FPGA Intrinsic PUFs and Their Use for IP Protection

Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls

Information and System Security Group
Philips Research Laboratories, Eindhoven, The Netherlands
{Jorge.Guajardo,Sandeep.Kumar,Geert.Jan.Schrijen,Pim.Tuyls}@philips.com

Abstract. Counterfeiting of valuable goods in general and that of IP (embedded software) in particular leads to big revenue losses and is therefore a threat to the industry. In [31], Simpson and Schaumont proposed a fundamentally different approach to IP protection on FPGAs based on the use of Physical Unclonable Functions (PUFs). Their work only assumes the existence of a PUF on the FPGAs without actually proposing a PUF construction. In this paper, we propose new protocols for the IP protection problem on FPGAs and provide the first construction of an intrinsic PUF based on SRAM memory randomness present on current FPGAs. We analyze SRAM-based PUF statistical properties and investigate the trade offs that can be made when implementing a fuzzy extractor.

1 Introduction

Many valuable goods such as banknotes, watches, clothes, software, content, DVDs, spare parts for cars, etc. are being counterfeited nowadays. Since the counterfeiters have much less research and development costs than the companies that originally developed these products, this leads without doubt to big revenue losses for those companies. On top of this fact their reputation is often damaged as well. Counterfeiting is also threatening the IP (i.e. the embedded software) economy. IP companies typically spend a lot of effort (and hence money) in developing new valuable designs. This ranges from blue prints of ASICs that consume much less energy or are much faster to smart functionality implemented on a FPGA (e.g. a clever filter function).

1.1 The Problem of IP Protection on Reconfigurable Hardware

The main example of reconfigurable hardware that we consider in this paper are S-RAM (Static RAM) Field Programmable Gate Arrays (FPGAs). Essentially they can be thought of as configurable hardware that can be programmed to carry out specific functionality. They are very popular for several reasons: i) the upfront investment cost is very low compared to that of ASICs and ii) they are very flexible since they can be reconfigured in the field.

In order to program a FPGA, a bitstream that embeds its functionality has to be developed. The bitstream is stored in external memory (*e.g.* PROM). At power-up, the bitstream is then transmitted to the FPGA. Once loaded the FPGA is configured and ready to carry out its functionality. We stress that most of the value is in the bitstream. Indeed when the bitstream is copied and stored in the external memory of another FPGA, another chip with the same functionality is obtained. Since, the bitstream is often loaded without any protection from the external memory to the FPGA it is relatively easy for an attacker to capture the bitstream and make a copy without further research and development costs. This attack which is easy to carry out, is nowadays called the *cloning* attack.

Clearly encryption of the bitstream with a key that is specific to a particular FPGA would solve the problem. This observation is due to Kean [21], who also proposes an associated protocol to support IP protection. The protocol is based on bitstream encryption using a key stored in non-volatile memory on the FPGA. By eavesdropping the bus between the external memory and the FPGA the attacker can only obtain an encrypted version of the design. As long as the secret key is securely stored on the FPGA, the attacker can not perform a successful cloning attack. One general problem with this solution is that there is no non-volatile memory on the vast majority of SRAM FPGAs to store a long-term key. In order to solve this problem two main solutions have been proposed: (i) some non-volatile memory such as flash is added to the FPGA and (ii) the FPGA stores a long-term key in a few hundred bits of dedicated RAM backed-up by an externally connected battery. It is clear that the previously mentioned solutions come with an additional cost. The second solution has the additional disadvantage that the battery has only a limited life time and that batteries can get damaged which shortens further their life-time. Therefore, it is interesting to look for new solutions.

Notice that there are certain problems that can not be easily solved via bitstream encryption alone. Simpson and Schaumont [31] have identified two potential problems if the aim of the solution is to secure third party intellectual property and software modules. These are: (i) Intellectual Property (IP) authentication by system (SYS) developers as well as authentication of the hardware platform (where the software IP is running) by the IP providers (IPP) and (ii) protection of the software that is running on the processors configured on the FPGA. We notice that there are other security services which can be envisioned between the different parties involved in the chain, from hardware manufacturer (HWM) to End User. Table 1 summarizes security services that can be required by different parties in the overall IP protection chain [20, 21, 15]. These parties include: the end user, the FPGA customer, the system integrator or designer (SYS), the hardware IP-Provider or core vendor (IPP), the hardware (FPGA) manufacturer (HWM) or vendor, the CAD software vendor, and a Trusted Third Party (TTP). In the remainder of the paper we will only deal with the SYS, IPP, HWM, and TTP. We refer to Kean [21] for a detailed description of the parties involved in the FPGA IP chain.

Table 1. Security Services in the IP Protection Chain

	Security Service	Description
S1	IP authenticates Hardware	IP can only be executed on one specific hardware device, hence it can not be cloned.
S2	Hardware authenticates IP	The hardware platform (FPGA) detects tampering with the IP and hence runs only authentic IP.
S3	Complete design confidentiality	The legitimate client (this could be the system integrator, the end user, etc.) has only access to the design functionality as a black box (input/output behavior). No other party (in addition to the design developer) knows anything about the hardware IP.
S4	Secure hardware IP updating	Given that there is already an authentic design running on the FPGA, the IP provider would like to update it and at a minimum keep all the security guarantees that the previous design kept.
S5	Design traceability	Given an IP block, the designer can trace back who the intended recipient of the design was.
S6	User privacy	A design should not be linkable to the identity of the end-user

1.2 Our Contributions

In this paper, we propose new and improved protocols for IP protection on FPGAs. We show that the protocols of [31], while secure (i.e. we do not present any attacks against them), can be considerably simplified. We describe simplifications in terms of communication complexity, assumptions, and number of encryptions performed. We believe that one reason for this is the fact that the assumptions made on the primitives used in [31] were not clearly stated. To this end, we provide a review of the primitives and of the encryption schemes that can be used in such protocols. We then clearly state the assumptions made about these primitives and base the security analysis of our newly proposed protocols on them. A second contribution of the paper is the introduction of protocols which provide privacy from the TTP in the so-called "*honest-but-curious model*"[1]. In other words, previous protocols allow the TTP to have access to the IP block exchanged between the IPP and the SYS. In practice, this might not be desirable from the IPP's point of view. Thus, we introduce a protocol that allows for this at the cost of introducing a public-key (PK) based operation (a *hybrid* approach). The cost is minimal and it does not affect the resource requirements of the FPGA implementation when compared to the work in [31]. This is achieved by performing the PK operation during the *online* phase of the protocol. The third and final contribution of the paper regards the implementation of an actual Physical Unclonable Function (PUF) on a FPGA which is *intrinsic* to the FPGA. By this we mean that the PUF is intrinsically present on the FPGA and thus, it requires no modifications to the actual hardware. As far as we are aware, this is the first time that such a PUF is reported in the literature. Notice that the work of [31] only *assumes* the existence of such PUF on a FPGA and models its behavior via an AES module.

In [15] we presented a solution for IP protection based on PUFs with a focus on protocols based on public-key cryptography. It is shown there that by using public-key cryptography the IP can be kept completely confidential, i.e. even the TTP can not learn any information about the IP. Cryptographically speaking

[1] In the "honest-but-curious" model, the parties follow the rules of the protocol (in contrast to the malicious model), but try to learn as much as they can.

one says that the results of [15] are secure in the *malicious* model while those in this paper are secure in the honest but curious model. The result of [15] come at the price of having to perform a public-key decryption on the FPGA.

Organization. In Section 2 we provide an overview of the cryptographic components, such as PUFs and symmetric-key primitives, used in this paper. In Sects. 3 and 4, we use these constructions to simplify the protocols proposed in [31]. We also introduce a protocol that provides total privacy, even from the TTP (in the honest-but-curious model). Section 5 introduces intrinsic PUFs and a construction based on the properties of SRAM blocks present on FPGAs. In addition, we analyze the entropy and statistical properties of SRAM-based PUFs. We end in Sect. 6 with a brief analysis of possible fuzzy extractor implementation options.

2 Preliminaries

2.1 Physical Unclonable Functions

Physical Unclonable Functions have essentially two parts: i) a physical part and ii) an operational part. The physical part is a physical system that is very difficult to clone[2]. It inherits its unclonability from uncontrollable process variations during manufacturing. In the case of PUFs on an IC such process variations are typically deep-submicron variations such as doping variations in transistors. The operational part corresponds to the function. In order to turn the physical system into a *function* a set of challenges C_i (stimuli) has to be available to which the system responds with a set of sufficiently different responses R_i.

We briefly repeat the assumptions on the PUF that we need [15]:

1. It is assumed that the response of a challenge-response pair (CRP) (C_i, R_i) gives only a negligible amount of information on the response R_j of another CRP (C_j, R_j) with $i \neq j$.
2. Without the PUF, the response R to a challenge C can only be guessed with negligible probability.
3. Under attack, the function implemented by the PUF is drastically changed.

We distinguish between two different situations. First, we assume that there is a large number of challenge response pairs (C_i, R_i), $i = 1, \ldots, N$ available for the PUF; i.e. the PUF has so many CRPs such that an attack (performed during a limited amount of time) based on exhaustively measuring the CRPs only has a negligible probability of success and, in particular, $1/N \approx 2^{-k}$ for large $k \approx 100$ [26, 32]. We refer to this case as strong PUFs. If the number of different CRPs N is rather small, we refer to it as a weak PUF.

Examples of PUFs include optical PUFs [26, 27], silicon PUFs [13] and coating PUFs [35]. Although coating PUFs are very cheap to produce they still need

[2] Note that this stands in sharp contrast to Quantum Cryptography when cloning is impossible due to the basic laws of nature.

a small additional manufacturing step. In this paper we introduce the notion of an *Intrinsic* PUF (IPUF), *i.e.* a PUF that is inherently present in a device due to its deep-submicron manufacturing process variations and no additional hardware has to be added for embedding the PUF. We introduce an example of such a PUF in Sect. 5.

The responses of a PUF can not be straightforwardly used as a key (as in e.g. [35]) in a cryptographic primitive for two reasons. First, the responses of a PUF are obtained through measurements which are typically noisy. When a PUF is challenged with C_i a response R'_i which is a noisy version of R_i is obtained. This leads to a problem since cryptographic functions are very sensitive to noise on their inputs. Second, the responses of a PUF are not uniformly distributed. Hence, even if there was no noise, the response would not form a cryptographically secure key.

In order to deal with both issues a Fuzzy Extractor or Helper data algorithm has to be used. For the precise definition of a Fuzzy Extractor and Helper Data algorithm we refer to [10, 25]. This primitive deals with both issues by implementing first an *information reconciliation phase* and secondly by applying a *privacy amplification* or randomness extraction primitive. The Fuzzy Extractor is applied as well during the *enrollment* as the *key reconstruction* phase. During the enrollment or training phase, the Fuzzy Extractor takes as input the PUF response R. It generates the key K for the first time together with so-called *helper data* W. The helper data W is constructed in such a way that it allows to correct for the noise in the PUF responses while at the same time revealing no information on the secret key K. During the key reconstruction phase, the Fuzzy Extractor takes as input the helper data W and a noisy PUF response R' to reconstruct the key K.

2.2 On Authenticated Encryption

There has been considerable work in the crypto community on authenticated encryption. In other words, how to obtain privacy and integrity at the same time in the symmetric-key setting. Our aim in this section is to summarize known results and to caution against combining primitives without any formal analysis. In later sections, we will use these results to justify the security of the schemes that we propose or to notice potential vulnerabilities of the proposed schemes. Throughout the paper we will refer to *encrypting* [6], denoted $\texttt{Enc}_K(\cdot)$, meaning an encryption scheme providing semantic security under chosen plaintext attacks[3] [14, 11], commonly written IND-CPA. Finally, we write $\texttt{MAC}_K(\cdot)$, to indicate a message authenticating code (MAC) computed with the secret-key K providing integrity of plain texts (see [5]). Next, we recall different constructions considered in the literature and their conclusions.

Bellare and Namprempre [5] analyze three *generic* composition paradigms to provide privacy and authentication via symmetric-key encryption schemes. We

[3] There are stronger versions of security, such as semantic security under chosen ciphertext attacks (IND-CCA), however, common modes of operation (e.g. CBC) only provide IND-CPA.

emphasize that their analysis is for generic composition, meaning that they make black-box use of symmetric encryption and MAC schemes. Three composition methods are considered: (i) Encrypt-and-MAC := $\texttt{Enc}_{K_{enc}}(M)||\texttt{MAC}_{K_{MAC}}(M)$, (ii) MAC-then-encrypt:= $\texttt{Enc}_{K_{enc}}(M||\texttt{MAC}_{K_{MAC}}(M))$, and (iii) Encrypt-then-MAC:= $D||\texttt{MAC}_{K_{MAC}}(D)$, where $D = \texttt{Enc}_{K_{enc}}(M)$. It is proved in [5] that under generic composition the Encrypt-and-MAC scheme fails to preserve privacy, while providing integrity. Furthermore, this is true for *any* deterministic MAC such as [4, 3, 23]. The other two constructions preserve privacy under CPAs and provide integrity of plaintexts. We refer to [5] (see also [22]) for the details but notice that the third construction is the one that provides the strongest security guarantees. In [1], An and Bellare study whether adding redundancy to a message and then encrypting it (i.e., $\texttt{Enc}_K(M||\tau)$ where $\tau = h(M)$, h some function of M), provides both privacy and authenticity. They show that the privacy of the encryption-with-redundancy is inherited from the original encryption scheme $\texttt{Enc}_K(\cdot)$. However, integrity depends on whether the function h is public or keyed with a secret key. In particular, for redundancy computed via public functions known to the adversary (e.g. via a keyless hash function like SHA-1), the resulting scheme does not provide integrity. On the other hand, if the redundancy function is computed incorporating a secret key, then the resulting scheme provides integrity. We notice that this is probably the reason why in [31], the integrity information is encrypted with a second key[4]. Finally, a number of schemes have been explicitly developed to provide authentication and privacy in the symmetric-key setting (see for example [36, 19, 29]).

3 Offline HW/SW Authentication for FPGAs

In the remainder of this paper, we will denote an IP block by *SW* and use this terminology interchangeably. In [31], Sympson and Schaumont describe a protocol which provides hardware IP authentication (S1) and hardware platform authentication (S2). For completeness, the protocol is shown in Fig. 1. In Fig. 1, we have written $\texttt{Enc}(\cdot)$ to mean the symmetric encryption of the argument. Although, no assumption is mentioned in [31], we assume that $\texttt{Enc}(\cdot)$ is IND-CPA secure. The protocol in [31] assumes that the hardware manufacturer implements a security module on the FPGA. This security module includes a PUF and an AES decryption module, which allows to decrypt encrypted configuration files and/or other software IP blocks. However, in [31] there is no discussion about fuzzy extractors, which are required to deal with noise and extract randomness from a PUF. The protocol assumes secure and authenticated channels between all parties involved in the protocol during the enrollment and online phases. During the offline phase an unauthenticated public channel is assumed. Notice that the public channel allows the TTP to have access to *SW* since it is only encrypted with a PUF response, which is stored in the TTP database. We ask the following questions:

[4] Reference [31] uses a public hash function for integrity.

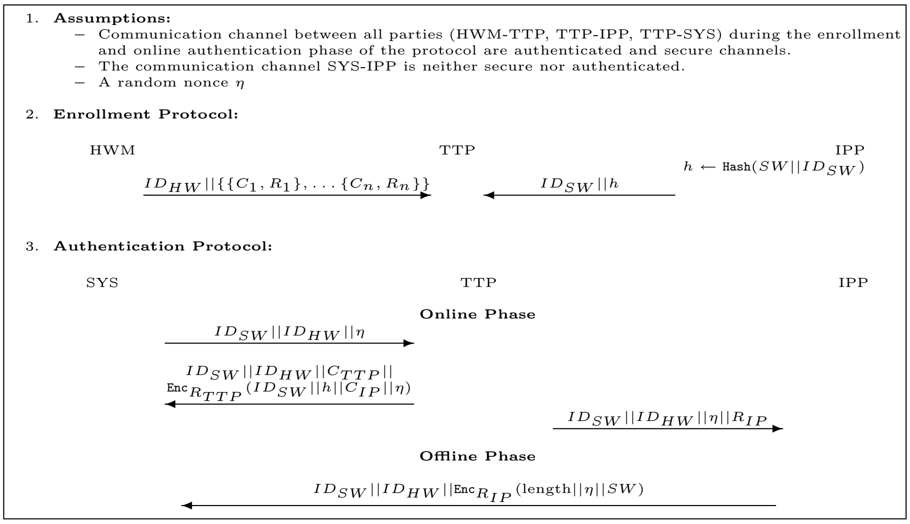

Fig. 1. Offline HW/SW authentication for FPGAs according to [31]

1. Can we simplify the protocol of [31] and still attain the same security guarantees? In particular, the protocol of [31] does not take advantage of the assumptions made on the primitives, which leads to unnecessarily complicated protocols. For example, is it possible to come up with a similar protocol, which does not require secure channels during the online phase of the protocol?
2. Can we design a protocol with similar security guarantees and which does not allow the TTP to know the software SW? In other words, can we provide complete privacy of the SW (even the TTP has no access to SW)? Notice that the protocol in [31] does not provide this type of privacy guarantee since the TTP knows R_{IP} and the SYS-IPP channel is public.
3. Is a protocol with four messages required or can we simplify it? In other words, can we reduce the communication complexity of the protocol in [31].
4. In Sect. 2.2 we saw how in general $\mathtt{Enc}_K(M||\tau)$, where $\tau = h(M)$ and h a public function, does not provide integrity. Similarly, Encrypt-and-MAC provides integrity but violates privacy. As a result, [31] provide the following construction $\mathtt{Enc}_{K_1}(h(M))||\mathtt{Enc}_{K_2}(M)$. This requires two decryptions and one hash computation. Is it possible to simplify the protocol, so that only *one* encryption and one MAC are required ?

In the next section, we provide answers to these questions in a constructive manner. In particular, we design simplified protocols which (in some cases) do not allow the TTP to learn any information about the IP block. In addition, our protocols require only one encryption and one MAC as opposed to two encryptions and one MAC (hash) operation as in [31].

4 New HW/SW Authentication Protocols for FPGAs

In this section, we introduce two new protocols and analyze them. First, we propose a protocol that provides partial privacy (only the TTP is able to learn the IP block) and integrity. Then, we introduce a protocol which provides total privacy, in the sense that not even the TTP has access to the IP block originating from the IP provider. Notice that in our protocols C_i denotes the PUF challenge *and* the corresponding helper data, which we use to reconstruct the PUF response R_i from a noisy version R'_i. Finally, we assume, as implicitly done in [31] and explicitly stated in [15], that the circuit used to obtain CRPs during the enrollment protocol is destroyed (e.g. by blowing fuses) after enrollment. Similarly, we also assume that given a challenge C_i the corresponding response R'_i is *only* available internally to the FPGA's decryption circuit. Without, this assumption, anyone could access R_i, and the protocols proposed (including those in [31]) would be completely broken. We begin by describing how the combination of bitstream encryption and a key extracted from a PUF works in practice. It consists of the following steps: (i) loading the encrypted bitstream, (ii) challenging the PUF with a challenge C_i, (iii) measuring the PUF response R'_i, (iv) retrieving helper data W from memory, (v) using a fuzzy extractor to extract the key $K \leftarrow \texttt{Rep}(R'_i, W)$, (vi) decrypting the bitstream, and finally (vii) configuring the FPGA.

4.1 New IP Protection Protocols.

For the sake of simplicity we assume that the length information is already contained[5] in the IP block denoted by SW.

1. **Assumptions:**
   – Communication channel between SYS-TTP and TTP-IPP are authenticated and secure
   – Fully trusted TTP
2. **Enrollment Protocol:**

   HWM $\xrightarrow{ID_{HW} \| \{\{C_1, R_1\}, \ldots \{C_n, R_n\}\}}$ TTP
3. **Authentication Protocol:**

   SYS $\xrightarrow{ID_{SW} \| ID_{HW}}$ TTP

   TTP $\xrightarrow{ID_{SW}}$ IPP

   IPP $\xrightarrow{SW}$ TTP

   TTP: $D \leftarrow \texttt{Enc}_{R_i}(SW \| ID_{SW})$

   TTP $\xrightarrow{C_i \| C_j \| D \| \texttt{MAC}_{R_j}(C_i \| C_j \| D)}$ SYS

Fig. 2. New IP Protection Authentication Protocol

[5] This is also a realistic assumption as bit stream configuration files for current FPGAs already have length information embedded in them.

ANALYSIS. Notice that the TTP is fully trusted in this model. Thus, it is allowed for the TTP to have access to the SW. Confidentiality of the SW follows immediately from the assumptions on the PUF. Authentication during the running of the protocol follows from the fact that we have an authenticated channel between TTP and SYS. However, after running of the protocol, $C_i||C_j||D||\mathtt{MAC}_{R_j}(C_i||C_j||D)$, where $D = \mathtt{Enc}_{R_i}(SW||ID_{SW})$ are stored in insecure non-volatile memory. In this case, privacy follows from the inability of an attacker to generate R_i corresponding to the challenge C_i and integrity of SW from $\mathtt{MAC}_{R_j}(C_i||C_j||\mathtt{Enc}_{R_i}(SW||ID_{SW}))$ and the inability of the attacker to generate R_j from C_j. This protocol has the drawback that all communications go through the TTP. In particular, every SYS has to contact the TTP to obtain the desired IP block, which could prove to be a system bottleneck. One can solve this by simply having the TTP forward pairs $\{C_i, R_i\}, \{C_j, R_j\}$ to IPP and having IPP, in turn, send $C_i||C_j||D||\mathtt{MAC}_{R_j}(C_i||C_j||D)$, where $D = \mathtt{Enc}_{R_i}(SW||ID_{SW})$ directly to the SYS. In this case, we do not assume an authenticated or secure channel between the IPP-SYS. The privacy of the SW follows simply from having SW encrypted with R_i and integrity from checking $\mathtt{MAC}_{R_j}(C_i||C_j||D)$. Notice that the pairs $\{C_i, R_i\}, \{C_j, R_j\}$ are only available to the TTP and to authentic IPPs in touch with the TTP, by assumption.

4.2 New IP Protection Protocols Providing SYS-IPP Confidentiality.

In this section, we answer positively the question of whether it is possible to develop protocols with similar properties to the previous ones but without having the TTP have access to the SW. In the following, we do not assume any of the channels to be secure. However, we make the following assumptions: (1) the channels TTP-SYS, TTP-IPP, SYS-IPP are authentic (e.g. man-in-the-middle attacks are not possible), (2) it is possible to obtain the public-key of IPP (in an authenticated way) and use it for sending encrypted data to it, and (3) the TTP is "honest-but-curious". In other words, the TTP follows the protocol in an honest manner but tries to find out as much information as possible (i.e. he wants access to SW). The resulting protocol is shown in Fig. 3.

ANALYSIS. We assume that the SYS and TTP have obtained the IPP's authentic public key and that they have established authenticated channels (SYS-TTP, TTP-IPP, IPP-SYS). Privacy and authenticity of SW follows from the Encrypt-then-Authenticate scheme, the inability of an attacker to derive R_i, R_j corresponding to C_i, C_j, and the fact that the keys used to encrypt and authenticate depend on R_i, R_j and the nonce η which is only known to the SYS and IPP. Notice that the TTP is *not* allowed to tamper with $\mathtt{Enc}_{K_{pub_{IPP}}}(\eta)$ (e.g. substitute it) since we are in the honest-but-curious setting. Thus, the protocol provides privacy with respect to the TTP as well. Notice that the cost of the protocol on the SYS side is now one decryption, one MAC, and two additional hash func-

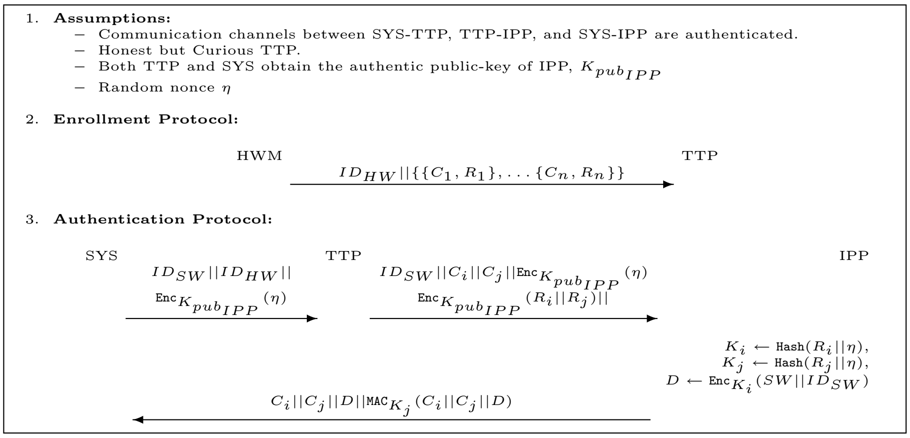

Fig. 3. IP Protection Authentication Protocol with SYS-IPP Confidentiality

tion computations. The hash function computations do not require additional hardware resources if performed via an AES-based hash as in [31].

5 FPGA Intrinsic PUFs

The key component of the previously discussed protocols is the existence of a PUF. Before introducing our new construction, we review previous PUF constructions. In [26, 27], Pappu et al. introduced optical PUFs consisting of pieces of glass with air bubbles. These PUFs can be challenged by irradiating them with a laser beam. The response is the resulting speckle pattern and it is a function of the location of the bubbles, the challenge, and the laser positioning. Gassend et al. [12] defined a Controlled Physical Random Function (CPUF) which is a PUF that can only be accessed via an algorithm that is physically bound to the PUF in an inseparable way. This control algorithm strengthens the PUF by making sure that any tampering with the control logic also destroys the PUF. Based on this idea, Gassend et al. introduced silicon Physical Random Functions (SPUF) [13] which use manufacturing process variations in integrated circuits (ICs) with identical masks to uniquely characterize each IC. The basic construction of an SPUF in [13] is based on parameterized self oscillating circuits. An additional circuit is added to measure the resulting frequency, which in turn characterizes uniquely each IC. However, SPUFs have been shown to be very sensitive to environmental variations like temperature and voltage [13]. This led to the introduction of the concept of an *arbiter-based* PUF [24]. In an arbiter-based PUF an arbiter is used to distinguish the difference in delay between two identical delay paths. By using this differential structure, the impact of environmental variations on the PUF response are minimized. In [35], Tuyls et al. presented a coating PUF. The basic idea is to cover an IC with a protective coating, doped with random dielectric particles at random locations. The PUF is challenged and measured via an array of capacitance sensors in the IC's top metal layer. Since the measurement circuit is integrated into the IC circuitry, this turns out to be a controlled PUF. Recently, Su et al. presented in [34] a custom built circuit array of cross-coupled NOR gate latches to uniquely identify an IC. Here, small transistor threshold voltage V_t differences that are caused due to process variations lead to a mismatch in the latch to store a 1 or a 0. The disadvantage of most of these approaches is the fact that custom built circuits are used or that a modification of the IC manufacturing process is required. We approach the problem by identifying an *Intrinsic* PUF which we define as a PUF generating circuit already present in the device and that requires no modification to satisfy the security goals. In the next sections, we show that SRAM memories in FPGAs can be used as an Intrinsic PUF.

5.1 PUFs Based on SRAM Memories

We begin this section by describing the structure of a typical six transistor CMOS SRAM cell [2] as shown in Fig. 4. Such a cell consists of two cross-coupled inverters (load transistors PL, PR, NL and NR) and two access transistors (AXL and AXR) connecting to the data bit-lines (BLC and BL) based on the word-line signal (WL). Each inverter consists, in turn, of a p-junction transistor (PL, PR) and an n-junction transistor (NL, NR). A key characteristic of an SRAM

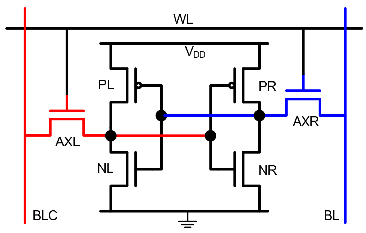

Fig. 4. Six transistor SRAM cell

cell is the static-noise margin (SNM), defined as the minimum DC noise voltage to flip the cell state. In fact, much research is aimed at optimizing the SNM while at the same time reducing the size of the cell (which tend to be opposing aims). Optimizing (increasing) the SNM results in a more stable cell, thus requiring a higher voltage to flip the state of the cell. Notice that the SNM, in turn, has been shown to be directly influenced by the threshold voltage of the cell's transistors [30]. In addition, the authors in [7] show that microscopic variations in the dopant atoms in the channel region of the MOSFET induce differences in the threshold voltage V_t of the transistors of an SRAM cell. This implies that different SRAM cells in a SRAM memory array will have slightly different threshold voltages and as a result different SNMs. We refer to [9] for a discussion of other device characteristic variations caused by intrinsic parameter fluctuations in a CMOS SRAM cell. Since such variations are well known to occur, memory designers construct SRAM cells with proper width/length ratios between the different transistors [30]. This guarantees that known fluctuations out of their control do not affect the reading and writing process of the cell under normal operation. However, during power-up, the SRAM cell's cross-coupled inverters are "floating". Therefore, the previously discussed SNM differences in the transistors will cause the stored value to tend toward a 0 or a 1, depending on the cell's specific characteristics. This effect is increased by the amplifying effect of each inverter acting on the output of the other inverter. As a result (and as we show next experimentally), the same SRAM cell will tend to start in the same state upon power-up whereas different SRAM cells will behave randomly and independently from each other.

The previous paragraph describes a device that while behaving randomly, (different cells will start-up with different values) it also has repeatable behavior (i.e. the same cell will tend to start on the same state). These, in fact, are key properties of any Physical Unclonable Function and thus, we based our new Intrinsic PUF on the start-up values of SRAM memory. We consider as a challenge a range of memory locations within a SRAM memory block. For example, we show in Sect. 6 that to derive a 128-bit secret we require about 4600 SRAM memory bits (under extreme conditions). The response are the start-up values at these locations. If the memory block used is about 512 kbits, we can expect to have close to 110 CRPs. As previously discussed, we assume a security module that allows reading of the SRAM start-up values only by the manufacturer during the enrollment process. Upon successful enrollment a fuse is blown such that the response to a challenge is only available internally inside the FPGA. As pointed out in [15], an additional advantage of SRAM-based PUFs is that their responses are immediately in binary form. This is in contrast to previously reported PUFs in which a quantization step is needed to turn an analog measurement into a binary response. Hence, the complexity of the measurement circuit is reduced.

FPGA SRAM PUF. Most of FPGAs in use today belong to the category of volatile SRAM FPGAs. The biggest manufacturers of these FPGAs, Altera and Xilinx, also provide extra built-in SRAM memory blocks that can be used by the designer to store data. For our proof of concept, we use such a FPGA with dedicated RAM blocks.

5.2 Statistical Analysis of SRAM PUFs

As previously mentioned, in order to allow for identification of a large number of FPGAs, different SRAM cells should start-up with random and independent values. In addition, SRAM startup values should exhibit robustness over time, robustness to temperature variations, and aging robustness. These criteria are described in the remainder of this section.

Robustness over Time. The Hamming distance between bit strings from repeated measurements of the same SRAM block (intra-class measurements) should be small enough, such that errors between enrollment and authentication measurements can be corrected by an error correcting code admitting efficient decoding. The main criteria here is to check the stability of the startup values over a series of intra-class measurements done over a two week period. Figure 5 shows the fractional Hamming distance between a first measurement and repeated measurements of the same SRAM block that were carried out over approximately two days. The experiment was done with four different RAM blocks, located in two different FPGAs. The measurements show that less than 4% of the startup bit values change over time.

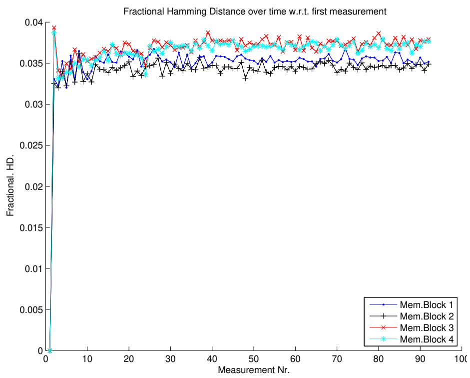

Fig. 5. SRAM startup values time test.

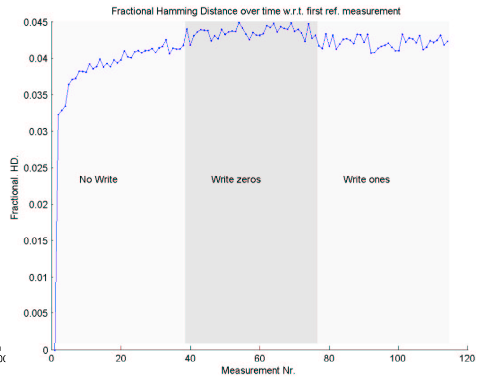

Fig. 6. SRAM startup values aging test.

Robustness to Temperature Variations. The Hamming distance between bit strings measured in the same SRAM block (intra-class) at different environmental temperatures should be small (for the same reason as mentioned above). Figure 7 shows the fractional Hamming distance between a reference measurement at 20°C and startup values measured at temperatures ranging from −20°C to 80°C. At each temperature the startup values were measured 5 times (where the FPGA was re-powered in between of the measurements) for 10 different FPGAs. Figure 7 shows that the highest number of bit errors with respect to the reference measurement occur at −20°C. The fractional Hamming distance does not exceed 14%.

Aging Robustness. Intra-class Hamming distances of the SRAM startup values should remain small, even when other data has been written into the memory before the FPGA was restarted. In particular, it is important that the startup values are unaffected by aging and the use of the SRAM blocks to store data. SRAM memory retention has been previously considered in [16, 17, 33] from a security point of view. Gutmann [16, 17] writes that SRAM memories can retain some data that has been previously stored and that this phenomenon can also affect the startup values. How long the data is retained varies with temperature.

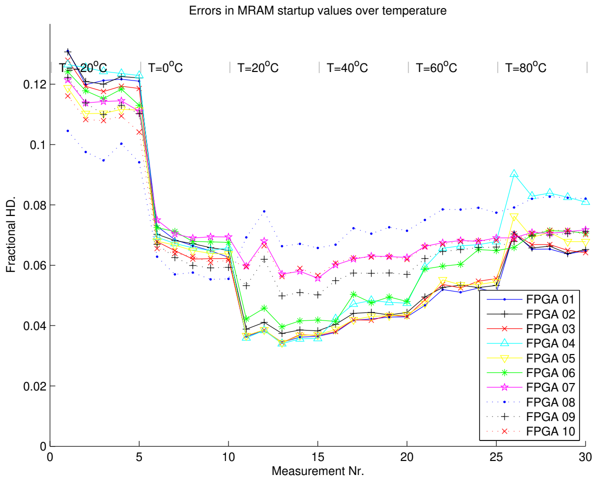

Fig. 7. SRAM startup values temperature test

Skorobogatov in [33] presents experimental evidence that show that retained data in SRAM memory is rapidly lost in a small amount of time (few msec) after startup. We have performed measurements to test the behavior of SRAM startup values after "normal memory usage". We simulated this usage by writing zeros or ones into the memory and maintaining this memory state for over 10 minutes at a time. The SRAM startup values were then read out after restarting the FPGA. Figure 6 shows the fractional Hamming distance between the bit string of a reference measurement and bit strings of startup values measured shortly after writing zeros and ones into the SRAM memory. The figure shows that storing zeros or ones into the memory has very little influence in the SRAM start-up values. The fractional Hamming distance between bit strings from an enrollment (reference) measurement and any of the other measurements does not exceed 4.5% in this test.

Identification Performance. The fractional Hamming distance between bit strings of different SRAM blocks (and different FPGAs) should be close to 50%, such that each SRAM block (and thus each FPGA) can be uniquely identified. In order to get an idea of how well the start-up bit strings from different memory blocks can be distinguished from each other, we have investigated the distribution of Hamming distances between bit strings of length 8190 bytes derived from different SRAM blocks (inter-class distribution). A histogram of inter-class Hamming distances is depicted in Fig. 8. The startup bit values of seventeen different SRAM blocks were used to create this graph. Our analysis shows that the inter-class fractional Hamming distance distribution closely matches a normal distribution with mean 49.97% and a standard deviation of 0.3%. Figure 8 also shows the histogram of intra-class Hamming distance measurements. This histogram was created by comparing 92 repeated measurements of the same SRAM block. The intra-class fractional Hamming distance distribution of startup bit strings has an average of 3.57% and a standard deviation of 0.13%.

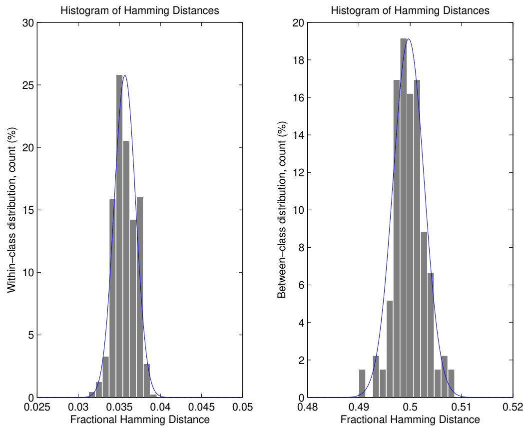

Fig. 8. Histogram of intra-class (left) and inter-class (right) Hamming distances between startup bit strings of SRAM blocks and their approximating normal distributions.

6 On the Cost of Extracting a 128-bit Key

We explained in Sect. 2.1 a Fuzzy Extractor is required to extract a secure cryptographic key from the PUF responses. It follows from Sect. 2.1 that we basically need to implement an error correction code and a universal hash function. Since the FPGA has to be able to reconstruct the key, the decoding algorithm of the error correction code has to be implemented on the FPGA.

The next subsection describes the choices that can be made to derive a 128-bit key, which could be used in combination with an IND-CPA encryption scheme and corresponding MAC in the protocols proposed in Sect. 4. Since the computations for the symmetric-key case are analogous to those of the public-key case [15] we present a brief summary of the results.

Secrecy Rate. Using the method presented in [18], to estimate the secrecy rate based a universal source coding algorithm, we obtain the following result $\mathbf{I}(R, R') = 0.76$. That implies that in order to extract a secret key of size N, we need at least $\lceil 1.32N \rceil$ source bits (in the response of the SRAM PUF).
Error Correction. Although this computation follows the same line of reasoning as in [15], we briefly repeat it here to emphasize the special properties of the symmetric-key case. In order to choose an adequate error correcting code,

we first consider the number of bits of information, which have to be at least $\lceil 1.32N \rceil$ bits, which for $N = 128$ is 171. Assuming that all bits are independent, the probability that a string of S bits will have more than t errors, denoted by P_{total}, is given by $1 - \sum_{i=0}^{t} \binom{S}{i} p_b^i (1-p_b)^{S-i}$, where p_b denotes the bit error probability. Since it follows from the experimental results that the maximum number of errors is about 14% we assume a bit error probability $p_b = 0.15$ (to be conservative). Furthermore we put our failure rate at $P_{total} = 10^{-6}$. Under the assumption that the errors are independent, a binary BCH code is a good candidate code (see for example [8, 28]). It follows from above that we have to generate at least 171-bits of information, it becomes an optimization problem to choose the best code in terms of hardware resources, number of SRAM bits required, performance, etc. For example, using $[511, 19, t = 119]$-BCH, we would need $9 \times 511 = 4599$ bits to generate 171 information bits. We note that the symmetric-key based scheme requires hence 25% less bits[6] than the public-key extraction module in [15].

7 Conclusions

In this paper, we have proposed new and efficient protocols for the IP-protection problem based on symmetric-key primitives (in contrast to [15] which purely focuses on a public-key based approach). In addition, we have introduced a new PUF construction which is unique in the sense that it is intrinsic to FPGAs and thus, it does not require modification of the hardware or the manufacturing process to be used. Currently, we are investigating other intrinsic properties of FPGAs that could be used as a PUF.

References

1. J. H. An and M. Bellare. Does Encryption with Redundancy Provide Authenticity? In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 512–528. Springer, May 6-10, 2001.
2. A. Bellaouar and M. I. Elmasry. *Low-Power Digital VLSI Design. Circuits and Systems.* Kluwer Academic Publishers, first edition, 1995.
3. M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. In N. Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, volume 1109 of *LNCS*, pages 1–15. Springer, August 18-22, 1996.
4. M. Bellare, J. Kilian, and P. Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.
5. M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, December 3-7, 2000.

[6] To be fair, we should compare 128-bit symmetric-key to a 256-bit key elliptic curve key and not to a 160-bit key as used in [15]. However, the point is that to implement a public-key based scheme, more bits are required than for a symmetric-key based one, as expected.

6. M. Bellare and P. Rogaway. Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In T. Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, December 3-7, 2000.
7. A. J. Bhavnagarwala, X. Tang, and J. D. Meindl. The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability. *IEEE Journal of Solid-State Circuits*, 36(4):658–665, April 2001.
8. R. E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley Publishing Company, first edition, 1985.
9. B. Cheng, S. Roy, and A. Asenov. The impact of random doping effects on CMOS SRAM cell. In *European Solid State Circuits Conference*, pages 219–222, Washington, DC, USA, 2004. IEEE Computer Society.
10. Y. Dodis, M. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology —- EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.
11. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography (Extended Abstract). In *ACM Symposium on Theory of Computing — STOC'91*, pages 542–552. ACM, May 6-8, 1991.
12. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled Physical Random Functions. In *ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference*, page 149, Washington, DC, USA, 2002. IEEE Computer Society.
13. B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas. Silicon physical unknown functions. In V. Atluri, editor, *ACM Conference on Computer and Communications Security — CCS 2002*, pages 148–160. ACM, November 2002.
14. S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.
15. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. Physical Unclonable Functions and Public Key Crypto for FPGA IP Protection. In *Proceedings of the 2007 International Conference on Field Programmable Logic and Applications — FPL 2007, Amsterdam, The Netherlands*, pages 189–195. IEEE, August 27-30, 2007.
16. P. Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Sixth USENIX Workshop on Smartcard Technology Proceedings*, pages 77–89, San Jose, California, July 1996. Available at `http://www.cs.cornell.edu/people/clarkson/secdg/papers.sp06/secure_deletion.pdf`.
17. P. Gutmann. Data remanence in semiconductor devices. In *10th USENIX Security Symposium*, pages 39–54, August 2001. Available at `http://www.cryptoapps.com/~peter/usenix01.pdf`.
18. T. Ignatenko, G.J. Schrijen, B. Skoric, P. Tuyls, and F. Willems. Estimating the Secrecy-Rate of Physical Unclonable Functions with the Context-Tree Weighting Method. In *IEEE International Symposium on Information Theory*, pages 499–503, Seattle, USA, July 2006.
19. C. S. Jutla. Encryption Modes with Almost Free Message Integrity. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 529–544. Springer, May 6-10, 2001.
20. A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. Watermarking techniques for intellectual property protection. In *Design Automation Conference — DAC '98*, pages 776–781, New York, NY, USA, 1998. ACM Press.

21. T. Kean. Cryptographic rights management of FPGA intellectual property cores. In *ACM/SIGDA tenth international symposium on Field-programmable gate arrays — FPGA 2002*, pages 113–118, 2002.
22. H. Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume 2139 of *LNCS*, pages 310–331. Springer, August 19-23, 2001.
23. H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. Internet RFC 2104, February 1997. Available at `http://www-cse.ucsd.edu/~mihir/papers/rfc2104.txt`.
24. D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, October 2005.
25. J.-P. M. G. Linnartz and P. Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In J. Kittler and M. S. Nixon, editors, *Audio-and Video-Based Biometrie Person Authentication — AVBPA 2003*, volume 2688 of *LNCS*, pages 393–402. Springer, June 9-11, 2003.
26. R. S. Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, March 2001. Available at `http://pubs.media.mit.edu/pubs/papers/01.03.pappuphd.powf.pdf`.
27. R. S. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297(6):2026–2030, 2002. Available at `http://web.media.mit.edu/~brecht/papers/02.PapEA.powf.pdf`.
28. W. W. Peterson and E. J. Weldon, Jr. *Error-Correcting Codes*. The MIT Press, second edition, 1972.
29. P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.
30. E. Seevinck, F. J. List, and J. Lohstroh. Static-Noise Margin Analysis of MOS SRAM Cells. *IEEE Journal of Solid-State Circuits*, 22(5):748–754, Oct 1987.
31. E. Simpson and P. Schaumont. Offline Hardware/Software Authentication for Reconfigurable Platforms. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *LNCS*, pages 311–323. Springer, October 10-13, 2006.
32. B. Skoric, P. Tuyls, and W. Ophey. Robust Key Extraction from Physical Uncloneable Functions. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security — ACNS 2005*, volume 3531 of *LNCS*, pages 407–422, June 7-10, 2005.
33. S. P. Skorobogatov. Low temperature data remanence in static RAM. Technical Report 536, University of Cambridge, Computer Laboratory, June 2002.
34. Y. Su, J. Holleman, and B. Otis. A 1.6pJ/bit 96% Stable Chip-ID Generating Cicuit using Process Variations. In *ISSCC '07: IEEE International Solid-State Circuits Conference*, pages 406–408, Washington, DC, USA, 2007. IEEE Computer Society.
35. P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and R. Wolters. Read-Proof Hardware from Protective Coatings. In *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 369–383. Springer, October 10-13, 2006.
36. D. Whiting, R. Housley, and N. Ferguson. Counter with CBC-MAC (CCM). NIST Proposed Mode of Operation, June 2002. Available at `http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/ccm/ccm.pdf`.