Title	A pragmatic per-device licensing scheme for hardware IP cores on SRAM-based FPGAs
Author(s)	Zhang, Li; Chang, Chip Hong
Citation	Zhang, L., & Chang, C.-H. (2014). A pragmatic per-device licensing scheme for hardware IP cores on SRAM-based FPGAs. IEEE transactions on information forensics and security, 9(11), 1893-1905.
Date	2014
URL	http://hdl.handle.net/10220/25025
Rights	© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [http://dx.doi.org/10.1109/TIFS.2014.2355043].

A Pragmatic Per-device Licensing Scheme for Hardware IP Cores on SRAM based FPGAs

Li Zhang, *Student Member*, *IEEE* and Chip-Hong Chang, *Senior Member*, *IEEE*

***Abstract*—The modus operandi of upfront IP licensing model is impractical for the developers of small to medium volume of FPGA-based applications. FPGA IP market is in dire need for a more competitive and secure IP licensing scheme to flourish. In this paper, a pragmatic security protocol that could support the licensing of IP cores on a per-device basis without the need for a contractual agreement of an external trusted third party, large bandwidth and complicated flow of communications is proposed. Besides assuring that the incentives of all stakeholders involved in the FPGA IP core trading are not scarified if not enhanced, the proposed protocol guarantees total secrecy and integrity of licensed IP cores based on available primitives of contemporary SRAM based FPGA devices. It also prohibits the implementation of protected IP cores on unscreened excess devices and counterfeit chips sold in the gray market. The adoption of IP fingerprinting further deters the IP licensees from abusing the IP cores. Using the self-reconfiguration feature of FPGAs today, the resources consumed by the control module used for the implementation of protected IP cores on the authorized device are temporary and marginal.**

***Index Terms*—FPGA Hardware IP, IP licensing, IP protection, Hardware security, Reconfigurable architecture**

I. Introduction

FIELD-PROGRAMMABLE GATE ARRAY (FPGA) was first introduced in 1980s. Its greatest competitive advantage over application specific integrated circuit (ASIC) is its agility to change or update the implemented functionality and the much lower non-recurring engineering (NRE) cost. FPGAs have been widely used in telecommunications, networking, consumer, automotive and industrial applications, etc. [1]. Their ever increasing integration density has allowed the implementable design to evolve from simple glue logic at the early stage to sophisticated complete systems today. Even then, designing a complete system on FPGA from scratch to meet the reduced time-to-market window has been a daunting task. To shorten the design turnaround time, reuse-based design methodology has been adopted, wherein reusable design blocks, also known as intellectual property (IP) cores, are either fetched from internal repository or purchased from an external party. Such modular design methodology has been well supported by mainstream FPGA development tools to reduce the designers' workload. Adoption of standardized IP interface, e.g., AXI4 interconnect [2], has paved the way for easy adoption of third party IP cores, and boosted the business of IP core licensing.

However, two major concerns, if left unattended, could hinder the prosperity of FPGA IP market. The first concern is the security threat of IP core trading. The IP core venders have invested huge sum of money, time and effort in the IP core development. The returns of their investment are obtained by granting the usage of these IP cores to other parties through IP licensing. However, such revenues can be easily extorted as the bitstreams of native or inadequately protected IP cores are vulnerable to piracy, misappropriation and reverse engineering by malicious licensees or adversaries. Consequently, some core developers are hesitant to launch their precious IP cores into the FPGA IP market.

The second concern is the compatibility of IP licensing model for FPGA. Prevailing IP licensing model is based on upfront licensing like Xilinx's Sign Once agreement [3], in which the system developer pays the core vendor a lump sum of money for the unlimited use of an IP core. While an expensive blanket license of unrestricted IP usage makes sense for ASIC products as the costly license fee is amortized over the mass production of ASIC chips, it is out of the budget of most FPGA-based system developers who target applications of small to medium volume. Hence, the IP cores for FPGA-based applications must be strategically priced such that the unit cost of the end product remains competitive.

For FPGA IP market to thrive, a secure transaction environment must be established with an attractive licensing model for products of low to medium volume. The protocol of such an environment must assure the core vendors that their IP cores cannot be abused and no design information of their IP cores will be divulged. At the same time, the system developers must also be assured that the received IP cores are authentic and have not been compromised.

Legal measures such as patent protection act and copyright law are feeble in fostering secure IP transactions and need to be fortified by technological means. Recent FPGA chips are equipped with on-chip non-volatile key storage and hardwired decryption engine to support the implementation of encrypted bitstreams. These cryptographic features enable secure

Manuscript received August 31, 2013, revised January 28, May 4, and August 28, 2014.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, 50, Nanyang Avenue, Singapore 639798 (e-mail: lzhang2@e.ntu.edu.sg, echchang@ntu.edu.sg).

configuration of FPGA bitstream to effectively prevent eavesdropping and deter the original design from being stolen. As the entire design is protected as a monolithic IP at the system level, it aids only the digital right management for upfront licensing model but does not permit the system developer to independently authenticate different third party IP cores at the core level. To prevent the misuse of IP cores by malicious licensees in a limited usage licensing model, a potent measure capable of providing core-level IP protection is required. In this paper, a per-device licensing scheme is proposed to achieve the goal. The IP instance for each system developer is encrypted, and its decryption key can only be generated on specific chips with the aid of the corresponding license tokens. This guarantees that the licensed IP core is only permitted to be implemented in those contracted devices. The encrypted IP core and all the security sensitive data are communicated in the encrypted authenticated form so that only unimpaired IP core is allowed to be implemented on the target devices. To deter the system developer from misappropriating or counterfeiting the IP core, his licensed IP instance can be tracked by his unique fingerprint embedded in it. By endowing the FV with the capacity to enroll the IP cores and bind the notarization of their core installation modules with the authenticity of the chips, our scheme has an added advantage that the use of overproduced and counterfeit FPGA chips is inhibited. The proposed scheme can be realized without the requirement for an external trusted third party (TTP) and is directly applicable to IP cores designed for commercially available FPGA devices, which makes the business model more attractive. By exploiting the self-reconfiguring feature of modern FPGAs, the fabrics used to enroll and install the licensed IP cores are marginal and can be released upon the secure configuration of IP cores onto the target FPGA. The released fabrics can then be used for the implementation of other free or in-house functional blocks.

The rest of this paper is organized as follows: Section II presents the background information about the FPGA IP market, including the key stakeholders, common IP core attacks, existing protection proposals and desiderata for an effective scheme. The proposed scheme is described in Section III, followed by its evaluation in Section IV where the security, enhancements, practicality and implementation overhead are discussed. Finally the paper is concluded in Section V.

II. Background of FPGA IP Market

A. Stakeholders of FPGA IP market

Identifying the roles and capabilities of the principal stakeholders involved in the FPGA IP market is a prerequisite for developing a veritably practical and secure IP licensing scheme. The following main parties are involved either directly or indirectly in an FPGA IP transaction.

FPGA vendor (FV) is the party who design and sell FPGA chips. A bulk of the fabric in most chips is uncommitted logic, but it is also in the interest of the FV to include in some of their product families hard-wired functional primitives needed by most customers to boost their sales. With the increasing cost of running a state-of-the-art fabrication facility, it has been common for the FVs to outsource their chip fabrication to external fabs called *hardware manufacturers* (HMs) here. The FV and HM sign a contract for a fixed amount of chips to be produced. However, the cost of excess production is marginal for the HM after the masks for an FPGA have been created. There is a rising concern that the HM may produce excess chips and sell them in the grey market to undermine the FV's market revenue.

C*ore vendors* (CVs) form the third principal of the market. They are mostly design houses specialized in providing innovative solutions to some design problems. Usually CVs target their solutions to specific device families of chosen FVs for optimized circuit performance and sell them as IP cores. When a transaction is committed, these IP cores are usually customized according to the implementation requirements of *system developer* (SD) who is the consumer of FPGA IP market. Using external IP cores and those built internally, the SD implements a complete application on the FPGA chips purchased from the FV. The chips will then be sold as producer goods or end products. As the IP core license fee is the CV's source of revenue but part of the SD's expenditure, an implicit conflict of interest exists between these two parties. The CV wants to protect his revenue stream by preventing their IP cores from being misused, while the SD wants to cut cost and may evade the license fee or amortize the cost by reselling the legally purchased IP cores to other SDs or exchanging with them other IPs. The connections and business interests among these four parties are depicted in Fig. 1. Besides the four main parties described above, there exists an additional party called the *malicious attacker* (MA). The MAs may steal valuable IP cores from the FPGA chips by reverse engineering or eavesdropping on and intercepting the communications between the SD and CV. An MA is not necessarily a stand-alone party. It can also be acted by some SDs.

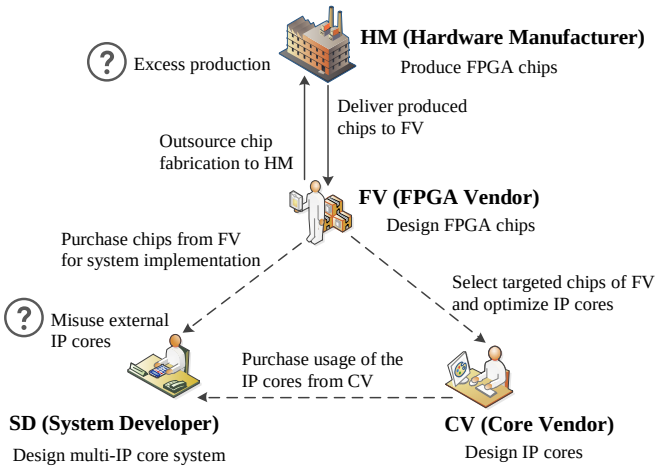


Fig. 1. The four main parties and their interests in FPGA IP market.

Some FVs also provide a portfolio of IP cores to aid the design task of SDs in order to boost the sale of their chips, hence sharing similar risks as the CVs. The FV and CV share a commitment to maximize each other profits through a subtle and indirect supplier-client relationship through the SD. It should be noted that the competition between the FV and CV is benign, yet their interdependency requires a level of trust that is often compromised by the unfair competitive advantage if the

FV is given privileged access to the proprietary design secret of the CV's IP cores implemented by the SD on his FPGAs. For this reason, the environment or scheme that deals with the FPGA IP core transaction should ensure that the IP cores are hard to be snooped or even stolen by the FV besides the SD and MA.

B. Security Concerns

Commonly used FPGA devices can be classified into three main categories according to the type of configuration memory, i.e., antifuse-based, flash-based and SRAM-based. SRAM-based FPGAs have the widest range of applications due to their denser construction, lower cost and better performance than the other two types. On the other hand, the configuration bitstream of the SRAM-based FPGA are also the most vulnerable to attacks. Unlike antifuse or flash-based memory, SRAM memory is volatile. The bitstream needs to be stored in an external non-volatile memory (NVM) and then loaded from the NVM upon power up to configure the FPGA fabrics. As a result, the bitstream can be easily stolen by eavesdropping on the bus between the NVM and SRAM-FPGA. The possible attacks to the poached bitstream, as well as the attacks that can be performed after the bitstream is configured onto the chip, are summarized in Table I.

Table I
Possible attacks to the bitstream of the SRAM-based FPGA

Cloning	The poached bitstream, if in plaintext, can be directly implemented on other FPGA chips.	
Reverse engineering	A plaintext bitstream can be reverse engineered to netlist [4, 5] to expose the valuable design secret.	
Tampering	The bitstream can be modified to leak confidential information or the normal functionality of the circuitry can be deranged.	
Physical attacks	Invasive	Fault attacks: the device is tampered with for a desired fault and then the difference between the correct and faulty outputs is analyzed to extract the sensitive information like secret key.
		Sophisticated tools like mechanical probes and focused ion beam are used to study the inner structure of the chip and learn design information.
	Non-invasive	Side-channel information generated during data processing, such as computation time, consumed power and emitted electromagnetic radiation, is used to infer the design secrets.

Cloning of the unprotected plaintext bitstream can be performed with almost zero cost. Reverse engineering the bitstream enables the attacker to devise competitive IP cores from the extracted proprietary design information at a much lower cost. These attacks immensely harm the benefit of the CV. Due to the high risks and disastrous consequences of malicious tampering, e.g., insertion of hardware Trojan (HT), rigorous integrity check is demanded for the outsourced chips and third-party IP cores for applications used in sectors like military, finance, energy and politics. The licensing protocol cannot assure that the uncommitted FPGA chips sold by the FV are HT-free (such sanity check is assumed to have been carried out by the FV before the chips are shipped to the customers), but it can accommodate measures that will protect and check the integrity of the reconfigurable IP cores before they are allowed to run on the chips.

Even though the bitstream of the IP core is well protected against cloning, reverse engineering and tampering, secrets related to the IP core may be deduced by fault attacks or side-channel attacks. Expensive physical attacks may also be performed to study the internals of the chip and learn about the design information. Due to the complexity and integration density of recent FPGA chips, these physical attacks are not easy to succeed. The major concerns of our protocol design are the low to medium cost attacks, i.e., cloning, reverse engineering, malicious tampering, common side-channel and fault attacks.

C. Existing Protection Proposals

In view of the above security problems for designs implemented on SRAM-based FPGA chips, various protection measures have been proposed.

One straightforward method is to move the NVM into the package of the FPGA chip or even integrate the NVM onto the same die as a part of the FPGA logic [6]. This way, eavesdropping on the bitstream is no longer that easy. However, the former approach improves the situation only marginally as relatively basic tools are enough to remove the packaging material to intercept the inter-die communications. The latter approach is more secure but such integration incurs higher manufacturing cost and limits the size of reconfigurable logic.

The second method is to incorporate an external secure device with each FPGA chip [7, 8]. Both the secure device and the bitstream contain a secret key and a keyed hash function. The design functionality will not be enabled until the implementation platform has been successfully authenticated by the secure device through a challenge-response mechanism. An alternative but similar method is to read the device identity (e.g. device DNA [9]), and process it with a keyed secure hash function and secret key contained in the bitstream to generate a code for comparison with a verification code stored in an external secure NVM. There are two obvious drawbacks for this kind of method. Firstly, additional hardware devices are required, which impose extra cost. Secondly, secret information is stored in the plaintext bitstream accessible by the SD, which is highly insecure.

Another protection method is to add an ancillary hardwired decryption engine and a non-volatile key storage component on chip [10, 11]. With this setup, the bitstream is stored in the external NVM in encrypted form and can only be decrypted by its associated FPGA chip with the correct key stored on chip. This method effectively prevents the bitstream from being cloned or reverse engineered. It is enhanced recently with the authentication feature [12] to deal with possible malicious tampering. However, the limitation of this method, which is also shared by the previous two methods, is that it assumes that the SD has access to the content of the whole system which is protected as a monolithic IP. In other words, this method works well with the up-front licensing model but is incapable of providing the desired core-level protection. The same limitation also exists in schemes like [13-15].

Current core-level protection methods can be divided into detective and preventive methods. FPGA-based IP watermarking [16-18] and fingerprinting techniques [19-21] belong to the detective methods and are helpful in proving the

authorship of IP owners. The deficiency of these techniques is that they just passively confirm the infringement of the suspected chip. In contrast, core-level preventive methods are effective in stopping IP infringement by making any attempt to infringe the IP more difficult and costly.

The first preventive scheme at core level is proposed in [22]. A secret key is stored on each FPGA device and used to generate security tokens by encrypting the user keys. The user keys, security tokens, together with the device ID, are stored in the database of a TTP who also manages the billing of IP cores. In this scheme, the encrypted cores from the CV undergo the decryption and encryption processes within the trusted computer-aided design (CAD) tool and programming software before the final encrypted bitstream, appended with a security token, is loaded onto the designated FPGA chip. As only the designated FPGA chip can extract the correct user key from the token to decrypt the bitstream, this scheme provides the core-level protection.

In general, the higher level the protected IP cores reside, the more parties and design tools are involved, and the more complicated is the task of protecting the IP cores. As pointed out by [23], the most suitable target for core-level FPGA IP protection is the bitstream, which is generated at the last stage of FPGA design and implementation flow. This allows the security boundary to be set at the FPGA chip, which means that the IP cores are to be kept in encrypted form in transit from the CV to the device. The delivery can even be carried out through publicly accessible channels (if authentication is supported) as long as the IP cores can only be decrypted within the designated FPGA chip using the safely stored secret key in the device. In [23], FPGA fabrics are used to implement a key derivation function based on a public-key algorithm to securely transport and install the keys for bitstream decryption. This scheme avoids the burden of public-key functionality by temporarily occupying the reconfigurable logic. A similar scheme was proposed in [24]. It employs symmetric cryptography and an external TTP instead to secure the key transport, and is applicable to commercially available devices. Using symmetric instead of asymmetric cryptography helps to reduce the size of temporarily occupied reconfigurable logic for establishing the IP decryption key.

The main features as well as the unique drawbacks of these three core-level IP protection schemes are summarized in Table II. It should also be noted that pragmatically, the lack of a direct vendor and client relation in the FPGA IP market makes it difficult to find an independent entity that is mutually trustable by all parties involved.

Table II
Features and drawbacks of existing core-level IP protection schemes

Scheme	Feature	Drawback
[22]	Use an external TTP. Store secret information in design and programming tools.	A successful crack to any of the tools will cause a leakage of the IP content.
[23]	Use the FV as the TTP. Use asymmetric crypto to establish IP decryption key.	Need to modify current devices to add special protected registers. FV gains easy access to the IP content.
[24]	Use an external TTP. Use symmetric crypto to establish IP decryption key.	Requires the FV to transfer all his devices to and fro the TTP's secure site to install the unique device key.

D. Desiderata of FPGA IP Licensing Scheme

As remarked in [25], a secure transaction system should be designed such that a potential attack requires expensive equipment, and special skills and knowledge to succeed. This consideration also applies to the core-level IP protection. A protection scheme that is immune to any attacks at all cost is impossible. A practical scheme is one that makes a successful attack too expensive to be compensated by the benefits obtained from the stolen IP cores and at the same time incurs a market-viable implementation cost. Based on the analyses of existing proposals and the discussions of [26], the essential quality attributes are listed in Table III.

The scheme to be proposed in Section III possesses all the above attributes and can be adopted immediately for FPGA IP core licensing on a per-device basis in existing marketplace. This proposition is different from the schemes [27-29] that require a hardware primitive called physical unclonable function (PUF) [30] to generate the unique device identity or secret device key. The latter schemes are not immediately adoptable in the IP market of the SRAM-based FPGA because no SRAM-based FPGA has yet been equipped with the PUF primitive and no FV, including Xilinx and Altera, has announced such a plan. A more practical scheme is that of [31]. However, it is designed specifically for multi-FPGA systems in the automotive market and can implement only one IP core on one chip. In contrast, our scheme is applicable to many different scenarios of device-aware IP core licensing, including multi-core licensing from different CVs and each CV's IP cores are independently protected by the protocol of this scheme.

Table III
Desiderata of core-level protection in FPGA IP licensing

Broad protection	The scheme protects the IP cores well against all the perceivable low to medium cost attacks such as cloning, reverse engineering, malicious tampering and common side channel attacks and fault attacks.
Low cost penalty	The scheme does not increase the complexity or lower the yield of current manufacturing process. Neither does it require new ancillary hardware other than those available in existing chips.
Transparency	No change to current CAD tools is required. The scheme does not impose any difficulty in reconfiguring or updating the chip, nor impacting the chip reliability.
Cryptographically secure	Cryptographic algorithms used in the scheme, if any, are widely accepted as secure. Secret information is well protected and needs not be stored or resided in software tools.

III. Proposed Secure FPGA IP Licensing Scheme

A. Prerequisite and Assumptions

(1) *Employed Device Features*: The existing features of FPGA devices required by the scheme are listed as follows:

- *Device Identifier*: It is a bit string that can uniquely identify the device, denoted as *#ID*. One example is the 57-bit device DNA that is hardwired on some Xilinx chips and can be read from the DNA port [9].
- *On-chip NVM and Decryption Engine*: The on-chip NVM stores the secret key which can only be accessed by the hardwired symmetric key decryption engine. A commonly used standard for the decryption engine is Advanced Encryption Standard (AES) [32].

- *Keyed-Hash Message Authentication Code (HMAC)*: It is used for bitstream authentication, which is enabled in tandem with the encryption/decryption feature for recent FPGA devices like Xilinx Vertex 6 and 7 series [33]. On these devices, HMAC based on SHA-256 algorithm [34] is employed. The encrypted authenticated bitstream[1] protects both the design confidentiality and integrity.
- *Self-reconfiguration*: The fabric can gain access to the configuration controller via internal configuration access port (ICAP) to reconfigure part of its committed logic.

It should be noted that the configuration readback feature supported by most FPGA families, which allows the configured bitstream to be read for debugging purpose, must be disabled. Some readback attacks [35] can deactivate the security bits used for disabling the readback feature. Hardened readback disabling circuitry [33] is used in Xilinx Vertex 6 and 7 devices to ensure that external readback is disabled when an encrypted bitstream is loaded.

(2) *Trust towards the FV*: Our scheme does not require an external TTP. Instead, this role is indirectly played by the FV. There are several advantages in this arrangement. Firstly, there have already been direct interactions between the FV and both the SD and CV. Having the FV as the middleman simplifies the communications among different parties. Secondly, there are strong incentives for all stakeholders involved to have the FV facilitate and manage the communications between them. By offering IP repository services to the CVs, the FV can now provide a more complete portfolio of IP cores to value add to his devices, making them more competitive and marketable. SDs find it convenient to look for a wide range of cores that meet their system requirements without having to deal with many different parties and have greater assurance of the tool support to integrate different cores on the FV's devices. CVs will also benefit immensely from the broad customer base of the FV. Thirdly, the FV is committed to his device and service quality and cannot afford to stain his market reputation and credibility by conducting malicious acts such as colluding with the SD to steal the IP cores.

Even though the FV is trusted to act honestly in discharging this duty, our scheme prevents the FV from accessible to the secret information such as the content of IP core and the encryption key. This is different from [15, 23, 31] where the FV has easy access to those secrets.

(3) *Communication between parties*: Our scheme assumes that the communication between the CV and FV is authenticated and confidential (by using a standard internet security protocol like transport layer security (TLS)). This is easy to be achieved considering the mutually beneficial strategic partnership between the CV and FV. As the SD is the party which may conduct maliciously, the communication between the CV and SD is assumed to be made through an unprotected public channel.

B. Core Installation Module (CIM)

A CIM is used exclusively for the secure installation of the encrypted authenticated bitstream onto the FPGA chip. To ensure the design confidentiality and integrity, the protected IP core is delivered in the form of an encrypted authenticated bitstream from the CV to the SD, where the IP bitstream, along with an HMAC key and a HMAC digest of the IP bitstream, is encrypted with a secret IP key K_{IP}, as described in Section III.A. The CIM, delivered together with the protected IP core, is used to generate K_{IP} to decrypt the protected IP core, authenticate its integrity and direct its configuration.

The CIM is created by the CV and has a typical block diagram shown in Fig. 2. Basically, the CIM comprises a symmetric decryption engine, a hash function, a comparator, an ICAP interface, a device identifier (e.g., device DNA) interface, three key registers and the associated control unit. One of the three key registers is pre-loaded with the CV's secret key K_{CV}, which is to be used for deriving the secret IP key K_{IP} for IP decryption; the other two registers are empty initially. The secret key K_{CV} is hidden and obfuscated in the routing information of the CIM's bitstream, which is generally believed to be the hardest to reverse accurately [36].

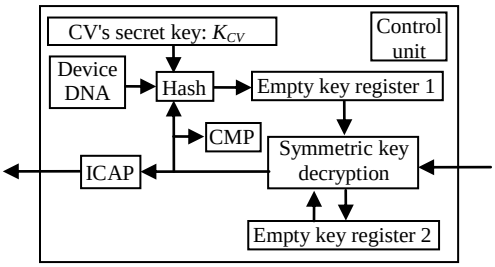


Fig. 2. Block diagram of the core installation module (CIM).

C. Protocol Execution

The proposed licensing protocol is executed in three phases detailed as follows.

(1) *Device Enrollment:* After the HM produces and delivers the FPGA chips to the FV according to the contract, the FV divides the chips into small batches and stores a *secret device key* K_{FV}^{i} in the on-chip NVM of the devices, where *i* is the batch index. This key will be used to encrypt and decrypt the CIMs of the IP cores that are licensed for use on its corresponding batch of FPGA devices. After the device enrollment, each chip has a record of $\{\#ID, i, K_{FV}^{i}\}$ in the FV's database.

(2) *IP Core Enrollment:* CVs who want to sell their IP cores in the FV's *IP store* also need to enroll them into the FV's database. The CV needs to provide the description of the core functionality and resource consumption together with a compiled IP model that can be used only for functional simulation but not synthesis. Besides, the CV also sends to the FV the associated CIM of the IP core. To enhance the secrecy of K_{CV} and attack resistance of the cryptographic components (i.e., the symmetric decryption engine and the hash algorithm) in the CIM, the CV can refresh the CIM in the FV's database after certain time with an updated version using a different K_{CV} and the state-of-the-art cryptographic components.

For an enrolled IP core IP_j, the FV issues a core identity #IP to the CV. Hence, each IP core is enrolled into the FV's database as {*#IP*, CIM_j, core description}. The CV also keeps a

[1] The encrypted authenticated bitstream is generated by encrypting the plaintext bitstream, an HMAC key and the HMAC digest of the bitstream with a secret key.

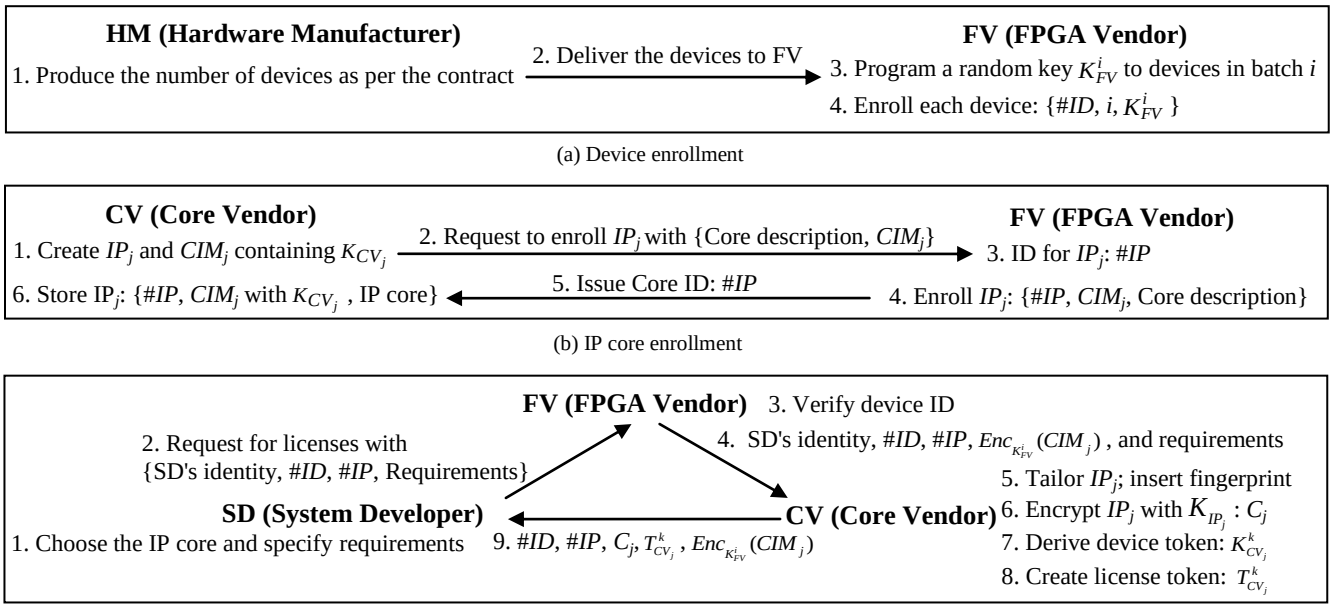


Fig. 3. Proposed protocol for the per-device IP licensing scheme.

record of the IP core as {*#IP*, CIM_j with K_{CV_j} , IP content IP_j and its related information}.

(3) *IP Core Licensing*: When an SD decides to adopt an external IP core for his system, he searches in the FV's IP store, checks the descriptions of the qualified IP cores and their functional simulation results, and selects the one that best suits his requirements. It should be noted that the CIM is securely stored by the FV and inaccessible to the SD. The SD initiates a licensing request for the chosen IP core to the FV. The request includes the identification information of the SD, the ID of the selected IP core, the device IDs of the chips purchased from the FV, and the implementation requirements. Upon receiving the licensing request, the FV checks each received device ID against his database of enrolled devices. If no matching ID is found, the device is not legally produced and the licensing request for the device will be rejected. The FV may be alerted to take action to trace the source of illegal chips. If all the devices' IDs are found in his database, the FV will select a random HMAC key, use the key to generate the digest for the CIM, and then encrypt the CIM, HMAC key and digest with the on-chip secret key K_{FV}^i . The encrypted authenticated CIM is denoted as $Enc_{K_{FV}^i}(CIM)$ [2]. Finally, the FV transmits the IP request to the CV of the selected IP core, along with the device IDs and the encrypted CIM, $Enc_{K_{FV}^i}(CIM)$. Depending on the number of batches to which the devices belong, there can be several instances of encrypted CIM.

The following steps are to be carried out by the CV. The requested IP_j is first tailored to meet the implementation requirements of the SD. To deter the SD from IP abuse and thwart collusion attacks, a fingerprint [19-21] that contains the SD's identity and the CV's ownership information can be inserted into the IP core to identify the IP licensee from each licensed IP instance. Fingerprinting techniques such as [21] can insert a blind fingerprint that carries both the SD's signature as well as the CV's watermark at gate-level with little design effort and overhead. Interested readers can refer to [19-21] for more details about the fingerprinting of FPGA IPs. A *secret IP key* K_{IP_j} is then selected to encrypt the IP core in the encrypted authenticated form, i.e., $C_j = Enc_{K_{IP_j}}(IP_j)$, for this transaction, and a different K_{IP_j} is used for each SD. Using a different K_{IP_j} in each transaction enhances the security of C_j. After all, the IP_j for each SD has to be independently encrypted and is unique due to the different implementation requirement. Meanwhile, a *secret device token* $K_{CV_j}^k = Hash(K_{CV_j}, \#ID)$ for each device k is also generated by the CV based on the device ID and his secret key K_{CV_j} . The device token is then used to encrypt the secret IP key to create a *device-specific license token* $T_{CV_j}^k = Enc_{K_{CV_j}^k}(K_{IP_j})$ in tamper-proof encrypted authenticated form. After invoicing the SD, the CV delivers the encrypted IP core C_j along with the license token $T_{CV_j}^k$ and the encrypted CIM, $Enc_{K_{FV}^i}(CIM)$ for each device k of batch i ordered or purchased from the FV.

The above protocol communications are depicted in Fig. 3.

D. Implementation of Multi-IP Core System

After completing the system design that contains in-house IP modules and externally licensed IP cores, the SD stores and queues all the design modules up in the external NVM as shown in Fig. 4. The queue can be divided into two parts, where the first part comprises external IP cores obtained through licensing and the second part comprises the internally designed cores, free third-party IP cores and other non-security sensitive functional blocks. The first part consists of several segments, each corresponding to an IP core. Each segment contains one encrypted CIM, one license token and the encrypted core. Upon

[2] In fact, all the encrypted data in our scheme are in the encrypted authenticated form. For brevity, they will be referred to as encrypted data.

powered up, the chip is first loaded with the encrypted CIM_1 of the first IP core. The CIM_1 is deciphered by the on-chip decryption engine and authenticated against its HMAC code before it is implemented onto the FPGA. Once implemented, CIM_1 takes control of the configuration of IP_1. Firstly, the hash function of CIM_1 uses the two inputs, K_{CV_1} and #*ID*, to generate the secret device token $K^k_{CV_1}$. This device token is stored into one of the two empty key registers of CIM_1. Then the license token $T^k_{CV_1}$ is fed to the decryption engine of CIM_1 and is decrypted using $K^k_{CV_1}$ to recover the secret IP key K_{IP_1}. Upon successful authentication against the HMAC digest contained in the decrypted $T^k_{CV_1}$, the IP key K_{IP_1} is stored in the other empty key register. Next, the encrypted IP core C_1 is decrypted with K_{IP_1}. The decrypted IP_1 is authenticated against the HMAC code contained in the decrypted C_1. If the authentication is passed, the bitstream will be delivered to the configuration controller through the ICAP port for implementation. Failure in any of the above authentications will either render the clearing of the configuration and loading of a fallback bitstream or the disabling of the configuration interface to block any access to the device until a power-on reset is applied.

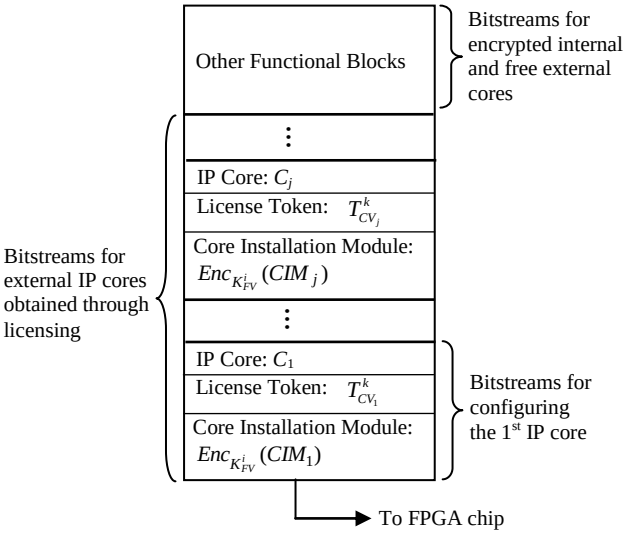
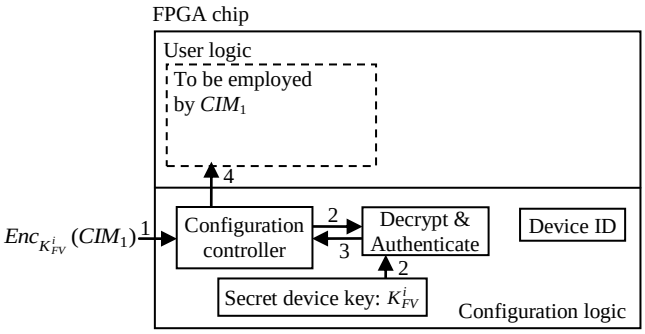
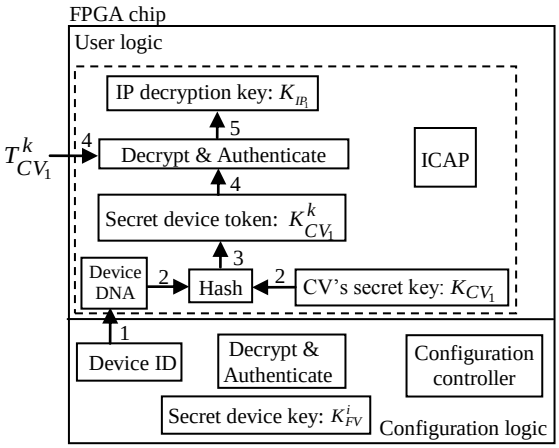


Fig. 4. Queue of design modules in external NVM of device *k* in batch *i*.

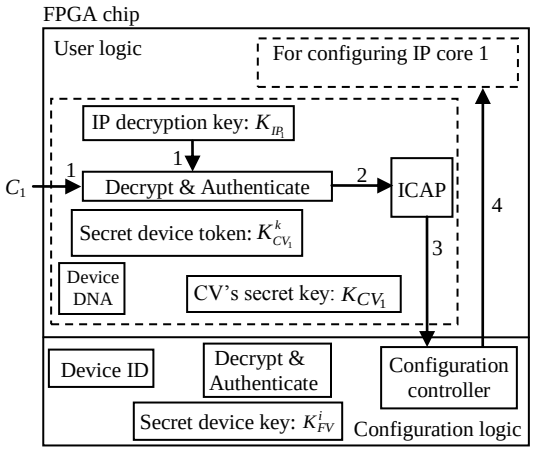
These steps are depicted in Fig. 5. The hash component and the comparator for the authentication is combined with the symmetric key decryption and represented by the component named *Decrypt & Authenticate*. After the successful configuration of IP_1, the bitstreams of the second segment is loaded. A new CIM replaces the CIM of the first IP core to control the configuration of the second core. The above steps repeat until all the licensed IP cores have been implemented. Finally, the fabrics occupied by the last CIM are released for use with the available uncommitted logic to implement the rest of the functional blocks.



(a) Steps for implementing the encrypted CIM_1



(b) Steps for establishing the secret IP key K_{IP_1}



c). Steps for configuring IP_1 with K_{IP_1}

Fig. 5. Configuration steps for protected IP_1 on device *k* in batch *i*.

IV. Evaluation and Discussion

A. Security Analyses

Fig. 6 provides an overview of the communications among FV, CV and SD for authorizing the use of IP_j on device *k*. The IP core is protected by keeping all its sensitive data (i.e., CIM, license token and IP core) secure in encrypted authenticated form at all time in transit from one party to another and when they are stored in the external NVM of the chips. Authenticity of these data is further verified with the HMAC code before they are configured onto the device to stop maliciously tampered IP core from being used. Our scheme uses standard

cryptographic primitives, i.e., AES, SHA-256 and TLS security protocol, so that the protected data in our scheme is immune to common attacks such as chosen and known plaintext attacks, replay, parallel session and type-flaw attacks.

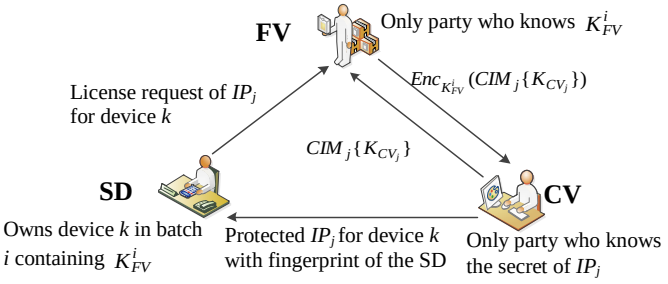


Fig. 6. Communication of IP core pertinent data among FV, CV and SD.

In [24], similar standard and provably secure symmetric cryptographic components are also used for the key establishment. Instead of directly passing the IP decryption key to the metering authority like [24], our scheme passes the CIM that contains the secret information for deriving the key. This setup can better protect the secret key from being stolen.

The following analyses show that the licensing protocol will safeguard the IP core of the CV against infringement and misappropriation by the SD, external MA and FV.

(1) *Evasion of license fee by SD*: In order to configure an encrypted IP_j onto a device k, the CIM_j associated with IP_j and a device specific license token $T_{CV_j}^k$ are required. To evade the licensing fee by implementing IP_j on other devices, an SD needs to know the secret IP key K_{IP_j}, which can only be recovered from the license token $T_{CV_j}^k$ using the secret device token $K_{CV_j}^k$. There are two possible attacks from the SD. One attack is to extract the secret device key K_{FV}^i from the on-chip NVM and use it to obtain the plaintext CIM_j. By reverse engineering CIM_j, the CV's secret key for the IP core K_{CV_j} may be extracted, from which the decryption key can be derived. The other attack is to directly steal the secret device token $K_{CV_j}^k$ or the IP key K_{IP_j} during the configuration process through the fault injection attack, side-channel attack or other more expensive physical attacks.

Extracting K_{FV}^i from the on-chip NVM using some invasive physical attacks is too expensive for the first attack. Even if the SD can succeed in retrieving K_{FV}^i, our scheme offers an added protection to track the illegal usage of his licensed IP core, let alone the barrier of reverse engineering the CIM to extract K_{CV_j}. The fraudulence of the SD can be detected by the fingerprint present in the IP instance configured in the unauthorized devices. Removal of the fingerprint from an IP instance is another high-cost investment as discussed in [21].

The second attack is even more costly. The device token and IP key are generated only on demand instead of being permanently stored in the on-chip NVM. They are stored deep inside the general purpose memory blocks and exist only for a brief duration when the licensed IP core is configured onto the device, which makes it extremely difficult to crack. Even if the SD may succeed in extracting the secret keys, the SD is unable to repudiate the alleged IP infringement when his fingerprint is found in a legal IP configured in an unauthorized device.

(2) *Attacks by MA*: It is relatively easy for an external MA to obtain the encrypted data sent from the CV to an SD by eavesdropping on the communication channel between them if the channel is not secure. However, as the cryptographic components adopted in our scheme are widely accepted as computationally secure, extracting the plaintext IP core from the encrypted data without knowing the keys is prohibitively expensive, if not impossible. From Fig. 6, it is observed that the only way for the MA to succeed in stealing the IP core is to steal first the plaintext CIM and then an encrypted IP instance. The MA needs to reverse engineer the CIM to recover the embedded K_{CV_j} before he could derive the secret key for the IP instance. As the communication channel between the FV and the CV is secure and authenticated, it prevents the plaintext CIM from being stolen by the MA through attacks such as eavesdropping on the channel, replay attacks, parallel session attacks and type-flaw attacks. Even though the MA may obtain the CIM by some sophisticated attacks and extract K_{CV_j} by reverse engineering the CIM, he has not yet succeeded in stealing the IP core. This is because K_{CV_j} is changed after certain period by the CV. The MA could only succeed in his attack provided that he also manages to obtain an encrypted IP instance that has its encryption key derived from the same K_{CV_j} for the CIM in his possession. The probability that he could intercept a matching IP instance transmitted through a different channel between another pair of communicators (CV and SD) at a different time is very slim, and the jeopardy is limited to only those matching IP instances of the stolen CIM.

(3) *Breaching of trust by FV*: In [15, 23], the FV can derive the secret key straightforwardly for the IP decryption, which increase the potential liability of the FV for frauds. To minimize the liability of the FV, our protocol does not assume that the FV is fully trusted. The secret information pertaining to the IP core is refrained from being accessible by the FV, except the plaintext CIM. Some additional measures to fortify the CIM to increase the barrier for the FV to steal the IP core will be presented in Section IV.B. Collusion attack between the FV and the SDs is also thwarted by fingerprinting the IP core. The stakes are high as the fingerprint of the misappropriated IP instance will expose the SD who participated in the collusion.

To satisfy the desiderata in Table I, only the plaintext CIM is made known to the FV and standard secure internet protocol is adopted for the communications between the FV and the CV. These make it harder to exploit the FV as an oracle to mount a successful parallel session attack. The above analyses show that the cost of a successful attack to steal an IP core will be prohibitively expensive if the FV acts honestly. Even when this assumption is not met, i.e., the FV conducts maliciously, our scheme is still more secure than the protocols of [15, 23].

B. Possible Security Enhancements

The FV can use fault tolerant and side-channel resistant on-chip cryptographic components to protect the on-chip secret

device key K_{FV}^{i}, which make it harder for an MA to obtain the plaintext CIM. In our scheme, each K_{FV}^{i} is used for only a small batch of devices. The leakage of a K_{FV}^{i} affects only the IP cores that are implemented on a particular batch of devices. By carefully controlling the size of a batch, the benefit of extracting the key can be reduced to an extent that the expense of the attack can hardly be justified. In fact, the benefit can be further reduced by blacking out the batches of devices from the list of legal devices once the security of their device keys are suspected to be compromised. Only after these devices are reprogrammed by the FV with the new device keys can they be used as trusted implementation platforms and participating in the licensing protocol again.

To make the extraction task harder for the MA, K_{CV_j} is proposed to be hidden in the routing information of CIM's bitstream as in [36]. Alternatively, white-box block ciphers [37, 38], which can protect the embedded secret key through strong obfuscation, can be used as the decryption engine of CIM to raise the barrier of successfully extracting the key.

Besides, the proposed scheme can be strengthened by adding an ancillary *establishment module* (EM) in the external NVM of each FPGA chip. The EM consists of an Elliptic Curve Diffie-Hellman (ECDH) core, an AES core and a unique private key SK_{EM}^{k} for device *k*. The EM is encrypted by the secret device key K_{FV}^{i}. The public-private key pair { PK_{EM}^{k}, SK_{EM}^{k} } of the EM for each device is stored in the FV's database. The public key PK_{EM}^{k} for the device *k* is sent with the IP request from the FV to the CV in Step 4 of Fig. 3(c). Using ECDH with PK_{EM}^{k} and his private key SK_{CV}, the CV will generate a shared key to encrypt his CIM by AES. The public key PK_{CV} of the CV is transmitted to the SD along with the encrypted CIM in Step 9 of Fig. 3(c). The implementation steps are similar to the current scheme, except that the decryption key for the CIM is firstly generated by the ECDH core of the EM using SK_{EM}^{k} and PK_{CV}. The additional cost of the strengthened scheme is only the ECDH core as the same AES core can be shared by the EM and the CIM. The advantages are that all the important data related to the IP core, including the CIM, are in encrypted form when they leave the site of the CV. The secret device key K_{FV}^{i} is used merely to protect the EM of the FV instead of various CIMs of different CVs, thus effectively avoiding the chosen and known plaintext attacks. Although ECDH core is employed as in [15, 23], it does not require the FV to act as the fully trusted TTP and avoids any party except the CV from knowing the secret information related to the IP core. More importantly, there is no need to transport the devices to the third party site for key installation.

Finally, although our scheme is based on commercially available hardware primitives, emerging new primitives can also be used once they become commercially available. For example, if the PUF primitive is available in commodity chips, it can be used in place of the current hardwired device ID to generate the secret device key K_{FV}^{i}. With the tamper-resistant PUF generating the secret key only on demand, the invasive and time consuming micro-probing attack can also be thwarted.

C. Practicality of the Scheme

Our scheme can be immediately applied in current FPGA IP market. Apart from basing on only the commercially available primitives and features, the scheme does not change current FPGA design and implementation flow. Most FPGA chips that support on-chip decryption are equipped with more than one secret key memory. For example, even the FPGA devices from the low-cost Xilinx Spartan-6 family support the hardwired decipher to read the secret key from a battery-backed SRAM-based key storage besides the on-chip NVM. Hence, the way by which the SD used to protect his own internally designed functional blocks will not be constrained by the on-chip NVM key storage taken up by the FV's secret device key K_{FV}^{i}. He can protect his own designs by encrypting them using a secret key and store the secret key in the alternative key storage. During the implementation of the final system, the SD's own IP cores will be loaded onto the configuration controller after the externally purchased IP cores have been implemented. The configuration controller will instruct the hardwired decipher to decrypt these cores before configuring them to the previously planned locations of the chip.

Modular design methodology and partial re-configuration are required in our scheme. Functional blocks are implemented onto the chip one by one at the bitstream level. This makes it impossible to combine all the functional blocks of the system before a system-level synthesis is run for optimal performance. In fact, both the modular design method and partial re-configuration may incur some additional design effort and resource consumption. Nonetheless, these features have already been supported by the design tools from the mainstream FVs. With the continual advancement of these tools, the additional design effort and performance overhead of the overall system is expected to be reduced substantially. Each IP core has actually been maximally optimized by the CVs. Precise physical synthesis information, which includes the resource allocation and interface logic, can be incorporated into the floorplanning to better tailor the IP cores to the system requirements. By using standardized bus interface like AXI and specific tools like Xilinx PlanAhead to floorplan the functional blocks in the system, the impact on the performance of the final system due to the lack of global system optimization will become marginal.

D. Implementation Considerations

Our scheme does not require any modification to currently available FPGA devices. The FV needs only to extend his existing upfront IP core licensing services to clients who opt for pay-per-device licensing. The proposed scheme effectively inhibits the use of gray market chips and counterfeit chips for the installation of legally licensed cores, which gives the FV the added incentives to act as the principal of the CV and SD besides the validity of his service obligation established in Section III. Our scheme puts less burden on the FV than the existing schemes [15, 23]. It prevents the FV from issuing and updating the CIM, and from acting as a completely trusted TTP that may lead to greater tort liability. Unlike [15, 23], our protocol enables the CV who has the greater interest and responsibility to protect his IP cores to design and enhance the

CIM based on his security requirements. Communications among the involved parties in our scheme are simplified tremendously comparing to the protocol in [24] which requires an external TTP to act as the metering authority. An immediate benefit of our setup is that the high logistic cost of transporting the chips to and fro between the FV and the external TTP sites for programming the secret device key is avoided.

Although the reconfigurable resources occupied by the CIM will be released for other functional blocks after the configuration of protected IP cores, it is still necessary to keep the resource consumed by the CIM low to preserve as much uncommitted user logic for the configuration of IP cores and to keep the operations performed by the CIM simple and fast so as to minimize the overall system configuration time. Unlike the CIM in [15, 23], which needs to establish an ECDH core of 2706 slices, the most resource hungry component of our CIM is the symmetric key decryption engine and the hash function based authentication component if the security-enhanced EM option in Section IV.C is not considered. This is similar to the CIM of [24] whose main component is an authenticated symmetric decryption engine.

As an example, we evaluate the resource consumption of the CIM on Xilinx Vertex-6 devices. Except for the symmetric decryption core and the HMAC core, our CIM contains only three key registers, the device DNA and ICAP ports, and the associated control unit. The device DNA and ICAP ports are simply the device-specific primitives provided by the FPGA and consume no more than 100 equivalent slices together with the three key registers and the simple control unit of CIM. Both the decryption engine and the hash function can be constructed using commercially available IP cores, such as the AES core of 331 slices running at 5236 Mbps [39] and the SHA-256 core of 312 slices running at 2071 Mbps [40]. All in all, a succinct CIM consumes just less than 800 slices. The smallest Vertex-6 device XV6VLX75T has 11,640 slices, the largest device XV6VLX760 has 118,560 slices and the average size of all device types is 54,828 slices. Our CIM utilizes 6.4%, 0.6% and 1.4% of the available resources in the smallest, largest and average size devices, respectively.

To the best of our knowledge, the implementation overheads reported for the existing FPGA based IP protection schemes [13-15, 22-24, 27-29, 31] have never included the overheads required for countermeasures against common types of physical attacks. The algorithm is usually secured against direct attacks by its underlying crypto components. AES and SHA-256 are NIST-approved standards widely adopted by FVs like Xilinx and Altera in the hardwired on-chip decryption and authentication engine. Commercially available SHA-256 cores implemented on various FPGA platforms that have been validated as conforming to the Secure Hash Standard [34] using tests described in the updated Secure Hash Algorithm Validation System (SHAVS) in May 2014, can be found in [41]. Similar list can also be found for AES. Authoritative security simulation results show that the state-of-the-art attack requires $2^{126.1}$ operations to recover an AES-128 key, $2^{189.7}$ operations for AES-192 and $2^{254.4}$ operations for AES-256 [42]. The best public attacks found the input that hashes to a specific output for 52-round SHA-256 out of 64 rounds with a time complexity of 2^{255} and 57-round SHA-512 out of 80 rounds with a time complexity of 2^{511} [43]. However, the secret key of an unprotected cryptographic design is relatively much easier to be deduced by low-cost physical attacks such as power analysis and fault injection. Hence, the cost of designing a CIM with reasonable level of protection against these attacks needs to be accounted. Countermeasures [44-48] against the power analysis attacks can be broadly divided into two main types: algorithmic based and generic hardware based. The former category masks the security-critical processes by computational manipulation while the latter masks the side-channel signals by using noise generators, non-deterministic processors or side-channel resistant logic styles. The effectiveness of different countermeasures can be evaluated based on the number of samples required for a successful power analysis attack. An AES-128 module with and without generic countermeasures was implemented on an Xilinx Vertex II-Pro FPGA of a SASEBO circuit board dedicated for side-channel attack evaluation [48]. The instantaneous power traces were collected by measuring the voltage drop across a 1 Ω resistor placed in the VCCINT (1.6 V) path of the FPGA with a 1.5 GHz oscilloscope at a sampling rate of 2.5 GS/s. 3000 power traces is needed to leak the key of an unprotected AES-128 module, but the number of measurements increases to 8,000 with the noise generator, 3,000,000 with clock randomizing, and more than 100, 000, 000 with block memory content scrambling [48]. If the masked logic style is used to implement the design, the resistance against power analysis attack can be further increased by a factor of about 100 [49].

The overheads of some masked AES implementations against power analysis attacks are listed in Table IV, where the area overheads vary from 20% to 300%. For the generic hardware countermeasures [48], a short-circuit based noise generator consumes only 48 LUTs on Xilinx Vertex II pro and the block memory content scrambling method uses 8 BRAMs, 1706 LUTs and 1169 FFs. Comparing with the unprotected AES implementation, which uses 8 BRAMs, 1182 LUTs and 397 FFs, the overheads of generic hardware countermeasures may vary widely from less than 4% to more than 140%. In fact, the CV can select a countermeasure or combine any of the above mentioned methods to trade the overhead for security. For example, a combination of noise generator, clock randomizing and block memory content scrambling [48] strengthens the CIM's protection level by increasing the area and timing overheads to around 200% and 60%, respectively. The masked s-box implementation in [45] can be used together with the block memory content scrambling technique at the expense of 200% area overhead and negligible timing overhead. The CIM is estimated to be well protected against common power analysis attacks by a countermeasure of 200% or less area overhead and an acceptably low timing overhead, as exemplified by the strong masked SHA-256 based HMAC [50] with 100% area overhead and 13.4% timing penalty.

Table IV
Overheads of Boolean and multiplicative masking [44], masked S-box [45], masked S-box optimized by larger LUT in recent FPGA [46] and dual-rail precharge logic [47] methods of masked AES implementation

Performance	[44]	[45]	[46]	[47]
Area overhead	20%	60.1%	200%	300%
Timing overhead	30%	4%	50%	33.3%

An overview of countermeasures against fault attacks can be found in [51], where two major types of techniques, namely fault detection using space redundancies and using duplication are illustrated. Security analysis shows that the fraction of undetectable errors can be reduced from 2^{-r} to 2^{-2r} by using *r*-bit nonlinear redundant code instead of linear code [52]. According to [51], the actual costs of many countermeasures are close to duplication if fair comparisons are performed. Although some recent proposals claim to have area overhead lower than 100%, e.g., the method in [53] incurs less than 50% area and 2% timing overheads, we conservatively assume a countermeasure like [54] for the AES core with about 100% area overhead and no throughput penalty.

The countermeasure against the power analysis attack is applied after the duplication method on the cryptographic components in the CIM. The resulting cryptographic components are expected to consume 6 times the resources of the unprotected design. This means that the protected AES and SHA-256 cores will consume 1986 and 1872 slices, respectively. The entire CIM will consume around 4000 slices, which is still less than 8% of the resources of an average size Vertex 6 device. The resources consumed by the four versions of CIM are shown in Table V.

Table V
Resource consumptions (in # equivalent slices) on Xilinx Vertex-6 devices of the unprotected, side-channel attack (SCA) resistant, fault attack (FA) resistant, and final physical attack (PA) resistant versions of CIM

CIM	Decryption	HMAC	Others	Total	Utilization rate		
					smallest	average	largest
Unprotected	331	312	100	743	6.4%	1.4%	0.6%
SCA resistant	993	936	100	2029	17.4%	3.7%	1.7%
FA resistant	662	624	100	1386	11.9%	2.5%	1.2%
PA resistant	1986	1872	100	3958	34.0%	7.2%	3.3%

The AES and SHA-256 based HMAC can also be substituted by any state-of-the-art cryptographic algorithms, such as Keccak, the winner of the NIST secure hash algorithm (SHA-3) competition in 2012. A good overview of its implementations, including those with the protection against SCA, is provided in [55]. Keccak does not have the security hole of SHA-256, which leads to a simplified HMAC digest generation without the need for a nested approach. It also provides the mode of authenticated encryption (based on duplex construction), making it possible to replace the AES and HMAC cores by a single Keccak core to compact the CIM.

V. Conclusion

We proposed a pragmatic per-device licensing scheme in this paper, which can be directly applied to existing devices without requiring any modifications. The scheme empowers the FPGA vendor to act as a broker for the system developers and core venders. It facilitates the secure transaction of protected IP cores and effectively prevents the IP core content from being cloned, reverse engineered or tampered with by even sophisticated attackers. By fingerprinting the IP cores, misappropriation and infringement of their contractual usage by the system developers can be tracked. The scheme incurs low implementation overhead, even after adding the countermeasures against common physical attacks. This is ascribed to the simple symmetric cryptography and hash function used in the core installation module and its temporary occupancy of user logic. Without requiring an external TTP, our scheme can be applied immediately in the FPGA IP market to replace the current upfront licensing model with a more competitive per-device licensing model. Not only will the per-device cost be considerably reduced for the system developer, but the core vendor can also control the number of FPGA chips that is implemented with the IP core licensed to the system developer. An extra benefit of binding the distributed IP core to the chip identity is that the chip is identified before the licensing request is processed, which effectively denies the use of overproduced FPGA chips and counterfeit chips.

References

[1] C. Maxfield, *The Design Warrior's Guide to FPGAs: Devices, Tools and Flows*. 1st ed., Elsevier, 2004.

[2] "AXI4 interconnect paves the way to plug-and-play IP (v1.0)," Xilinx White Paper 379, October 2010.

[3] Xilinx, "Common license consortium for intellectual property," [Online]. Available: www.xilinx.com/products/alliance/signonce.htm.

[4] J.-B. Note and E. Rannaud, "From the bitstream to the netlist," in *Proc. ACM/SIGDA Int. Symp. Field-programmable gate arrays*, CA, USA, February 2008, pp. 264-264.

[5] F. Benz, A. Seffrin, and S. A. Huss, "Bil: A tool-chain for bitstream reverse-engineering," in *Proc. Int. Conf. Field Programmable Logic and Applications*, Oslo, Netherlands, 2012, pp. 735-738.

[6] "Third generation non-volatile FPGAs enable system on chip functionality," Lattice semiconductor White Paper, June 2007.

[7] "An FPGA design security solution using a secure memory device (v1.0)," Altera White Paper 01033, October 2007.

[8] "FPGA IFF copy protection using Dallas semiconductor/Maxim DS2432 secure EEPROMs (v1.1)," Xilinx App. Note 780, May 2010.

[9] "Security solutions using Spartan-3 generation FPGAs (v1.1)," Xilinx White Paper 266, April 2008.

[10] "Using high security features in Virtex-II series FPGAs (v1.0)," Xilinx App. Note 766, July 2004.

[11] "Design security in Stratix III devices (v1.5)," Altera White Paper 01010, September 2009.

[12] S. McNeil, "Solving today's design security concerns (v1.2)," Xilinx White Paper 365, July 2012.

[13] T. Kean, "Secure Configuration of a field programmable gate array," in *Proc. IEEE Symp. Field-Programmable Custom Computing Machines*, CA, USA, March-Arpil 2001, pp. 259-260.

[14] L. Bossuet, G. Gogniat, and W. Burleson, "Dynamically configurable security for SRAM FPGA bitstreams," *International Journal of Engineering Science*, vol. 2, no. 1/2, pp. 73-85, 2006.

[15] T. Guneysu, B. Moller, and C. Paar, "Dynamic intellectual property protection for reconfigurable devices," in *Proc. Int. Conf. Field-Programmable Technology*, Kitakyushu, Japan, December 2007, pp. 169-176.

[16] A. K. Jain *et al.*, "Zero overhead watermarking technique for FPGA designs," in *Proc. ACM Great Lakes symposium on VLSI*, Washington, USA, April 2003, pp. 147-152.

[17] D. Ziener, and J. Teich, "Power signature watermarking of IP cores for FPGAs," *Journal of Signal Processing Systems for Signal Image and Video Technology*, vol. 51, no. 1, pp. 123-136, April 2008.

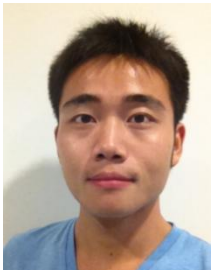
[18] W. Liang *et al.*, "A chaotic IP watermarking in physical layout level based on FPGA," *Radioengineering*, vol. 20, no. 1, pp. 118-125, April 2011.

[19] J. Lach, W. H. Mangione-Smith, and M. Potkonjak, "Fingerprinting techniques for field-programmable gate array intellectual property protection," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs.*, vol. 20, no. 10, pp. 1253-1261, October 2001.

[20] A. E. Caldwell *et al.*, "Effective iteractive techniques for fingerprinting design IP," *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Systs.*, vol. 23, no. 2, pp. 208-215, February 2004.

[21] C. H. Chang, and L. Zhang, "A blind dynamic fingerprinting technique for sequential circuit intellectual property protection," *IEEE Trans.*

Comput.-Aided Des. Integr. Circuits and Systs., vol. 33, no. 1, pp. 76-89, January 2014.
[22] T. Kean, "Cryptographic rights management of FPGA intellectual property cores," in Proc. *ACM/SIGDA Symp. Field-Programmable Gate Arrays*, CA, USA, February 2002, pages 113-118.
[23] S. Drimer *et al.* "Protecting multiple cores in a single FPGA design," 2008, [Online]. Available: http://www.saardrimer.com/sd410/papers/protect_many_cores.pdf.
[24] R. Maes, D. Schellekens, and I. Verbauwhede, "A pay-per-use licensing scheme for hardware IP cores in recent SRAM-based FPGAs," *IEEE Trans. Infor. Forensics and Security*, vol. 7, no. 1, pp. 98-108, Feb. 2012.
[25] D. G. Abraham *et al.*, "Transaction security system," *IBM Systems Journal*, vol. 30, no. 2, pp. 206-229, 1991.
[26] "FPGA design security issues: using the ispXPGA family of FPGAs to achieve high design security," Lattice White Paper, December 2003.
[27] E. Simpson, and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems*, Yokohama, Japan, October 2006, pp. 311-323.
[28] J. Guajardo *et al.*, "FPGA intrinsic PUFs and their use for IP protection," in *Proc. Workshop on Cryptographic Hardware and Embedded System*, Vienna, Austria, September 2007, pp. 63-80.
[29] S. Pappala, M. Niamat, and W. Sun, "FPGA based key generation technique for anti-counterfeiting methods using physically unclonable functions and artificial intelligence," in *Proc. Int. Conf. Field-Programmable Logic and Applications*, Oslo, Netherlands, August 2012, pp. 388-393.
[30] R. Pappu *et al.*, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026-2030, September 2002.
[31] L. Gaspar *et al.*, "Two IP protection schemes for multi-FPGA systems," in *Proc. Int. Conf. Reconfigurable Computing and FPGAs*, Cancun, Mexico, December 2012, pp. 1-6.
[32] Advanced Encryption Standard, FIPS PUB 197, 2001.
[33] "Developing tamper resistant designs with Xilinx Vertex-6 and 7 series FPGAs", Xilinx App. Note 1084, August 2012.
[34] Secure Hash Standard (SHS), FIPS PUB 180-3, 2008.
[35] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: state-of-the-art implementations and attacks," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534-574, August 2004.
[36] M. A. Gora, A. Maiti, and P. Schaumont, "A flexible design flow for software IP binding in FPGA," *IEEE Trans. Industrial Informatics*, vol. 6, no. 4, pp. 719-728, November 2010.
[37] S. Chow *et al.*, "White-box cryptography and an AES implementation," in *Proc. Workshop on Selected Areas in Cryptography*, Newfoundland, Canada, August 2002, pp. 250-270.
[38] Z. Cherif *et al.*, "Evaluation of white-box and grey-box Noekeon implementations in FPGA," in *Proc. Int. Conf. Reconfigurable Computing and FPGAs*, Quintana Roo, Mexico, Dec. 2010, pp. 310-315.
[39] "Overview datasheet - high performance AES (Rijndael) cores for Xilinx FPGA (Rev 2.4.0)," Helion Technology, January 2009.
[40] "Full datasheet - fast hash core family for Xilinx FPGA (Rev 1.9)," Helion Technology, November 2011.
[41] "SHS validated implementations," August 2014, [Online]. Available: http://csrc.nist.gov/groups/STM/cavp/documents/shs/shaval.htm.
[42] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full AES," in *Proc. Int. Conf. The Theory and Application of Cryptology and Information Security*, Seoul, South Korea, December 2011, pp. 344-371.
[43] D. Khovratovich, C. Rechberger, and A. Savelieva, "Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family," in *Proc. Int. Workshop on Fast Software Encryption*, Washington D.C., USA, March 2012, pp. 244-263.
[44] N. Mentens *et al.*, "An FPGA implementation of Rijndael: trade-offs for side-channel security," in *Proc. IFAC Workshop on Programmable Devices and Systems*, Cracow, Poland, November 2004, pp. 493-498.
[45] N. Kamoun, L. Bossuet, and A. Ghazel, "SRAM-FPGA implementation of masked S-Box based DPA countermeasure for AES," in *Proc. Int. Design and Test Workshop*, Monastir, Tunisia, Dec. 2008, pp. 74-77.
[46] F. Regazzoni, Y. Wang, and F.-X. Standaert, "FPGA implementations of the AES masked against power analysis attacks," in *Proc. Int. Workshop on Constructive Side-Channel Analysis and Secure Design*, Darmstadt, Germany, February 2011, pp. 56-66.
[47] M. Nassar *et al.*, "BCDL: a high speed balanced DPL for FPGA with global precharge and no early evaluation," in *Proc. Design, Automation and Test in Europe Conference & Exhibition*, March 2010, Dresden, Germany, pp. 849-854.
[48] T. Güneysu, and A. Moradi, "Generic side-channel countermeasures for reconfigurable devices," in *Proc. Cryptographic Hardware and Embedded Systems*, September 2011, Nara, Japan, pp. 33-48.
[49] M. Kirschbaum, and T. Popp, "Evaluation of a DPA-resistant prototype chip," in *Proc. Annual Computer Security Applications Conference*, Honolulu, Hawaii, USA, December 2009, pp. 43-50.
[50] R. McEvoy *et al.*, "Differential power analysis of HMAC based on SHA-2, and countermeasures," in *Proc. Int. Conf. Information security applications*, Jeju Island, Korea, August 2007, pp. 317-332.
[51] T. G. Malkin, F.-X. Standaert, M. Yung, "A comparative cost/security analysis of fault attack countermeasures," in *Proc. Int. Conf. Fault Diagnosis and Tolerance in Cryptography*, Yokohama, Japan, October 2006, pp. 159-172.
[52] M. Karpovsky, K. J. Kulikowski, and A. Taubin, "Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard", in *Proc. Int. Conf. Dependable Systems and Networks*, Florence, Italy, June 2004, pp. 93-101.
[53] C. N. Zhang, Q. Yu, and X. W. Liu, "A hybrid fault tolerant approach for AES," *Int. Journal of Network Security*, vol. 15, no. 1, pp. 263-269, January 2013.
[54] M. Joye, P. Manet, and J.-B. Rigaud, "Strengthening hardware AES implementations against fault attacks," *Information Security, IET*, vol. 1, no. 3, pp. 106-110, September 2007.
[55] G. Bertoni *et al.*,"Keccak implementation overview, version 3.2," May 2012, [Online]. Available: http://keccak.noekeon.org/Keccak-implementation-3.2.pdf.



Li Zhang (S'11) received the B.Eng. (Hons) degree in Electrical and Electronics Engineering from Nanyang Technological University (NTU), Singapore, in 2010. He is currently pursuing the Ph.D degree in the division of Circuits and Systems, School of EEE, NTU, Singapore.

His research interests are in hardware security, which includes hardware IP watermarking and fingerprinting, active IC metering, and hardware Trojan detection and prevention.

Chip-Hong Chang (S'92-M'98-SM'03) received the B.Eng. (Hons.) degree from the National University of Singapore in 1989, and the M. Eng. and Ph.D. degrees from Nanyang Technological University (NTU) in 1993 and 1998, respectively. He served as a Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering (EEE) of NTU in 1999, where he is currently an Associate Professor. He holds joint appointments with the university as Assistant Chair of Alumni of the School of EEE from 2008 to 2014, Deputy Director of the Center for High Performance Embedded Systems from 2000 to 2011, and the Program Director of the Center for Integrated Circuits and Systems from 2003 to 2009. He has coedited one book, published four book chapters and around 200 research papers in refereed international journals and conferences. His current research interests include hardware security and trust, low power and fault-tolerant arithmetic circuits and digital filter design.

Dr. Chang has served as Associate Editor of IEEE Access since 2013, IEEE Transactions on Circuits and Systems-I from 2010-2013, IEEE Transactions on Very Large Scale Integration (VLSI) Systems since 2011, Integration, the VLSI Journal since 2013 and Microelectronics Journal since 2014, and Editorial Advisory Board Member of Open Electrical and Electronic Engineering Journal and Journal of Electrical and Computer Engineering. He also guest edited several journal special issues and served in many international conference advisory and technical program committees. He is a Fellow of the IET.