PUF-PRFs: A New Tamper-resilient Cryptographic Primitive

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

[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

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 [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. [24, 23, 1]. This motivated new security definitions to capture the effect of (physical) information leakage or tampering. The concept of Physically Observable Cryptography [29] considers a physical leakage channel, however, under some (unavoidable) assumptions that idealize this channel. The Algorithmic Tamper Proof (ATP) model [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 [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 [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 [34]. In an even more sobering attack it was shown that an arbitrary bit location in memory could be modified by optical induction [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 $.18\mu m$ 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.[4] 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.

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.

Similar to PRFs, PUF-PRFs generate almost uniform outputs. However, as opposed to PRFs, they require some additional data for correct execution which in turn possibly leaks side information.

Organization. This paper is organized as follows. In Section 2, we give a formal model for certain types of PUFs and explain how these can be turned into a new cryptographic primitive, termed PUF-PRFs. In Section 3, we provide a practical instiation of a PUF that meets the conditions of the previously 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 $\mathbb{F}_{\ell,n}$ denote the set of all Boolean functions $\{0,1\}^\ell \to \{0,1\}^n$. Roughly said, a function $f \in \mathbb{F}_{\ell,n}$ is pseudorandom if the advantage of winning the following game is negligible. A distinguisher has black-box access to an orcale 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. *Π accepts inputs (challenges) $x \in \{0,1\}^\ell$ and generates outputs (responses) $y \in \{0,1\}^m$.*
2. *There exists a Boolean function $f : \{0,1\}^\ell \to \{0,1\}^m$ 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. In other words, the responses can be interpreted as being the outputs of f transmitted via an additively noisy channel. The vector e is called the* noise vector*.*
3. *All noise vectors have a Hamming weight of δ 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 ϵ':*
1. *$(\Pi(x_1), \ldots, \Pi(x_q))$ where Π is uniformly chosen from $\mathcal{P}$.*
2. *$(y_1, \ldots, y_q)$ with q independent samples $y_i \leftarrow \mathbb{D}^q$.*

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 an element $y' \in \{0,1\}^m$ and a bit string $\omega \in \{0,1\}^*$ as inputs. The correctness property of fuzzy extractors guarantees that if the Hamming distance $\mathsf{dist}(y, y') \leq \delta$ and z, ω were generated by $(z, \omega) \leftarrow \mathsf{Gen}(y)$, then $\mathsf{Rep}(y', \omega) = z$. If $\mathsf{dist}(y, y') > \delta$, 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* μ*, the string* z *is nearly uniform even for those who observe* ω*: if* $(z,\omega) \leftarrow \mathsf{Gen}(\mathbb{D})$*, then it holds that* $\mathrm{SD}((z,\omega),(\mathbb{U}_n,\omega)) \leq \epsilon$.

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* $(\mathsf{E} \circ \Pi)$ *where*

- Π *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 Gen algorithm is run to produce a tuple (z,ω). To reproduce the value z in later executions, one first applies the PUF Π to x and then uses the Rep algorithm together with the previously generated helper data ω.

Using the theory of fuzzy extractors, one can show that the values z are almost uniformly distributed, even if the helper data ω 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 variations 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 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 $.18\mu m$ 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* (ℓ, 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* $\widetilde{\boldsymbol{M}} \in \{0,1\}^{2^\ell \times m}$ *denote the state of the SRAM matrix immediately after a particular power-up of the memory. Each row of* $\widetilde{\boldsymbol{M}}$ *is uniquely labeled with an element* x *from* $\{0,1\}^\ell$*, and a specific row vector is denoted as* $\widetilde{\boldsymbol{M}}_x$.

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 $\mathbf{M}_{x,j}$ 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 $\widetilde{\mathbf{M}}_{x,j} = \mathbf{M}_{x,j} \oplus e$, with e a Bernoulli distributed random variable with probability of succes $p_e < \frac{1}{2}$. e is drawn independently at the powerup of every SRAM cell.

From these observations, we show that:

Theorem 1. *Let $\widetilde{\boldsymbol{M}}$ be the noisy powerup 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 $y = \widetilde{\boldsymbol{M}}_x$ 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 $20°C$, and by 12% at large temperature variations between $-20°C$ and $80°C$. The probability of more than δ bit errors occurring decreases exponentially with increasing δ according to the Chernoff bound. δ is chosen high enough such that in practice, more than δ 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 $H^\epsilon_\infty(\mathbf{M}_x)$ is close to $H(\mathbf{M}_x)$. Therefore we put $H^\epsilon_\infty(\mathbf{M}_x) = 0.95 \cdot m \approx \mu$. 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 ℓ. 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

1. Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The em side-channel(s). In *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 29–45, London, UK, 2003. Springer-Verlag.
2. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT*, pages 139–155, 2000.
3. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pages 232–249, London, UK, 1994. Springer-Verlag.

4. Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, New York, NY, USA, 1995. ACM.
5. D. Boneh, R. A. Demillo, and R. J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 2:101–119, 2001.
6. S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impacts on circuits and microarchitecture. In *Design Automation Conference – DAC 03*, 2003.
7. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 453–474, London, UK, 2001. Springer-Verlag.
8. Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In *Advances in Cryptology – EUROCRYPT 2008*, pages 545–562. Springer Verlag, 2008.
9. Choongyeun Cho, D.D. Kim, Jonghae Kim, J.-O. Plouchart, Daihyun Lim, Sangyeun Cho, and R. Trzcinski. Decomposition and analysis of process variability using constrained principal component analysis. *Semiconductor Manufacturing, IEEE Transactions on*, 21(1):55–62, 2008.
10. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.
11. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled Physical Random Functions. In *Annual Computer Security Applications Conference — ACSAC 2002*, page 149, Washington, DC, USA, 2002. IEEE Computer Society.
12. G. Gaubatz, B. Sunar, and M. G. Karpovsky. Non-linear residue codes for robust public-key arithmetic. In *In Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC '06)*, 2006.
13. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography*, volume 2951, pages 258–277, 2004.
14. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 276–288, New York, NY, USA, 1985. Springer-Verlag New York, Inc.
15. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
16. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems — CHES 2007*, volume 4727 of *LNCS*, pages 63–80. Springer, September 10-13, 2007.
17. Jorge Guajardo, Sandeep Kumar, Pim Tuyls, Roel Maes, and Dries Schellekens. Reconfigurable Trusted Computing with Physical Unclonable Functions. Submitted, under review., June 2008.
18. Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *CHES*, pages 63–80, September 2007.
19. L. He, A. B. Kahng., K. Tam, and J. Xiong. Variability-driven considerations in the design of integrated-circuit global interconnects. In *IEEE VLSI Multilevel Interconnection Conference*, 2004.
20. Tanya Ignatenko and Frans Willems. On the security of the XOR-method in biometric authentication systems. In *Twenty-seventh symposium on Information Theory in the Benelux*, pages 197–204, 2006.
21. M. Karpovsky, K. Kulikowski, and A. Taubin. 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.
22. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *Advances in Cryptology – EUROCRYPT 2007*, pages 115–128. Springer Verlag, May 2007.
23. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Lecture Notes in Computer Science*, 1666:388–397, 1999.
24. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pages 104–113, London, UK, 1996. Springer-Verlag.
25. 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.
26. M. Luby. *Pseudo-randomness and applications*. Princeton University Press, 1996.
27. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
28. Ueli Maurer, Renato Renner, and Stefan Wolf. *Unbreakable keys from random noise*, pages 21–44. Springer Verlag, 2007.
29. Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *TCC 2004, LNCS*, pages 278–296. Springer, 2004.

30. Tal Moran and Gil Segev. David and goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In *Advances in Cryptology – EUROCRYPT 2008*, pages 527–544. Springer Verlag, 2008.
31. Erdinc Ozturk, Ghaith Hammouri, and Berk Sunar. Towards robust low cost authentication for pervasive devices. *Pervasive Computing and Communications, IEEE International Conference on*, 0:170–178, 2008.
32. R. S. Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, March 2001.
33. R. S. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297(6):2026–2030, 2002.
34. G. Piret and J.-J. Quisquater. A differential fault attack technique against spn structures, with application to the AES and KHAZAD. In *Cryptographic Hardware and Embedded Systems Workshop (CHES-2003)*, pages 77–88, 2003.
35. R. Posch. Protecting devices by active coating. *Journal of Universal Computer Science*, 4:652–668, 1998.
36. Sergey Skorobogatov. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems Workshop (CHES-2002)*, volume LNCS 2523, pages 2–12. Springer-Verlag, 2007.
37. P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters. Read-Proof Hardware from Protective Coatings. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *LNCS*, pages 369–383. Springer, October 10-13, 2006.
38. H. Veendrick. *Deep-submicron CMOS IC*. Kluwer academic publishers, second edition, 2000.