PUF-PRFs: A New Tamper-resilient Cryptographic Primitive

Frederik Armknecht\textsuperscript{1} and Roel Maes\textsuperscript{2} and Ahmad-Reza Sadeghi\textsuperscript{1} and Berk Sunar\textsuperscript{3} and Pim Tuyls\textsuperscript{2}

\textsuperscript{1} Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
\textsuperscript{2} ESAT/COSIC and IBBT, Catholic University of Leuven, Belgium
\textsuperscript{3} Cryptography & Information Security, WPI, MA USA

Abstract. With the proliferation of physical attacks that may compromise even the theoretically strongest cryptographic schemes, the need for affordable physical protection of cryptographic devices becomes more visible by each day. In this context, Physically Unclonable Functions (PUFs), a promising new technology, provide a low cost technique to realize tamper-resilient storage for secret keys in integrated circuits (ICs). However, PUFs possess some unusual properties that set them apart from ordinary hardware components: their responses are noisy and tend to change when PUFs are manipulated through external influences. These properties have limited the applications of PUFs so far to only physically protecting stored key material. This raises the question as to what extent PUFs can be used to construct other cryptographic schemes.

In this paper, we take the first step towards answering this question and place PUFs in the core of a pseudorandom function (PRF) construction. PRFs are one of the most important cryptographic primitives used to design various cryptographic schemes such as stream or block ciphers. We first give a theoretical model for PUFs and justify it by real-life PUF-implementations. Then, we show how to use PUFs to construct tamper-resilient PRFs, termed as PUF-PRFs. However, for several reasons that we outline in this work, PUF-PRFs cannot directly replace PRFs. Nevertheless, we show that PUF-PRFs represent a new cryptographic primitive with very useful properties: they are inherently resilient to tampering attacks and allow for generating (almost) uniformly distributed values.

Keywords: Pseudorandom functions, Physically Unclonable Functions, Block Ciphers, Tamper Resilience.

1 Introduction

Modern cryptography provides a variety of useful tools and methodologies to analyze and to prove the security of cryptographic schemes such as in \cite{3, 4, 2, 7}. These theoretical frameworks focus mainly on the algorithmic aspects under the so-called black-box assumption. Here it is assumed that security-critical information, e.g., key material, can be kept secret. However, real world cryptographic applications are embedded in physical environments that may leak information, e.g., about the cryptographic key material being processed.

A large body of literature has therefore explored various forms of passive and active side-channel attacks, e.g. \cite{24, 23, 1}. This motivated new security definitions to capture the effect of (physical) information leakage or tampering. The concept of Physically Observable Cryptography \cite{29} considers a physical leakage channel, however, under some (unavoidable) assumptions that idealize this channel. The Algorithmic Tamper Proof (ATP) model \cite{13}, establishes the conditions under which cryptographic primitives such as signature schemes with black-box security reductions may be converted to primitives secure against a tampering adversary, however, under the strong assumption that tamper-proof and read-proof storages are necessary. In \cite{22, 30, 8}, it was discussed how tamper-proof tokens can be used to relax setup assumptions that in practice require some trusted party for the system initialization.

It may be argued that the biggest threat on cryptographic devices stems from tampering attacks. In these attacks, the adversary is playing an active role, i.e., he injects faults into the device and observes their manifestations with the goal of deducing internal secrets. For instance, the attack in \cite{5} recovers the RSA factors by introducing an arbitrary fault in one of two RSA signature computations. Another fault injection attack on AES requires only two faulty ciphertexts to retrieve the 128-bit secret key \cite{34}. In an even more sobering attack it was shown that an arbitrary bit location in memory could be modified by optical induction \cite{36}.
A straightforward tamper detection approach is to integrate on-chip sensors. However, it is difficult to assess the strength of this technique since the sensitivity depends on the type of sensors and their exact placement. Another tamper detection technique is to employ a protective coating layer [35]. The coating approach provides comprehensive protection. However, it requires the manufacturing process to be modified and hence is more costly. A number of process agnostic techniques based on recoding operands using error detection codes (EDCs) were proposed [21, 12]. For EDCs one may precisely quantify the error detection performance. Unfortunately, to achieve an acceptable degree of protection these techniques commonly require in the order of 200%-300% area overhead. Moreover, for highly non-linear components it is very difficult to design a protection network.

A recent line of work proposes tamper-detection at the physical level. More precisely, deep submicron and nano-scale physical phenomena are used to build low-cost tamper-evident key storage devices [32, 37]. The most promising approach in this context is to use hardware primitives called Physically Unclonable Functions (PUFs) introduced in [32, 33] and further developed, e.g., in [18]. A PUF is a primitive that maps challenges to responses which are highly dependent on the physical properties of the device in which the PUF is embedded. PUFs are based on the subtleties of the operating conditions as well as random variations that are imprinted into an integrated circuit during the manufacturing process. This phenomenon, i.e., manufacturing variability, creates minute differences in circuit parameters, e.g., capacitances, line delays, threshold voltages etc., in chips which otherwise were manufactured to be logically identical. Manufacturing variability requires additional margins to be implemented. There are numerous references modeling manufacturing variability and proposing techniques to mitigate its effects, e.g., [9, 6, 19]. However, it is well known that in smaller technology nodes (below $0.18\mu m$ technology), the relative impact of deep submicron process variations becomes larger and larger and cannot be removed anymore [38]. These random variations vary from chip to chip and cause it to be physically unique. For some PUF instantiations it is assumed that copying (cloning) of these random structures is infeasible or prohibitively costly.

PUFs are extremely powerful tools for tamper detection. In contrast to the previously introduced techniques, PUFs combine the storage, detection, and secret destruction features within a single unit. The storage itself is sensitive to tampering. Typically, with other techniques one must ensure that the error detection or secret destruction hardware are also protected from tampering. The arbiter-PUF approach proposed in [11, 25], for instance, provides secure key storage and protection of circuit components placed sufficiently close to PUF. The number of possible challenges is exponential in the arbiter-PUF size, but since it is susceptible to linear modelling attacks [31], the number of challenge-response pairs that can actually be used is limited. Another PUF construction derives a secret key directly from a protective coating layer [37]. When the coating is disturbed via a tampering attack the derived secret key is also modified. Despite the comprehensive protection it provides, similar to the protective layer approach discussed in [35], this technique requires modifications to the manufacturing process. A more recently introduced promising technique is the SRAM PUF [16]. Since SRAM cells are standard components used in chips, they do not require expensive modifications to the manufacturing process. SRAM PUFs are built from standard semiconductor components which are available early in a new manufacturing technology and do not require changes to design rules or processing. When uniformly placed throughout the device, the SRAM PUF cells protect the other components placed in close distance to the PUF as well, hence providing comprehensive protection.

**Our contribution.** In this paper, we place the PUFs in 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 [15] are fundamental primitives in cryptography and have many applications (see, e.g., [14, 26, 27]).

To the best of our knowledge, all construction methods for PRFs so far are purely software-based and hence would require additional measures for tamper-resilience. The tight integration of PUFs as PRFs into a cryptographic construction improves the tamper-resilience of the overall design. Any attempt at accessing the internals of the device will result in change of the pseudorandom functions. Hence, no costly error detection networks or alternative anti-tampering technologies are needed. The unclonability

---

4 Note that although True Random Number Generators (TRNGs) also exploit physical phenomena to generate randomness they cannot be used as functions since challenging a TRNG twice with the same input does not yield the same (or at least) similar outputs. This disqualifies TRNGs for the considered cryptographic applications.
Physically Unclonable Functions. There exists a distribution uniformly distributed. This is captured by the following model:

\[ \text{haviour even if some challenge-response pairs are already sampled. Given their natural protection against the deep submicron and nano-scale level. For several PUF types, this makes it difficult to predict its be-} \]

\[ \text{tween both cases.} \]

\[ \text{y} \]

\[ P \]

\[ x \]

\[ \text{is pseudorandom if the advantage of winning the following game is negligible. A distinguisher has black-} \]

\[ \text{FE:} \]

\[ (\text{FE}), \text{e.g., see [10], are an established tool in cryptography. First recall the definition of non-uniform inputs into reliably reproducible, uniformly distributed random strings. For this purpose,} \]

\[ \text{Given the observations made above, the main obstacle in creating a PRF from a PUF is to convert noisy} \]

\[ \text{space, many technical details are omitted and also the definitions are only given informally.} \]

\[ \text{important cryptographic schemes can be constructed from them (see, e.g., [14, 26, 27]). In the following,} \]

\[ \text{In Section 3, we provide a practical instiation of a PUF that meets the conditions of the previously} \]

\[ \text{proposed model: the SRAM PUF. Finally, in Section 4 we present the conclusions.} \]

2 Physically Unclonable Functions and Pseudorandom Functions

The notion of pseudorandom functions (PRFs) [15] is established since long in cryptography and many important cryptographic schemes can be constructed from them (see, e.g., [14, 26, 27]). In the following, we will sketch how to use PUFs for realizing PRFs (more precisely, a variant of it). Due to the lack of space, many technical details are omitted and also the definitions are only given informally.

Let \( F_{\ell,n} \) denote the set of all Boolean functions \( \{0,1\}^{\ell} \rightarrow \{0,1\}^{n} \). Roughly said, a function \( f \in F_{\ell,n} \) is pseudorandom if the advantage of winning the following game is negligible. A distinguisher has black-box access to an oracle which on inputs \( x \in \{0,1\}^{\ell} \) returns values \( y \in \{0,1\}^{n} \). These values are either \( y = f(x) \) (case 0) or \( y \) is uniformly random chosen from \( \{0,1\}^{n} \) (case 1). The distinguisher has to decide between both cases.

As told in the introduction, the behaviour of PUFs is highly dependent on its physical structure at the deep submicron and nano-scale level. For several PUF types, this makes it difficult to predict its behaviour even if some challenge-response pairs are already sampled. Given their natural protection against tampering attacks, PUFs are ideal candidates for realizing PRFs with inherent physical protection.

However, PUFs differ in two aspects from PRFs: (i) the outputs are noisy and (ii) usually not uniformly distributed. This is captured by the following model:

Definition 1 (Physically Unclonable Functions). A \((\ell,m,\delta,q,\mu,\epsilon')\)-family of PUFs is a set of functions \( \mathcal{P} \) with the following properties:

**Noise:** Every \( \Pi \in \mathcal{P} \) is a probabilistic algorithm where

1. \( \Pi \) accepts inputs (challenges) \( x \in \{0,1\}^{\ell} \) and generates outputs (responses) \( y \in \{0,1\}^{m} \).
2. There exists a Boolean function \( f : \{0,1\}^{\ell} \rightarrow \{0,1\}^{m} \) such that for each query \( x \in \{0,1\}^{\ell} \), the response \( y \in \{0,1\}^{m} \) has the form \( y = f(x) \oplus e \) where \( e \in \{0,1\}^{m} \) is some random (noise) vector.
3. All noise vectors have a Hamming weight of \( \delta \) or less.

**Output distribution:** There exists a distribution \( \mathbb{D} \) on \( \{0,1\}^{m} \) with a min-entropy \( H_{\infty}(\mathbb{D}) \geq \mu \) such that it holds for every sequence of vectors \( x_{1}, \ldots, x_{q} \in \{0,1\}^{\ell} \) with \( x_{i} \neq x_{j} \) for \( i \neq j \) that the following two distributions have a statistical distance of at most \( \epsilon' \):

1. \((\Pi(x_{1}), \ldots, \Pi(x_{q})) \) where \( \Pi \) is uniformly chosen from \( \mathcal{P} \).
2. \((y_{1}, \ldots, y_{q})\) with \( q \) independent samples \( y_{i} \sim \mathbb{D}^{q} \).

A \((\ell,m,\delta,q,\mu,\epsilon')\)-PUF is a member of a \((\ell,m,\delta,q,\mu,\epsilon')\)-family of PUFs.

Given the observations made above, the main obstacle in creating a PRF from a PUF is to convert noisy non-uniform inputs into reliably reproducible, uniformly distributed random strings. For this purpose, fuzzy extractors (FE), e.g., see [10], are an established tool in cryptography. First recall the definition of FE:

Definition 2 (Fuzzy Extractor). A \((m,n,\delta,\mu,\epsilon)\)-fuzzy extractor \( E \) is a pair of randomized procedures, “generate” (Gen) and “reproduce” (Rep), with the following properties:

1. The generation procedure Gen on input \( y \in \{0,1\}^{m} \) outputs an extracted string \( z \in \{0,1\}^{n} \) and a helper string (also called helper data) \( \omega \in \{0,1\}^{*} \).
2. The reproduction procedure Rep takes as input a sample \( (y',\omega) \) and a random \( \omega \) as inputs. The correctness property of fuzzy extractors guarantees that if the Hamming distance \( dist(y,y') \leq \delta \) and \( z,\omega \) were generated by \( (z,\omega) \sim Gen(y) \), then \( Rep(y',\omega) = z \). If \( dist(y,y') > \delta \), then no guarantee is provided about the output of Rep.
3. The security property guarantees that for any distribution $\mathbb{D}$ on $\{0,1\}^m$ of min-entropy $\mu$, the
string $z$ is nearly uniform even for those who observe $\omega$: if $(z, \omega) \leftarrow \text{Gen}([\mathbb{D}])$, then it holds that
$\text{SD}(z|\omega, (U_n, \omega)) \leq \epsilon$.

In [10], several constructions for efficient fuzzy extractors have been presented. PUFs are most commonly
used in combination with fuzzy extractors based on error-correcting codes and universal hash functions.
In that 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. To construct a PRF from PUFs, one first invokes the PUF and then applies an appropriate FE
afterwards. This will be formalized in the following definition:

Definition 3 (PUF-PRFs). $A (\ell, n, q, \epsilon, \epsilon')$-PUF-PRF is a composition $(E \circ \Pi)$ where

- $\Pi$ is a $(\ell, m, \delta, q, \mu, \epsilon')$-PUF and
- $FE$ is a $(m, n, \delta, \mu, \epsilon)$-fuzzy extractor.

More precisely, a PUF-PRF can be used to either generate or to reproduce values. For generating values
from an input $x \in \{0, 1\}^\ell$, first the PUF is executed on it and afterwards the $\text{Gen}$ algorithm is run to
produce a tuple $(z, \omega)$. To reproduce the value $z$ in later executions, one first applies the PUF $\Pi$ to $x$
and then uses the $\text{Rep}$ algorithm together with the previously generated helper data $\omega$.

Using the theory of fuzzy extractors, one can show that the values $z$ are almost uniformly distributed,
even if the helper data $\omega$ is known. That is, PUF-PRFs and “traditional” PRFs have in common that
(part of) the output cannot be distinguished from uniformly random values. In that sense, the output
of PUF-PRFs can be used as a replacement for PRF outputs and one might be tempted to plug in
PUF-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 [20]. Hence, extra attention
must be paid when deploying PUF-PRFs in cryptographic schemes. In the following section, we discuss
possible theoretical and practical applications of PUF-PRFs.

3 Practical Instantiation: SRAM PUF

In this section, a practical instantiation of a Physically Unclonable Function is discussed: the SRAM
PUF. We will explain the SRAM PUF operation and further on show that it meets the PUF conditions
from Definition 1.

During the manufacturing process, subtleties of the operating conditions as well as random varia-
tions are imprinted into an integrated circuit. This phenomenon, i.e. manufacturing variability, creates
minute differences in circuit parameters, e.g. capacitances, line delays, threshold voltages etc., in chips
which otherwise were manufactured to be logically identical. Manufacturing variability requires addi-
tional margins to be implemented. There are numerous references modeling manufacturing variability
and proposing techniques to mitigate its effects, e.g., [9, 6, 19]. However, it is well known that in smaller
technology nodes (below 18µm technology), the relative impact of deep sub-micron process variations
becomes larger and larger and cannot be removed anymore [38]. These random variations vary from chip
to chip and cause it to be physically unique.

A number of PUFs, including SRAM PUFs, have been proposed that exploit the uniqueness of an IC
due to these process variations. Basically, an SRAM cell is a physical realization of a bistable memory
element that is able to store one binary digit.

Definition 4 (SRAM). A $(\ell, m)$-Static Random Access Memory (SRAM) is defined as a $2^\ell \times m$ matrix
of physical SRAM cells, each storing an element from $\{0, 1\}$. Let $\mathbf{M} \in \{0, 1\}^{2^\ell \times m}$ denote the state of the
SRAM matrix immediately after a particular power-up of the memory. Each row of $\mathbf{M}$ is uniquely labeled
with an element $x$ from $\{0, 1\}^\ell$, and a specific row vector is denoted as $\mathbf{M}_x$.

An SRAM cell is volatile by nature, meaning that it only stores a bit when powered. The value of its
state right after power up is undefined by its electrical characteristics. In practice, the power up state of
an SRAM cell is a random variable. In the full paper we demonstrate by experiments, simulations and
models that:
The noise-free power up state $M_{x,j}$ of an SRAM cell is fixed for a specific instantiation, but independently and uniformly distributed over $\{0, 1\}$ for a randomly sampled instantiation from the statistical SRAM cell population.

The actual noisy power up state of an SRAM cell is given by $\tilde{M}_{x,j} = M_{x,j} \oplus e$, with $e$ a Bernoulli distributed random variable with probability of success $p_e < \frac{1}{2}$. $e$ is drawn independently at the powerup of every SRAM cell.

From these observations, we show that:

**Theorem 1.** Let $\tilde{M}$ be the noisy power up state matrix that arises after a specific powerup of a specific physical SRAM realization. The procedure that accepts as input a challenge $x \in \{0, 1\}^\ell$ and thereupon returns the row vector $y = \tilde{M}_x$ as response, is a realization of an $(\ell, m, \delta, q, \mu, \epsilon')$-PUF as defined by Definition 1 and is called an SRAM PUF.

In [18] an SRAM PUF was constructed on an FPGA and the theoretical values for the min-entropy and the average bit error probability were experimentally verified. The performed experiments indicate that the average bit error probability of the response bits is bounded by 4% when the temperature is kept constant at $20^\circ C$, and by 12% at large temperature variations between $-20^\circ C$ and $80^\circ C$. The probability of more than $\delta$ bit errors occurring decreases exponentially with increasing $\delta$ according to the Chernoff bound. $\delta$ is chosen high enough such that in practice, more than $\delta$ bit errors will never be observed. Accurately determining the min-entropy from a limited amount of PUF instances and responses is unattainable. In [28] it was shown that the mean smooth min-entropy of a stationary ergodic process is equal to the mean Shannon entropy of that process. Since the SRAM PUF responses are distributed according to such a stationary distribution (as they result from a physical phenomenon) it was estimated in [17] that its Shannon entropy equals 0.95 bit/cell. Because the mean smooth min entropy converges to the mean Shannon entropy, it follows that $H^\infty_{\epsilon'}(M_x)$ is close to $H(M_x)$. Therefore we put $H^\infty_{\epsilon'}(M_x) = 0.95 \cdot m \approx \mu$. Since the power up states are independently distributed, the min-entropy of a response does not decrease after multiple queries, i.e. $\epsilon' = 0$ even after $q = 2^\ell$. The number of SRAM cells required to construct the PUF rises linearly in the output size $m$, but exponentially in the input size $\ell$. Therefore, SRAM PUFs more naturally yield an expanding PUF: $\ell \ll m$.

4 Conclusions

In this work, we presented a method for constructing PRFs from physically unclonable functions. The method is based on a theoretical model for PUFs. We provided arguments that showed that certain classes of PUFs are adequately captured by this model. Moreover, we demonstrated a practical PUF construction and determined its parameters for the model from experiments and simulations. Hence the proposed construction technique allows real-life instantiations of PUF-PRFs where the security can be proven under some reasonable physical assumptions.

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. Applications based on PUF-PRFs need to take the helper data leakage into account. We hope to provide cryptographic schemes based on PUF-PRFs soon.

References


