, our architecture must ensure that contents broadcast at time *t* can only be decrypted by devices with the latest hardware configuration at time *t*. For this property to hold, the following two conditions must be respected: 1) the FPGA should only make the latest transmission key available to the latest hardware; 2) the BE keys, used to derive transmission keys, should be inaccessible to attackers at all times.

As we have just shown above, the second condition holds since the BE keys are protected by the FPGA-rooted secure storage. To fulfill the first condition, we introduced the concept of a Configuration IDentifier (CID) used by the static logic as an input to *KDF* in deriving the transmission keys.

By design, the trusted static logic uses the CID of the last successfully applied hardware configuration update (authenticated by the static logic). Thus, the transmission key made available to the user logic can only decrypt current broadcasts if the reconfigurable hardware was configured with the latest bitstream. As a result, an attacker configuring an FPGA with an older bitstream can only achieve denial of service since doing so prevents decryption of new digital content.

6. Conclusion

Confidentiality and integrity of an FPGA chip's hardware configuration are fundamental to trusted computing on reconfigurable technology. We showed the forward security property defined in this paper is also essential for the secure broadcast of digital content to FPGA-based devices. To provide this novel security property in reconfigurable technology, we presented new protocols with supporting FPGA hardware. Our architecture allows for broadcast update of FPGA configurations and cryptographic binding of broadcast digital content to the latest bitstream version. Our solution is low-cost and scalable, as it only require a small and constant amount of non-volatile memory in the static logic.

Acknowledgments

The authors wish to thank Benoît Badrignans and Cédric Lauradoux for their valuable comments on the technical aspects of this paper.

References

[1] Actel, 2008, ProASIC®3 Handbook, available at: http://www.actel.com/documents/PA3_HB.pdf

[2] A.H. Adelsbach and A-R.U. Sadeghi, "Secure Software Delivery and Installation in Embedded Systems," in Proc. of Information Security Practice and Experience, LNCS 3439, pp. 255–267, 2005.

[3] R. Anderson, "Two Remarks on Public-Key Cryptology," Manuscript, 2000, and Invited Lecture at the Computer and Communications Security Conference, April 1997.

[4] B. Badrignans, R. Elbaz and L. Torres, "Secure FPGA configuration technique preventing system downgrade", In Proc. of the 18th IEEE International Conference on Field Programmable Logic and Applications (FPL), 2008

[5] T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1.", April 2006.

[6] S. Drimer, 2008, Volatile FPGA design security – a survey, Computer Laboratory, University of Cambridge, available at: www.cl.cam.ac.uk/~sd410/papers

[7] T. Eisenbarth, T. Güneysu, C. Paar, A. Sadeghi, D. Schellekens and M. Wolf, "Reconfigurable trusted computing in hardware," In Proc. of the ACM Workshop on Scalable Trusted Computing (STC'07), pp. 15-20, 2007.

[8] A. Fiat and M. Naor, "Broadcast encryption," in Advances in Cryptology, LNCS 773, pp. 480–491, 1994.

[9] T. Guneysu, B. Moller and C. Paar, "Dynamic Intellectual Property Protection for Reconfigurable Devices", In Proc. of the Int'l Conf. on Field-Programmable Technology (ICFPT'07), pp. 169-176, 2007.

[10] Jeremy Horwitz, "A Survey of Broadcast Encryption", http://math.scu.edu/~jhorwitz/pubs, 2003.

[11] Intel et al., "Advanced Access Content System (AACS) Specification," 2006, http://www.AACSla.com

[12] Y. Mu and V. Varadharajan, "Robust and secure broadcasting," In Proc. INDOCRYPT 2001, LNCS 2247, Springer Verlag, pp. 223-231, 2001.

[13] D. Naor., M. Naor, J. Lotspiech, Revocation and Tracing Schemes for Stateless Receivers. February, 2001.

[14] D. Schellekens, T. Tuyls and B. Preneel, "Embedded Trusted Computing with Authenticated Non-Volatile Memory", In Proc. of TRUST 2008, LNCS 4968, Springer-Verlag, 2008.

[15] A. Wool, "Key management for encrypted broadcast," ACM Trans. Inform. Syst. Security, vol. 3, no. 2, pp. 107–134, 2000.

[16] A. Lesea, IP security in FPGA, white paper Virtex-4 and Virtex-5 Devices, 2007, available at: http://www.xilinx.com/support/documentation/white_papers/wp261.pdf

[17] Design Security in Stratix III Devices, white paper ALTERA, 2006, available at: www.altera.com/literature/wp/wp-01010.pdf.

[18] LatticeECP2/M Family Data Sheet, 2008, available at: http://www.latticesemi.com/documents/DS1006.pdf