Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions

Frederik Armknecht[1], Roel Maes[2], Ahmad-Reza Sadeghi[1], Berk Sunar[3], and Pim Tuyls[2,4]

[1] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
[2] ESAT/COSIC and IBBT, Catholic University of Leuven, Belgium
[3] Cryptography & Information Security, WPI, MA USA
[4] Intrinsic ID, Eindhoven, the Netherlands

Abstract. Physical attacks on cryptographic implementations and devices have become crucial. In this context a recent line of research on a new class of side-channel attacks, called *memory attacks*, has received increasingly more attention. These attacks allow an adversary to measure a significant fraction of secret key bits directly from memory, independent of any computational side-channels.

Physically Unclonable Functions (PUFs) represent a promising new technology that allows to store secrets in a tamper-evident and unclonable manner. PUFs enjoy their security from physical structures at submicron level and are very useful primitives to protect against memory attacks.

In this paper we aim at making the first step towards combining and binding algorithmic properties of cryptographic schemes with physical structure of the underlying hardware by means of PUFs. We introduce a new cryptographic primitive based on PUFs, which we call PUF-PRFs. These primitives can be used as a source of randomness like pseudorandom functions (PRFs). We construct a block cipher based on PUF-PRFs that allows simultaneous protection against algorithmic and physical attackers, in particular against memory attacks. While PUF-PRFs in general differ in some aspects from traditional PRFs, we show a concrete instantiation based on established SRAM technology that closes these gaps.

1 Introduction

Modern cryptography provides a variety of tools and methodologies to analyze and to prove the security of cryptographic schemes such as in [7,8,6,9]. These proofs always start from a particular setting with a well-defined adversary model and security notion. The vast majority of these proofs assume a *black box* model: the attacker knows all details of the used algorithms and protocols but has no knowledge of or access to the secrets of the participants, nor can he observe any internal computations. The idealized model allows one to derive security guarantees and gain valuable insights.

M. Matsui (Ed.): ASIACRYPT 2009, LNCS 5912, pp. 685–702, 2009.

However, as soon as this basic assumption fails most security guarantees are off and a new open field of study arises. In cryptographic implementations long-term secret keys are typically stored by configuring a non-volatile memory such as ROM, EEPROM, flash, anti-fuses, poly or e-fuses into a particular state. Computations on these secrets are performed by driving electrical signals from one register to the next and transforming them using combinatorial circuits consisting of digital gates. Side-channel attacks pick up physically leaked key-dependent information from internal computations, e.g. by observing consumed power [27] or emitted radiation [1], making many straightforward algorithms and implementations insecure. It is clear that from an electronic hardware point of view, security is viewed differently, see e.g. [30,49,48,44].

Even when no computation is performed, stored secret bits may leak. For instance, in [43] it was shown that data can be recovered from flash memory even after a number of erasures. By decapsulating the chip and using scanning electron microscopes or transmission electron microscopes the states of anti-fuses and flash can be made visible. Similarly, a typical computer memory is not erased when its power is turned off giving rise to so-called cold-boot attacks [22]. More radical approaches such as opening up an integrated circuit and probing metal wires or scanning non-volatile memories with advanced microscopes or lasers generally lead to a security breach of an algorithm, often immediately revealing an internally stored secret [43].

Given this observation, it becomes natural to investigate security models with the basic assumption: *memory leaks information on the secret key.* Consequently, a recently started line of work has investigated the use of new cryptographic primitives that are less vulnerable to leakage of key bits [2,36]. These works establish security by adapting public-key algorithms to remain secure even after leaking a limited number of key bits. However, no security guarantees can be given when the leakage exceeds a certain threshold, e.g. when the whole non-volatile memory is compromised. Furthermore, they do not provide a solution for the traditional settings, e.g. for securing symmetric encryption schemes.

Here we explore an alternative approach: Instead of making another attempt to solve the problem in an algorithmic manner, we base our solution on a new physical primitive. So-called Physically Unclonable Functions (PUFs) provide a new cryptographic primitive able to store secrets in a non-volatile but highly secure manner. When embedded into an integrated circuit, PUFs are able to use the deep submicron physical uniqueness of the device as a source of randomness [15,14,20,47]. Since this randomness stems from the uncontrollable subtleties of the manufacturing process, rather than from hard-wired bits, it is practically infeasible to externally measure these values during a physical attack. Moreover, any attempt to open up the PUF in order to observe its internals will with overwhelming probability alter these variables and change or even destroy the PUF [47].

In this paper, we take advantage of the useful properties of PUFs to build an encryption scheme resilient against memory leakage adversaries as defined in [2]. We construct a block cipher that explicitly makes use of the algorithmic and

physical properties of PUFs to protect against physical *and* algorithmic attacks at the same time. Other protection mechanisms against physical attacks require either additional algorithmic effort, e.g. [24,34,45,39], on the schemes or separate (possibly expensive) hardware measures.

Our encryption scheme can particularly be used for applications such as secure storage of data on untrusted storage (e.g., harddisk) where (i) no storage of secrets for encryption/decryption is needed and keys are only re-generated when needed, (ii) copying the token is infeasible (unclonability), (iii) temporary unauthorized access to the token will reveal data to the adversary but not the key, or (iv) no non-volatile memory is available.

Contribution. Our contributions are as follows:

A new cryptographic primitive: PUF-PRF. We place the PUFs at the core of a pseudorandom function (PRF) construction that meets well-defined properties. We provide a formal model for this new primitive that we refer to as PUF-PRFs. PRFs [19] are fundamental primitives in cryptography and have many applications, e.g. see [18,32,33].

A PUF-PRF-based provably secure block cipher. One problem with our PUF-PRF construction is that it requires some additional helper data that inevitably leaks some internal information. Hence, PUF-PRFs cannot serve as a direct replacement for PRFs. However, we present a provably secure block cipher based on PUF-PRFs that remains secure despite the information leakage. Furthermore, no secret key needs to be stored, protecting the scheme against memory leakage attacks. The tight integration of PUF-PRFs into the cryptographic construction improves the tamper-resilience of the overall design. Any attempt at accessing the internals of the device will result in a change of the PUF-PRF. Hence, no costly error detection networks or alternative anti-tampering technologies are needed. The unclonability and tamper-resilience properties of the underlying PUFs allow for elegant and cost-effective solutions to specific applications such as software protection or device encryption.

An improved and practical PUF-PRF construction. Although the information leakage through helper data is unavoidable in the general case, the concrete case might allow for more efficient and secure constructions. We introduce SRAM-PRFs, based on so-called SRAM PUFs, which are similar to the general PUF-PRFs but where it can be shown that no information is leaked through the helper data if run in an appropriate mode of operation. Hence, SRAM-PRFs are in all practical views a physical realization of expanding PRFs.

Organization. This paper is organized as follows. First, we give an overview of related work in Section 2. In Section 3, we define and justify the considered attacker model. In Section 4, we introduce a formal model for PUFs. Based on this, we define in Section 5 a new cryptographic primitive, termed PUF-PRFs. Furthermore, we present a provably secure block cipher based on PUF-PRFs that is secure despite the information leakage through helper data. In Section 6,

we explain for the concrete case of SRAM PUFs an improved construction that shares the same benefits like general PUF-PRFs but where it can be argued that the helper data does not leak any information. Finally, in Section 7 we present the conclusions.

2 Related Work

In recent years numerous results in the field of physical attacks emerged showing that the classical black box model is overly optimistic, see e.g. [42,43,3,28,27]. Due to a number of physical leakage channels, the adversary often learns (part of) a stored secret or is able to observe some intermediate results of the private computations. These observations give him a powerful advantage that often breaks the security of the entire scheme. To cope with this reality, a number of new theoretic adversary models were proposed, incorporating possible physical leakage of this kind. Ishai et al. [24] model an adversary which is able to probe, i.e. eavesdrop, a number of lines carrying intermediate results in a private circuit, and show how to create a secure primitive within this computational leakage model. Later, generalizations such as Physically Observable Cryptography proposed by Micali et al. [34] investigate the situation where only computation leaks information while assuming leak-proof secret storages. Recently, Pietrzak [13,39] and Standaert et al. [45] put forward some new models and constructions taking physical side-channel leakage into account.

Complementary to the computation leakage attacks, another line of work explored memory leakage attacks: an adversary learns a fraction of a stored secret [2,36]. In [2] Akavia et al introduced a more realistic model that considers the security against a wide class of side-channel attacks when a function of the secret key bits is leaked. Akavia et al further showed that Regev's lattice-based scheme [41] is resilient to key leakage. More recently Naor et al [36] proposed a generic construction for a public-key encryption scheme that is resilient to key leakage. Although all these papers present strong results from a theoretical security point of view, they are often much too expensive to implement on an integrated circuit (IC), e.g. the size of private circuits in [24] blows up with $O(n^2)$ where n denotes the number of probings by the adversary. Moreover, almost all of these proposals make use of public-key crypto primitives, which introduce a significant overhead in systems where symmetric encryption is desired for improved efficiency.

Besides the information leakage attacks mentioned above, another important field of studies are tampering attacks. Numerous countermeasures have been discussed, e.g., use of a protective coating layer [40] or the application of error detection codes (EDCs) [25,16]. Observe that limitations and benefits of tamper-proof hardware have likewise been theoretically investigated in a series of works [17,26,35,10].

3 Memory Attacks

In this work we consider an extension of memory attacks as introduced by Akavia et. al. [2] where the attacker can extract a bounded number of bits of a stored

secret. The model allows for covering a large variety of different memory attacks, e.g., cold boot attacks described in [22]. However, this general model might not adequately capture certain concrete scenarios. For example, feature sizes on ICs have shrunk to nanometer levels and probing such fine metal wires is even for high-end IC manufacturers a difficult task. During a cryptographic computation a secret state is (temporarily) stored in volatile memory (e.g. in registers and flip-flops). In a typical IC, these structures are relatively small compared to the rest of the circuit, making them very hard to locate and scan properly. Thus, applying these attacks is usually significantly physically more involved for the case of embedded ICs than for the non-embedded PC setting where additional measures to access the memory exist, e.g., through software and networks.

On the other hand, storing long-term secrets, such as private keys, requires non-volatile memory, i.e. memory that sustains its state while the embedding device is powered off. Implementation details of such memories like ROM, EEPROM, flash, anti-fuses, poly or e-fuses and recent results on physical attacks such as [43] indicate that physically attacking non-volatile memory is *much* easier than attacking register files or probing internal busses on recent ICs, making non-volatile memory effectively the weak link in many security implementations.

Motivated by these observations, we consider the following attacker model in this work:

Definition 1 (Non-volatile Memory Attacker). *Let $\alpha : \mathbb{N} \to \mathbb{N}$ be a function with $\alpha(n) \leq n$ for all $n \in \mathbb{N}$, and let S be a secret stored in non-volatile memory. A* α-non-volatile memory attacker *can access an oracle $\mathcal{O}$ that takes as input adaptively chosen a polynomial-size circuits h_i and outputs $h_i(S)$ under the condition that the total number of bits that A gets as a result of oracle queries is at most $\alpha(|S|)$.*

The attacker is called a full non-volatile memory attacker *if $\alpha = \mathrm{id}$, that is the attacker can extract the whole content of the non-volatile memory.*

Obviously, protection against full non-volatile memory attackers is only possible if no long-term secrets are stored within non-volatile memory. One obvious approach is to require a user password before each invocation. However, this reduces usability and is probably subject to password attacks. In this paper, we use another approach and make use of a physical primitive called Physically Unclonable Function (PUF). PUFs allow to *intrinsicly* store permanent secrets which are, according to current state of knowledge, not accessible to a non-volatile attacker.

4 Physically Unclonable Functions

In this section, we introduce a formal model for Physically Unclonable Functions (PUFs). We start with some basic definitions. For a probability distribution $\mathbb{D}$, the expression $x \leftarrow \mathbb{D}$ denotes the event that x has been sampled according to $\mathbb{D}$. For a set S, $x \xleftarrow{*} S$ means that x has been sampled uniformly random from S. For $m \geq 1$, we denote by $\mathbb{U}_m$ the uniform distribution on $\{0,1\}^m$. The *min-entropy*

$H_\infty(\mathbb{D})$ of a distribution $\mathbb{D}$ is defined by $H_\infty(\mathbb{D}) \stackrel{\text{def}}{=} -\log_2(\max_x Pr[x \leftarrow \mathbb{D}])$. Min-entropy can be viewed as the "worst-case" entropy in a random variable sampled according to $\mathbb{D}$ [11] and specifies how many nearly uniform random bits can be extracted from it.

A *distinguisher* $\mathcal{D}$ is a (possibly probabilistic) algorithm that aims for distinguishing between two different distributions $\mathbb{D}$ and $\mathbb{D}'$. More precisely, $\mathcal{D}$ receives some values (which may depend on adaptively chosen inputs by $\mathcal{D}$) and outputs a value from $\{0,1\}$. The *advantage* of $\mathcal{D}$ is defined by $\text{Adv}(\mathcal{D}) \stackrel{\text{def}}{=} |Pr[1 \leftarrow \mathcal{D}|\mathbb{D}] - Pr[1 \leftarrow \mathcal{D}|\mathbb{D}']|$. Furthermore, we define the advantage of distinguishing between $\mathbb{D}$ and $\mathbb{D}'$ as $\max_{\mathcal{D}} \text{Adv}(\mathcal{D})$.

In a nutshell, PUFs are physical mechanisms that accept challenges and return responses, that is behaving like functions. The main properties of PUFs that are important in the context of cryptographic applications are *noise* (same challenge can lead to different (but close) responses), *non-uniform distribution* (the distribution of the responses is usually non-uniform), *independence* (two different PUFs show completely independent behavior), *unclonability* (no efficient process is known that allows for physically cloning PUFs), and *tamper evidence* (physically tampering with a PUF will most likely destroy its physical structure, making it unusable, or turn it into a new PUF). We want to emphasize that the properties above are of a physical nature and hence are very hard to prove in the rigorous mathematical sense. However, they are based on experiments conducted worldwide and reflect the current *assumptions* and observations regarding PUFs, e.g., see [47]. We first provide a formal definition for noisy functions before we give a definition for PUFs.

Definition 2 (Noisy functions). *For three positive integers $\ell, m, \delta \in \mathbb{N}$ with $0 \leq \delta \leq m$, a (ℓ, m, δ)*-noisy function *f^* is a probabilistic algorithm which accepts inputs (challenges) $x \in \{0,1\}^\ell$ and generates outputs (responses) $y \in \{0,1\}^m$ such that the Hamming distance between two outputs to the same input is at most δ. In a similar manner, we define a (ℓ, m, δ)*-noisy family of functions *to be a set of (ℓ, m, δ)-noisy functions.*

Definition 3 (Physically Unclonable Functions). *A $(\ell, m, \delta; q_{puf}, \epsilon_{puf})$*-family of PUFs *$\mathcal{P}$ is a set of physical realizations of a family of probabilistic algorithms that fulfills the following algorithmic and physical properties.*

Algorithmic properties

- ***Noise:*** *$\mathcal{P}$ is a (ℓ, m, δ)-noisy family of functions with $\delta < \frac{m}{2}$*
- ***Non-uniform output and independence:*** *There exists a distribution $\mathbb{D}$ on $\{0,1\}^m$ such that for any input $x \in \{0,1\}^\ell$, the following two distributions on $(\{0,1\}^m)^{q_{puf}}$ can be distinguished with advantage at most ϵ_{puf}.*
  1. *$(\Pi_1(x), \ldots, \Pi_{q_{puf}}(x))$ for adaptively chosen $\Pi_i \in \mathcal{P}$.*
  2. *$(y_1, \ldots, y_{q_{puf}})$ with $y_i \leftarrow \mathbb{D}$.*

  In order to have a practically useful PUF, it should be that $q_{puf} \approx |\mathcal{P}|$, ϵ_{puf} is negligible and $H_\infty(\mathbb{D}) > 0$.

Physical properties

- ***Unclonability:*** *No efficient technique is known to physically clone any member $\Pi \in \mathcal{P}$.*
- ***Tamper evidence:*** *For any PUF $\Pi \in \mathcal{P}$, any attempt to externally obtain its responses or parameters, e.g. by means of a physical attack, will significantly alter its functionality or destroy it.*

A number of constructions for PUFs have been implemented and most of them have been experimentally verified to meet the properties of this theoretical definition. For more details we refer to the literature, e.g. [47,20,29,31,46]. One important observation we make is that a number of PUF implementations can be efficiently implemented on an integrated circuit, e.g. SRAM PUFs [20]. Their challenge-response behavior can hence be easily integrated with a chip's digital functionality.

Remark 1. Due to their physical properties, PUFs became an interesting building block for protecting against full non-volatile memory attackers. The basic idea is to use a PUF for *implicitly* storing a secret: instead of putting a secret directly into non-volatile memory, it is derived from the PUF responses during run time [20,21].

5 Encrypting with PUFs: A Theoretical Construction

In the previous section, we explained how to use PUFs for protecting any cryptographic scheme against full non-volatile memory attackers (see Remark 1). In the remainder of the paper, we go one step further and explore how to use PUFs for protecting against algorithmic attackers *in addition*. For this purpose, we discuss how to use PUFs as a source of reproducible pseudorandomness. This approach is motivated by the observation that certain PUFs behave to some extent like unpredictable functions. This will allow for constructing (somewhat weaker) physical instantiations of (weak) pseudorandom functions.

5.1 PUF-(w)PRFs

Pseudorandom functions (PRFs) [19] are important cryptographic primitives with various applications (see, e.g., [18,32,33]). We recall their defininition.

Definition 4 ((Weak) Pseudorandom Functions). *Consider a family of functions $\mathcal{F}$ with input domain $\{0,1\}^\ell$ and output domain $\{0,1\}^m$. We say that $\mathcal{F}$ is $(q_{prf}, \epsilon_{prf})$-pseudorandom in respect to a distribution $\tilde{\mathbb{D}}$ on $\{0,1\}^m$, if the advantage to distinguish between the following two distributions for adaptively chosen pairwise distinct inputs $x_1, \ldots, x_{q_{prf}}$ is at most ϵ_{prf}:*

- $y_i = f(x_i)$ *where* $f \xleftarrow{*} \mathcal{F}$
- $y_i \leftarrow \tilde{\mathbb{D}}$

$\mathcal{F}$ is called weakly pseudorandom *if the inputs are not chosen by the distinguisher, but uniformly random sampled from $\{0,1\}^\ell$ (still under the condition of being pairwise distinct).*

$\mathcal{F}$ is called $(q_{prf}, \epsilon_{prf})$-(weakly)-pseudorandom if it is $(q_{prf}, \epsilon_{prf})$-(weakly)-pseudorandom with respect to the uniform distribution $\tilde{\mathbb{D}} = \mathbb{U}_m$.

Remark 2. This definition differs in several aspects slightly from the original definition of pseudorandom functions, e.g., [5,4]. First, specifying the output distribution $\tilde{\mathbb{D}}$ allows for covering families of functions which have a non-uniform output distribution, e.g., PUFs. The original case, as stated in the definition, is $\tilde{\mathbb{D}} = \mathbb{U}_m$.

Second, the requirement of pairwise distinct inputs x_i has been introduced to deal with noisy functions where the same input can lead to different outputs. By disallowing multiple queries on the same input, we do not need to model the noise distribution, which is sometimes hard to characterize in practice. Furthermore, in the case of non-noisy (weak) pseudorandom functions, an attacker gains no advantage by querying the same input more than once. Hence, the requirement does not limit the attacker in the non-noisy case.

Observe that the "non-uniform output and independence" assumption on PUFs (as defined in Definition 3) does not automatically imply (weak) pseudorandomness. The first considers the unpredictability of the response to a specific challenge after making queries to *several different* PUFs while the latter considers the unpredictability of the response to a challenge after making queries to the same PUF.

Obviously, the main obstacle is to convert noisy non-uniform inputs into reliably reproducible, uniformly distributed random strings. For this purpose, we make use of an established tool in cryptography, i.e. *fuzzy extractors* (FE) [12]:

Definition 5 (Fuzzy Extractor). *A $(m, n, \delta; \mu_{FE}, \epsilon_{FE})$-*fuzzy extractor *$\mathsf{E}$ is a pair of randomized procedures, "generate" $\mathsf{Gen} : \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^*$ and "reproduce" $\mathsf{Rep} : \{0,1\}^m \times \{0,1\}^* \to \{0,1\}^n$.*

The correctness property guarantees that for $(z, \omega) \leftarrow \mathsf{Gen}(y)$ and $y' \in \{0,1\}^m$ with $\mathsf{dist}(y, y') \leq \delta$, then $\mathsf{Rep}(y', \omega) = z$. If $\mathsf{dist}(y, y') > \delta$, then no guarantee is provided about the output of Rep.

The security property guarantees that for any distribution $\mathbb{D}$ on $\{0,1\}^m$ of min-entropy μ_{FE}, the string z is nearly uniform even for those who observe ω: if $(z, \omega) \leftarrow \mathsf{Gen}(\mathbb{D})$, then it holds that $\mathrm{SD}((z, \omega), (\mathbb{U}_n, \omega)) \leq \epsilon_{FE}$.

PUFs are most commonly used in combination with fuzzy extractor constructions based on error-correcting codes and universal hash functions. In this case, the helper data consists of a code-offset, which is of the same length as the PUF output, and the seed for the hash function, which is in the order of 100 bits and can often be reused for all outputs.

Theorem 1 (Pseudorandomness of PUF-FE-composition). *Let $\mathcal{P}$ be a $(\ell, m, \delta; q_{puf}, \epsilon_{puf})$-family of PUFs which are $(q_{prf}, \epsilon_{prf})$-pseudorandom with respect to some distribution $\mathbb{D}$. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Rep})$ be an $(m, n, \delta; H_\infty(\mathbb{D}), \epsilon_{FE})$*

fuzzy extractor. The advantage of any distinguisher that adaptively chooses pairwise distinct inputs $x_1, \ldots, x_{q_{prf}}$ *and receives outputs* $(z_1, \omega_1), \ldots, (z_{q_{prf}}, \omega_{q_{prf}})$ *to distinguish the following two distributions is at most* $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE}$*:*

- $(z_i, \omega_i) = \mathsf{Gen}(\Pi(x_i))$ *where* $\Pi \xleftarrow{*} \mathcal{P}$
- (z_i, ω_i) *where* $z_i \leftarrow \mathbb{U}_n$, $(z_i', \omega_i) = \mathsf{Gen}(\Pi(x_i))$ *and* $\Pi \xleftarrow{*} \mathcal{P}$

The analogous result holds if $\mathcal{P}$ *is* $(q_{prf}, \epsilon_{prf})$*-weak-pseudorandom and if the challenges* x_i *are sampled uniformly random (instead of being adaptively selected), still under the condition of being pairwise distinct.*

Proof. We introduce an intermediate case, named case 1', where $(z_i, \omega_i) = \mathsf{Gen}(y_i)$ with $y_i \leftarrow \mathbb{D}$ and $\Pi \xleftarrow{*} \mathcal{P}$. Any distinguisher between case 1 and case 1' can be turned into a distinguisher that distinguishes between PUF outputs and random samples according to $\mathbb{D}$. Hence, the advantage is at most ϵ_{prf} by assumption. Furthermore, by the usual hybrid argument and the security property of fuzzy extractors, case 1' and case 2 can be distinguished with advantage of at most $q_{prf} \cdot \epsilon_{FE}$. □

Definition 6 (PUF-(w)PRFs). *Consider a family of (weakly)-pseudorandom PUFs* $\mathcal{P}$ *and a fuzzy extractor* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Rep})$ *(where the parameters are as described in Theorem 1). A family of PUF-(w)PRFs is a set of pairs of randomized procedures, called* generation *and* reproduction*. The generation function* $\mathsf{Gen} \circ \Pi$ *for some PUF* $\Pi \in \mathcal{P}$ *takes as input* $x \in \{0,1\}^\ell$ *outputs* $(z, \omega_x) \stackrel{\text{def}}{=} \mathsf{Gen}(\Pi(x)) \in \{0,1\}^n \times \{0,1\}^*$*, while the reproduction function* $\mathsf{Rep} \circ \Pi$ *takes* $(x, \omega_x) \in \{0,1\}^\ell \times \in \{0,1\}^*$ *as input and reproduces the value* $z = \mathsf{Rep}(\Pi(x), \omega_x)$*.*

Theorem 1 actually shows that PUF-(w)PRFs and "traditional" (w)PRFs have in common that (part of) the output cannot be distinguished from uniformly random values. One might be tempted to plug in PUF-(w)PRFs wherever PRFs are required. Unfortunately, things are not that simple since the information saved in the helper data is also needed for correct execution. It is a known fact that the helper data of a fuzzy extractor *always* leaks some information about the input, e.g., see [23]. Hence, extra attention must be paid when deploying PUF-PRFs in cryptographic schemes. In the following section, we describe an encryption scheme that achieves real-or-random security *although* the helper data is made public.

5.2 A Luby-Rackoff Cipher Based on PUF-wPRFs

A straightforward approach for using PUF-wPRFs against full non-volatile memory attackers would be to use them for key derivation where the key is afterwards used in some encryption scheme. However, in this construction PUF-wPRFs would ensure security against non-volatile memory attackers only while the security of the encryption scheme would need to be shown separately. In the following, we present a construction that simultaneously protects against algorithmic and physical attacks while the security in both cases can be deduced to PUF-wPRF properties.

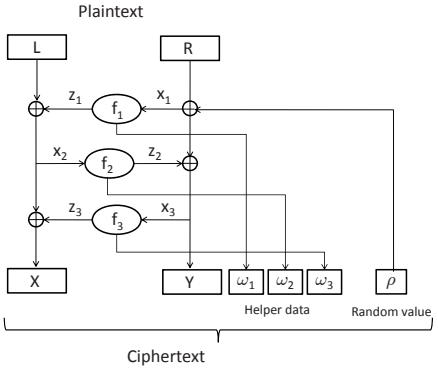

Fig. 1. A randomized 3-round Luby-Rackoff-cipher based on PUF-PRFs

One of the most important results with respect to PRFs was developed by Luby and Rackoff in [33]. They showed how to construct pseudorandom permutations from PRFs. Briefly summarized, a pseudorandom permutation (PRP) is a PRF that is a permutation as well. PRPs can be seen as an idealization of block ciphers. Consequently, the Luby-Rackoff construction is often termed as Luby-Rackoff cipher.

Unfortunately, the Luby-Rackoff result does not automatically apply to the case of PUF-PRFs. As explained in the previous section, PUF-(w)PRFs differ from (w)PRFs as they additionally need some helper data for correct execution. First, it is unclear if and how the existence and necessity of helper data would fit into the established concept of PRPs. Second, an attacker might adaptively choose plaintexts to force internal collisions and use the information leakage of the helper data for checking for these events.

Nonetheless, we can show that a Luby-Rackoff cipher based on PUF-wPRFs also yields a secure block cipher. For this purpose, we consider the set of concrete security notions for symmetric encryption schemes that has been presented and discussed in [4]. More precisely, we prove that a randomized version of a 3-round Luby-Rackoff cipher based on PUF-PRFs fulfills real-or-random indistinguishability against a chosen-plaintext attacker.

In a nutshell, a real-or-random attacker adaptively chooses plaintexts and hands them to an encryption oracle. This oracle either encrypts the received plaintexts (real case) or some random plaintexts (random case). The encryptions are given back to the attacker. Her task is to distinguish between both cases. The scheme is real-or-random indistinguishable if the advantage of winning the game is negligible (in some security parameter). Next, we first define the considered block cipher and prove its security afterwards.

Definition 7 (3-round PUF-wPRF-based Luby-Rackoff cipher). *Let $\mathcal{F}$ denote a family of PUF-wPRFs with input and output length n. The 3-round PUF-PRF-based Luby-Rackoff cipher $\mathcal{E}^{\mathcal{F}}$ uses three different PUF-wPRFs $f_i \in$*

$\mathcal{F}$, $i = 1, 2, 3$, as round functions. The working principle is very similar to the original Luby-Rackoff cipher and is displayed in figure 1. The main differences are twofold. First, at the beginning some uniformly random value $\rho \in \{0,1\}^\ell$ is chosen to randomize the right part R of the plaintext. Second, the round functions are PUF-wPRFs that generate two outputs: z_i and ω_i.

The ciphertext is $(X, Y, \omega_1, \omega_2, \omega_3, \rho)$. Decryption works similar to the case of the "traditional" Luby-Rackoff cipher where the helper data ω_i is used together with the Rep *procedure for reconstructing the output z_i of the PUF-PRF f_i and the value ρ to "derandomize" the input to the first round function f_1.*

Since there is no digital secret stored in non-volatile memory, even a full non-volatile memory attacker has no advantage in breaking this scheme. Although this makes encrypting digital communication between two different parties impossible, various applications are imaginable, e.g., for encrypting data stored in untrusted or public storage.

Theorem 2. *Let $\mathcal{E}^{\mathcal{F}}$ be the encryption scheme defined in Definition 7 using a family $\mathcal{F}$ of PUF-wPRFs (with parameters as specified in Theorem 1). Then, the advantage of a real-or-random attacker making up to q_{prf} queries is at most $4\epsilon_{prf} + 2q_{prf} \cdot \epsilon_{FE} + 2 \cdot \frac{{q_{prf}}^2}{2^n}$.*

Proof. Let $\{(L^{(i)}, R^{(i)}\}_{i=1,\ldots,q_{prf}}$ denote the sequence of the adaptively chosen plaintexts and $x_j^{(i)}$, $z_j^{(i)}$ be the respective inputs and outputs to round function f_j, and $\rho^{(i)}$ the randomly chosen values. We show the claim by defining a sequence of games and estimating the advantages of distinguishing between them. Let the real game be the scenario that the distinguisher receives the encryptions of the plaintext she did choose.

In game 1, the outputs $z_1^{(i)}$ of the first round function f_1 are replaced by some uniformly random values $\tilde{z}_1^{(i)} \xleftarrow{*} \{0,1\}^n$. Under the assumption that the values $x_1^{(i)}$ are pairwise distinct, the advantage to distinguish between both cases is at most $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE}$ according to Theorem 1. Furthermore, as the values $\rho^{(i)}$ are uniformly random, the probability of a collision in the values $x_1^{(i)}$ is at most $\frac{{q_{prf}}^2}{2^n}$. As a consequence, the advantage to distinguish between the real game and game 1 is upper bounded by $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE} + \frac{{q_{prf}}^2}{2^n}$.

Game 2 is defined like game 1 where now the inputs $x_1^{(i)}$ to the first round function f_1 are randomized to $\tilde{x}_1^{(i)} \xleftarrow{*} \{0,1\}^n$. Observe that the values $x_1^{(i)}$ are used in two different contexts: i) for computing the right part of the ciphertext (by XORing with the output of the second round function) and ii) as input to the first round function. Regarding i), observe that the outputs of the second round function are independent of the values $x_1^{(i)}$ as the values $\tilde{z}_1^{(i)}$ (and hence the inputs to f_2) are uniformly random by definition and that the values $x_1^{(i)}$ are independent of the plaintext (because of $\rho^{(i)}$). Hence, i) and ii) represent two independent features, possibly allowing for distinguishing between game 1 and game 2, and hence can be examined separately.

The advantage of distinguishing between games 1 and 2 based on i) is equivalent to deciding whether the values $R^{(i)} \oplus \rho^{(i)} \oplus Y^{(i)}$ are uniformly random or belong to the outputs of the second round function. With the same arguments as above, the advantage is upper bounded by $\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE} + \frac{{q_{prf}}^2}{2^n}$.

The advantage of distinguishing between game 1 and game 2 based on ii) is at most the advantage of distinguishing $(\Pi_1(x_1^{(1)}), \ldots, \Pi_1(x_{q_{prf}}^{(1)}))$ from $(\Pi_1(\tilde{x}_1^{(1)}), \ldots, \Pi_1(\tilde{x}_{q_{prf}}^{(1)}))$ where Π_1 denotes the PUF used in f_1. By the definition of wPRFs (Definition 4), the advantage of distinguishing $(\Pi_1(x_1^{(1)}), \ldots, \Pi_1(x_{q_{prf}}^{(1)}))$ from $(y_1, \ldots, y_{q_{prf}})$ where $y_i \leftarrow \tilde{\mathbb{D}}$ and $\tilde{\mathbb{D}}$ being an appropriate distribution is at most ϵ_{prf}. Actually, the same holds for $(\Pi_1(\tilde{x}_1^{(1)}), \ldots, \Pi_1(\tilde{x}_{q_{prf}}^{(1)}))$ (the fact that the values $\tilde{x}_i^{(1)}$ are unknown cannot increase the advantage). Hence, by the triangular inequality, it follows that the advantage regarding ii) is at most $2\epsilon_{prf}$. In total, the advantage to distinguish between game 1 and game 2 is less than or equal to $3\epsilon_{prf} + q_{prf} \cdot \epsilon_{FE} + \frac{{q_{prf}}^2}{2^n}$.

Finally, observe that it is indistinguishable whether $x_1^{(i)}$ or $R^{(i)}$ is randomized and likewise whether $z_1^{(i)}$ or $L^{(i)}$. Hence, game 2 is indistinguishable from the random game where the plaintexts are randomized. Summing up, the advantage of a real-or-random attacker is at most $4\epsilon_{prf} + 2q_{prf} \cdot \epsilon_{FE} + 2 \cdot \frac{{q_{prf}}^2}{2^n}$. □

6 SRAM PRFs

In the previous section, we showed that secure cryptographic schemes are possible even if helper data is used that leaks information. In this section, we show that in the concrete case, information leakage through helper data can be avoided completely. We illustrate this approach on SRAM PUFs that were originally introduced and experimentally verified in [20]. In respect to our modeling, an SRAM PUF is a realization of a $(\ell, m, \delta; q_{puf}, \epsilon_{puf})$-PUF that is $(2^\ell, 0)$-pseudorandom.

We introduce a new mode of operation that, similarly to the fuzzy extractor approach in the previous section, allows for extracting uniform values from SRAM PUFs in a reproducible way. This approach likewise stores some additional helper data but, as opposed to the case of fuzzy extractors, the helper data does not leak any information on the input. Hence, this construction might be of independent interest for SRAM PUF based applications. The proposed construction is based on two techniques: Temporal Majority Voting and Excluding Dark Bits.

We denote the individual bits of a PUF response as $y = (y_0, \ldots, y_{m-1})$, with $y_i \in \{0, 1\}$. When performing a response measurement on a PUF Π, every bit y_i of the response is determined by a Bernoulli trial. Every y_i has a most likely value $y_i^{(ML)} \in \{0, 1\}$, and a certain probability $p_i < 1/2$ of differing from this value which we define as its *bit error probability*. We denote $y_i^{(k)}$ as the k-th measurement or sample of bit y_i in a number of repeated measurements.

Definition 8 (Temporal Majority Voting (TMV)). *Consider a Bernoulli distributed random bit y_i over $\{0, 1\}$. We define temporal majority voting of y_i*

over N votes, with N an odd positive integer, as a function $TMV_N : \{0,1\}^N \rightarrow \{0,1\}$, that takes as input N different samples of y_i and outputs the most often occurring value in these samples.

We can calculate the error probability $p_{N,i}$ of bit y_i after TMV with N votes as:

$$p_{N,i} \stackrel{\text{def}}{=} Pr\left[\text{TMV}_N\left(y_i^{(0)}, \ldots, y_i^{(N-1)}\right) \neq y_i^{(ML)}\right] = 1 - \text{Bin}_{N,p_i}\left(\frac{N-1}{2}\right) \leq p_i, \tag{1}$$

with Bin_{N,p_i} the cumulative distribution function of the binomial distribution. From Eq. (1) it follows that applying TMV to a bit of a PUF response effectively reduces the error probability from p_i to $p_{N,i}$, with $p_{N,i}$ becoming smaller as N increases. We can determine the number of votes N we need to reach a certain threshold p_T such that $p_{N,i} \leq p_T$, given an initial error probability p_i. It turns out that N rises exponentially as p_i gets close to $1/2$. In practice, we also have to put a limit N_T on the number of votes we can perform, since each vote involves a PUF response measurement. We call the pair (N_T, p_T) a *TMV-threshold*.

Definition 9 (Dark Bit (DB)). *Let (N_T, p_T) be a TMV-threshold. We define a bit y_i to be* dark *with respect to this threshold if $p_{N_T,i} > p_T$.*

TMV alone cannot decrease the bit error probability to acceptable levels (*e.g.* $\leq 10^{-9}$) because of the non-negligible occurrence of dark bits. We use a *bit mask* γ to identify these dark bits in the generation phase, and exclude them during reproduction. Similar to fuzzy extractors, (N_T, p_T)-TMV and DB can be used for generating and reproducing uniform values from SRAM PUFs.

The Gen-procedure takes sufficient measurements of every response bit y_i to make an accurate estimate of its most likely value $y_i^{(ML)}$ and of its error probability p_i. If y_i is dark with respect to (N_T, p_T), then the corresponding bit γ_i in the bit mask $\gamma \in \{0,1\}^m$ is set to 0 and y_i is discarded, otherwise γ_i is set to 1 and y_i is appended to the bit string s. The procedure Gen outputs a helper string $\omega = (\gamma, \sigma)$ and an extracted string $z = \mathsf{Extract}_\sigma(s)$, with $\mathsf{Extract}_\sigma$ a classical strong extractor[1] with seed σ.

The Rep-procedure takes N_T measurements of a response y' and the corresponding helper string $\omega = (\gamma, \sigma)$, with $\gamma \in \{0,1\}^m$ as input. If γ_i contains a 1, then the result of $\text{TMV}_{N_T}\left(y_i'^{(0)}, \ldots, y_i'^{(N_T-1)}\right)$ is appended to a bit string s', otherwise, y_i' is discarded. Rep outputs an extracted string $z' = \mathsf{Extract}_\sigma(s')$.

A strong extractor [37] is a function that is able to generate nearly-uniform outputs from inputs coming from a distribution with limited min-entropy. It ensures that the statistical distance of the extracted output to the uniform distribution is negligible. The required compression rate of $\mathsf{Extract}_\sigma$ depends on the remaining min-entropy μ of the PUF response y after the helper data is observed. We call the above construction a TMV-DB-SRAM-PUF.

[1] See *e.g.* [37,12] for a definition of a strong extractor. Typical seed lengths of strong extractors are in the order of 100 bits, and in most cases the same seed can be reused for all outputs.

Using analogous arguments as in Theorem 1, one can show that the output of a TMV-DB-SRAM-PUF is indistinguishable from random except with negligible advantage. Additionally, in an SRAM PUF, the most likely value of a bit is independent of whether or not the bit is a dark bit, hence no min-entropy on the PUF output is leaked by the bit mask[2]. However, by searching for matching helper strings, an adversary might still be able to find colliding TMV-DB-SRAM-PUF inputs (especially as the input size is small), which can impose a possible security leak. In order to overcome this issue, we present the following way of using a TMV-DB-SRAM-PUF:

Definition 10 (All-at-once mode). *Consider a TMV-DB-SRAM-PUF as described above. We define the* all-at-once *mode of operation to be the pair of procedures* $(\mathsf{Enroll}, \mathsf{Eval})$.

The enrollment procedure Enroll *outputs a helper table* $\Omega \in \{0,1\}^{2^\ell \times *}$ *when executed. The helper table is constructed by running* $\forall x \in \{0,1\}^\ell$ *the generation function* $(\mathsf{Gen} \circ \Pi)(x)$*, and storing the obtained helper data* ω_x *as the* x*-th element in* Ω*, i.e.* $\Omega[x] := \omega_x$.

The evaluation function $\mathsf{Eval} : \{0,1\}^\ell \times \{0,1\}^{2^\ell \times *} \to \{0,1\}^n$ *takes an element* $x \in \{0,1\}^\ell$ *and a helper table* $\Omega \in \{0,1\}^{2^\ell \times *}$ *as inputs and (after internal computation) outputs a value* $\mathsf{Eval}(x, \Omega) = z \in \{0,1\}^n$*, with* $z = (\mathsf{Rep} \circ \Pi)(x, \Omega[x])$.

The Enroll-procedure has to be executed before the Eval-procedure, but it has to be run only once for every PUF. Every invocation of Eval can take the same (public) helper table Ω as one of its inputs. However, in order to conceal exactly which helper string is used, it is important that the Eval-procedure takes Ω *as a whole* as input, and does not just do a look-up of $\Omega[x]$ in a public table Ω. The all-at-once mode prevents an adversary from learning which particular helper string is used during the *internal* computation.

Definition 11 (SRAM-PRF). *An* SRAM-PRF *is a TMV-DB-SRAM-PUF that runs in the all-at-once mode.*

Using the arguments given above we argue that SRAM-PRFs are in all practical views a physical realization of PRFs. Observe that one major drawback of SRAM-PRFs is that the hardware size grows exponentially with the input length. Thus, SRAM-PRFs cannot be used as a concrete instantiation of PUF-PRFs for our construction from Section 5.2. This section rather shows up an alternative approach for constructing cryptographic mechanisms based on PUFs despite of the noise problem. As a possible application of SRAM-PRFs, we discuss an expanding Luby-Rackoff cipher where the round functions are replaced by SRAM-PRFs that take 8-bit challenges as input and produce 120-bit extracted outputs. According to [38], at least 48 rounds are necessary for security reasons.

As an instantiation for the PUF, we take an SRAM PUF with an assumed average bit error probability of 15% and an estimated min-entropy content of 0.95 bit/cell. We use TMV-threshold of $(N_T = 99, p_T = 10^{-9})$. Simulations and

[2] By consequence, also no min-entropy on the PUF *input* is leaked.

experiments on the SRAM PUF show that about 30% of the SRAM cells produce a dark bit with respect to this TMV-threshold. The strong extractor only has to compress by a factor of $\frac{1}{0.95}$, accounting for the limited min-entropy in the PUF response. Hence, $\frac{1}{0.95} \cdot \frac{2^8 \cdot 120}{70\%}$ *bits* $= 5.6$ *kbyte* of SRAM cells is needed to build one SRAM-PRF. Thus, the entire block cipher uses $48 \cdot 5.6$ *kbyte* ≈ 271 *kbyte* of SRAM cells. The helper tables also require 5.6 *kbyte* each.

Implementing 48 SRAM PUFs using a total of 271 *kbyte* of SRAM cells is feasible on recent ICs, and 48 rounds can be evaluated relatively fast. Storing and loading 48 helper tables of 5.6 *kbyte* each is also achievable in practice. Observe that the size depends linearly on the number of rounds. The according parameters for more rounds can be easily derived. Reducing the input size of the SRAM-PRF will yield an even smaller amount of needed SRAM cells and smaller helper tables, but the number of rounds will increase. A time-area trade-off is hence possible.

7 Conclusions

In this paper we propose a leakage-resilient encryption scheme that makes use of Physically Unclonable Functions (PUFs). The core component is a new PUF-based cryptographic primitive, termed PUF-PRF, that is similar to a pseudo-random function (PRF). We showed that PUF-PRFs possess cryptographically useful algorithmic and physical properties that come from the random character of their physical structures.

Of course, any physical model can only approximately describe real life. Although experiments support our model for the considered PUF implementations, more analysis is necessary. In this context it would be interesting to consider other types of PUFs which fit into our model or might be used for other cryptographic applications. Furthermore, a natural continuation of this works would be to explore other cryptographic schemes based of PUF-PRFs, e.g., hash functions or public key encryption.

Acknowledgements

The work described in this paper has been supported [in part] by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. The work of Berk Sunar was supported by the National Science Foundation Cybertrust grant No. CNS-0831416. The work of Roel Maes is funded by IWT-Flanders grant No. 71369 and is in part supported by the IAP Program P6/26 BCRYPT of the Belgian State and K.U.Leuven BOF funding (OT/06/04).

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The em side-channel(s). In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003)

2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
3. Anderson, R.J., Kuhn, M.G.: Low cost attacks on tamper resistant devices. In: Proceedings of the 5th International Workshop on Security Protocols, London, UK, pp. 125–136. Springer, Heidelberg (1998)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS 1997: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS 1997), Washington, DC, USA, p. 394. IEEE Computer Society, Los Alamitos (1997)
5. Bellare, M., Kilian, J., Rogaway, P.: The security of cipher block chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
7. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
8. Bellare, M., Rogaway, P.: Provably secure session key distribution: the three party case. In: STOC 1995: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing, pp. 57–66. ACM, New York (1995)
9. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
10. Chandran, N., Goyal, V., Sahai, A.: New constructions for UC secure computation using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
11. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM J. Comput. 17(2), 230–261 (1988)
12. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
13. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS '08: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science, Washington, DC, USA, pp. 293–302. IEEE Computer Society, Los Alamitos (2008)
14. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Controlled Physical Random Functions. In: Annual Computer Security Applications Conference — ACSAC 2002, Washington, DC, USA, p. 149. IEEE Computer Society, Los Alamitos (2002)
15. Gassend, B., Clarke, D.E., van Dijk, M., Devadas, S.: Silicon physical unknown functions. In: Atluri, V. (ed.) ACM Conference on Computer and Communications Security — CCS 2002, pp. 148–160. ACM, New York (2002)
16. Gaubatz, G., Sunar, B., Karpovsky, M.G.: Non-linear residue codes for robust public-key arithmetic. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 173–184. Springer, Heidelberg (2006)
17. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)

18. Goldreich, O., Goldwasser, S., Micali, S.: On the cryptographic applications of random functions. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 276–288. Springer, Heidelberg (1985)
19. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986)
20. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007)
21. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: Physical Unclonable Functions and Public Key Crypto for FPGA IP Protection. In: International Conference on Field Programmable Logic and Applications — FPL 2007, August 27-30, pp. 189–195. IEEE, Los Alamitos (2007)
22. Alex Halderman, J., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: Cold boot attacks on encryption keys. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 45–60. USENIX Association (2008)
23. Ignatenko, T., Willems, F.: On the security of the XOR-method in biometric authentication systems. In: Twenty-seventh symposium on Information Theory in the Benelux, pp. 197–204 (2006)
24. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
25. Karpovsky, M., Kulikowski, K., Taubin, A.: Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: Proc. Int. Conference on Dependable Systems and Networks (DNS 2004) (July 2004)
26. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
27. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
28. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
29. Kumar, S.S., Guajardo, J., Maes, R., Schrijen, G.-J., Tuyls, P.: The Butterfly PUF: Protecting IP on every FPGA. In: IEEE International Workshop on Hardware-Oriented Security and Trust – HOST 2008, June 9. IEEE, Los Alamitos (2008)
30. Lemke, K.: Embedded security: Physical protection against tampering attacks. In: Lemke, C.P.K., Wolf, M. (eds.) Embedded Security in Cars, ch. 2, pp. 207–217. Springer, Heidelberg (2006)
31. Lim, D., Lee, J.W., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: Extracting secret keys from integrated circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 13(10), 1200–1205 (2005)
32. Luby, M.: Pseudo-randomness and applications. Princeton University Press, Princeton (1996)
33. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. 17(2), 373–386 (1988)
34. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 278–296. Springer, Heidelberg (2004)

35. Moran, T., Segev, G.: David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
36. Naor, M., Segev, G.: Public-Key Cryptosystems Resilient to Key Leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
37. Nisan, N., Zuckerman, D.: More deterministic simulation in logspace. In: STOC 1993: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, pp. 235–244. ACM, New York (1993)
38. Patarin, J., Nachef, V., Berbain, C.: Generic attacks on unbalanced feistel schemes with expanding functions. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 325–341. Springer, Heidelberg (2007)
39. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
40. Posch, R.: Protecting devices by active coating. Journal of Universal Computer Science 4, 652–668 (1998)
41. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 84–93 (2005)
42. Samyde, D., Skorobogatov, S., Anderson, R., Quisquater, J.-J.: On a new way to read data from memory. In: SISW 2002: Proceedings of the First International IEEE Security in Storage Workshop, Washington, DC, USA, p. 65. IEEE Computer Society, Los Alamitos (2002)
43. Skorobogatov, S.P.: Data remanence in flash memory devices. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 339–353. Springer, Heidelberg (2005)
44. Smith, S.W.: Fairy dust, secrets, and the real world [computer security]. IEEE Security and Privacy 1(1), 89–93 (2003)
45. Standaert, F.-X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
46. Edward Suh, G., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation. In: Proceedings of the 44th Design Automation Conference, DAC 2007, San Diego, CA, USA, June 4-8, pp. 9–14. ACM, New York (2007)
47. Tuyls, P., Schrijen, G.-J., Škorić, B., van Geloven, J., Verhaegh, N., Wolters, R.: Read-proof hardware from protective coatings. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 369–383. Springer, Heidelberg (2006)
48. Verbauwhede, I., Schaumont, P.: Design methods for security and trust. In: Proc. of Design Automation and Test in Europe (DATE 2008), NICE, FR, p. 6 (2007)
49. Weingart, S.H.: Physical security devices for computer subsystems: A survey of attacks and defences. In: Paar, C., Koç, Ç.K. (eds.) CHES 2000. LNCS, vol. 1965, pp. 302–317. Springer, Heidelberg (2000)