# The PUF Promise (Short Paper)

Heike Busch[1], Miroslava Sotáková[2], Stefan Katzenbeisser[1], and Radu Sion[2]

[1] Technische Universität Darmstadt
[2] Stony Brook University

**Abstract.** Physical Uncloneable Functions (PUF) are systems whose *physical* behavior to different inputs *can* be measured reliably, yet *cannot* be cloned in a physical replica. Existing designs propose to derive uncloneability from an assumed practical impossibility of exactly replicating inherent manufacturing variations, e.g., between individual chipset instances. The PUF promise has drawn significant attention lately and numerous researchers have proposed to use PUFs for various security assurances ranging from authentication to software licensing. In this paper we survey the history of PUFs as well as the existing body of research proposing applications thereof.

## 1 Introduction

The idea to build secure cryptographic schemes, using tamper-proof hardware instead of relying on unproven number-theoretic assumptions, has been around for a long time [2, 21]. Research in this area has become more intensive recently, when Pappu [24] introduced the concept of *Physical Uncloneable Functions* (PUF) (also called “physical one-way functions”). A PUF is implemented by a physical device which can be seen as a source of randomness and due to uncontrollable manufacturing variations, is impossible to clone physically [4, 6]. The inputs to such a function are usually called *challenges* and specify measurements to be applied to the device. The outputs of a PUF are the corresponding measurement outcomes and are usually referred to as *responses*. Furthermore, the response to a challenge that has not been queried (i.e., the particular measurement has not been performed) should hard to guess. The nature of PUFs suggests making use of them in device authentication, key-agreement, and secure key-storage. For illustration purposes consider a simple authentication protocol in which case Bob, holding the device, wants to convince Alice about it: Alice queries a set of challenges, gets the responses, and stores all these *challenge-response pairs* (CRP). Then she sends the device to Bob, and Bob announces his response to Alice. If the response matches Alice’s key, Alice accepts, otherwise she rejects. Since the device is assumed to be uncloneable, an adversary cannot learn the responses, unless it manages to measure the device original. This could possibly happen before Bob receives the device. Nevertheless, if the adversary is limited in the number of CRPs it can measure, it is unlikely to guess the exact set of Alice’s challenges, before it is announced. Moreover, the responses of the PUF can be used to generate a secret key in order to use PUFs in key-agreement protocols.

Since several PUF-based challenge-response authentication protocols with PUF implementations have been proposed in [3, 8, 16, 33], the authors in [5] noted that current

PUF-based authentication only prevents the impersonation of the client and do not prevent impersonation of the server in the context of physical attacks. To solve this problem, one needs a mechanism that allows the client to distinguish between challenges selected by the server in the enrollment step and an attacker. Another interesting application of PUFs is protecting software that runs on embedded systems. Instead of building functionality entirely in hardware, many vendors utilize standard computing equipment and differentiate through software. Unfortunately, software can easily be copied and reverse-engineered which is a real problem in professional product piracy, where software is copied from one legitimate device, and installed on many other (unauthorized) products. Many vendors thus want to bind software against a specific hardware platform or even to a specific instance. The latter can be achieved by PUFs as shown by Simpson and Schaumont [27]. The basic idea is to include a mutual authentication protocol between the provider's software (also called Intellectual Property) and the hardware platform. In this scenario, the PUF is part of an FPGA and it is used for hardware authentication and key generation. The FPGA bitstream is distributed in encrypted form, where the key is derived from a response to a specific PUF challenge. Thus, the bitstream cannot be decrypted and run on a FPGA that it has not been personalized for.

## 2 How Did PUFs Come About

Pappu introduced PUFs in [24] where a Physical Uncloneable Function is realized as a physical system, which is easy to evaluate, but *assumed* hard to characterize. When a PUF is exposed to a physical stimulus, it answers with a *response*. The way the stimulus is applied to the PUF is specified (usually digitally) in the form of a *challenge*, while the response is measured and appropriately digitized. For a "secure" PUF, predicting the output of the physical system is intractable without actually having physical access to the device. Moreover, PUFs exploit natural manufacturing variations which make them uncloneable: even with highly complex manufacturing equipment it is (*assumed to be*) impossible to create a second, completely identical device with the same challenge-response behavior – this holds even for the manufacturer of the original device. Thus, PUFs can be used to produce unique and uncloneable objects without having to trust the manufacturer.

Note, since the responses of PUFs are noisy by nature the output of a PUF cannot directly be used in applications that require noise-free output with a perfectly uniform distribution (such as cryptographic keys). To deal with this problem, *fuzzy extractors* of Dodis et al. are applied – a secure form of error correction that enables a reliable extraction of an uniform key from a noisy non-uniform input [7]. Since almost all known PUF implementations produce noisy outputs, PUF implementations will have to be complemented with fuzzy extractors and helper data. Some errors can also be avoided by employing a calibration operation, which is driven by PUF CRPs, as described in [37].

### 2.1 The First Idea – Optical PUFs

As a first way of implementing PUFs, Pappu proposed an optical approach. An "optical PUF" consists of a transparent material, where many light scattering particles are added in a random way during production. Such a device causes a random speckle pattern when shining a laser beam onto it; here, the position and angle of the laser (and, we believe, possibly other parameters such as amplitude and wave-length) represent the challenge, while the speckle pattern is recorded, quantized and encoded to form the PUF response. In the original work, the author uses these challenge-response pairs to identify specific devices or to extract cryptographic keys [24, 25]. To this end, the PUF is measured right after production on a few random challenges (this step is referred to as "enrollment") to obtain a database of valid challenge-response pairs (CRPs) for a particular device. A device can subsequently be identified once it is placed in the field, by measuring the response for one of the challenges selected during enrollment (this process is called "verification"). If the response matches the expected, pre-recorded response, the device is authenticated and the response can be used to derive keys.

Naturally, a PUF should support a large number of CRPs in order to make it infeasible to learn responses to challenges that were not yet issued. In [34] the authors estimate the entropy of an optical PUF ($\geq 4 \cdot 10^6$ per $5\text{cm}^2$) and the information contained in one CRP. Based on this data, the authors calculate the corresponding number of independent CRPs ($\geq 3 \cdot 10^4$ per $5\text{cm}^2$), which turns out to be much lower than the number of all possible (not necessarily independent) challenges ($\sim 10^{10}$). As the number of independent CRPs is rather low and they can all be pre-recorded by an attacker who has unlimited physical access to the PUF once, *optical PUFs do not offer security in the information-theoretic sense*. However, in [34] the authors claim that interpolation of the PUF's behavior is computationally costly and therefore, a lot more challenges need to be measured to successfully predict the response for a fresh challenge. To prevent the attacker from exhaustively reading out all the CRPs (meaning, not only the independent ones), a method for decreasing the measurement-rate is proposed. For instance, if $10$ms are required to measure one challenge, the attacker can measure about $\frac{1}{100}$ of all challenges ($\sim 10^8$) in a week of uninterrupted access to the optical PUF [34]. As a drawback, developing a reliable measurement apparatus for optical PUFs is a complex problem, which requires costly high-precision mechanics and thus limits their usage.

Due to the internal structure of the PUF it is very difficult to produce a physical clone because it requires a difficult and costly process (e.g. put the particles in the right position). Furthermore, modeling the PUF is very hard since the scattering of the PUF response is very complex. Note, that there are many papers that investigate this topic [17, 24, 25, 34, 37], the effect of changing measurement conditions [37] or the secrecy rate of optical PUFs more in details [17].

## 2.2 IC-based Implementations – Silicon, Arbiter, and Ring Oscillator PUFs

Gassend et al. proposed a new instantiation of PUFs that uses silicon technology [12]. Based on the approach of [31], where it is shown that uncontrollable process variations during chip production make chips measurably different, *Silicon Physical Uncloneable Functions* (SPUF) exploit inherent variations in integrated circuits (IC) – that exist even for chips that were produced with identical layout masks [9, 11]. An important advantage of silicon PUFs is that their production does not require any special devices on top of classic chip manufacturing equipment.

Based on the observation that the timing behavior of chips differs [12], Lim et al. introduced *Arbiter Physical Uncloneable Functions* (APUF) [10, 20, 22, 23]. Arbiter PUFs consist of a number of switch delay elements, which are connected in series. Every element has two inputs and uses a two-to-one multiplexer[3] to swap its inputs depending on one challenge bit: If the challenge bit is $0$ both signals go straight trough the element. Otherwise, the top and bottom signals are switched. To compute the output for a specific challenge, a rising signal is given to the two inputs at the same time. Both signals race through the device; at the end, an arbiter circuit determines which signal passed the device faster. Thus, the challenge of the PUF still determines the path that both signals take through the device, while the response will now be a *single* bit $r \in \{0, 1\}$. Since each delay element doubles the number of paths the signals can possibly take, an APUF with $n$ elements can produce $2^n$ delay paths. To obtain an $m$-bit response, one can either duplicate the circuit $m$ times or evaluate the device consecutively on $m$ different challenges and paste the results together.

All PUF implementations that are based on delay characteristics in ICs are not protected against environmentally induced noise. Consequently, a PUF produces different measurements for the same stimulus. Furthermore, if the variations of the PUF measurements are to high and the measurement variations are not adequately improvable a PUF may not be uniquely identified. Lim et al. handle the problem of environmentally induced noise by analyzing the coherence between environmental variations and circuit delays such as temperature and power supply [22, 23]. Firstly, the authors measured an *inter-chip variation* which states how many bits of two responses measured by two different PUFs for the same challenge are diverse. The average inter-chip variation of a PUF should be close to $50\%$ whereas the bits of a PUF response are uniformly distributed and independent. Subsequently, the authors analyze the *environmental variation* which states how many bits of PUF responses will change if they are measured from the same PUF (the noise of the PUF response). The average environmental variation of a PUF should be ideally $0\%$. For an arbiter PUFs, the authors obtained the average inter-chip variation of $23\%$ and an environmental variation of $\approx 4,82\%$, if the temperature increases greater than $40°C$ from $27°C$, respectively $\approx 3,74\%$, if the voltage variation increases $\pm 2\%$. This shows that an arbiter-based PUF reduces the environmental variations well enough below the average inter-chip variation of $23\%$.

Concerning the security of PUFs, it was shown in [23] that the response of an IC-based PUF circuit can be represented as a linear function of a challenge. If an attacker knows all delays of each element of a path through the circuit it can derive (predict) a response for a given challenge by calculating the sum of the delays of each element. Since measuring the delays at each element is a hard problem, an attacker can use machine-learning-techniques to build a software circuit that models the PUF circuit. With this model, the attacker can simulate the PUF and can predict a response for a random challenge. Note, that using the linear delay model implies that the PUF response is ideally statistical distributed. In reality, however, this is not the case due to measurement or environmental variations. Nevertheless, Lim generalized this model to a probabilistic one

[3] A multiplexer is a device that selects one of many analog or digital input signals and forwards the selected input into a single line.

to model all the environmental variations. The author also suggests methods to modify the arbiter PUF such that the above mentioned model is no longer possible [23].

Adapted from arbiter PUFs, Suh and Devadas look for a higher reliability and an easier way of implementing PUFs on Application-Specific Integrated Circuits (ASIC) and Field-Programmable Gate Arrays (FPGA) [28]. Based on the "self-oscillating" approach in [12], the authors introduce *Ring Oscillator Physical Uncloneable Functions* (ROPUF). These PUFs are based on delay loops, which are commonly used to generate random bit strings. A delay loop, or ring oscillator, is a simple circuit that oscillates with a frequency influenced by manufacturing variations and thus cannot be predicted, yet can easily be determined by a counter. The PUF construction uses $n$ such circuits and compares the frequency of two selected ones: depending on which oscillator is faster, an output of $1$ or $0$ is produced. To produce an output of several bits, one picks randomly a set of such oscillators according to the challenge; comparing each pair produces one output bit. In this way, one can generate $\Theta(n \log n)$ bits of entropy out of $n$ oscillators. Suh et al. subsequently used their PUF in the development of the AEGIS processor [29, 30], which can resist both *software* and *physical* attacks. In particular, they use the PUF to store secrets in a secure, uncloneable and cost effective way.

Although ring oscillator PUFs are more reliable and easier to implement on both ASICs and FPGAs, arbiter PUFs are faster, smaller and consume less power. Thus, arbiter PUFs are better suitable for resource constrained platforms such as RFIDs, in which context they are also commercially available [18, 35].

## 2.3 Flip-Flop-based implementations – SRAM and Butterfly PUFs

As mentioned in the Introduction 1, protecting software that runs on embedded systems is a problem of growing importance. Guajardo et al. [14] revisited the results and improvements by Simpson and Schaumont [27] and instead of treating the PUF as a black-box, they propose a FPGA based IP protection mechanism, which relies on *SRAM-based Physical Uncloneable Functions* (SRAM stands for "static random access memory") [14, 15]. These PUFs consist of a number of memory cells, involving two *cross-coupled*[4] inverters, having two stable states, commonly denoted by $0$ and $1$. After power up, cells will randomly end up in state $0$ or $1$; the state that a specific memory cell will reach is mainly dependent on the production process, yet relatively constant per instance. A challenge is represented by a subset of the memory cells to be read-out after power-up; the response is their respective power-up state.

Moreover, the authors analyze how many secret bits can be extracted from the response in SRAM-based PUFs. The secrecy rate is $0.76$ bits per SRAM memory cell [14]. Note, currently available ICs can incorporate $\sim 10^6$ to $10^7$ SRAM cells. Yet, without any additional mechanism for decreasing the read-out rate, SRAM PUFs are vulnerable to an exhaustive read-out attack.

Since not all FPGAs support uninitialized SRAM memory, Guajardo et al. [19] enhanced the concept of SRAM-based PUFs to *Butterfly Physical Uncloneable Functions* (BPUF). These PUFs provide a new way of exploiting circuit delays. Butterfly PUFs use

[4] The output of the first inverter is connected to the input of the second one, and in the other way around.

unstable cross-coupled circuits, just like SRAM PUFs. While SRAM-cells are based on cross-coupled inverters, in butterfly-cells inverters are replaced by latches or flip-flops. Latches are circuits which store information and can be cleared (turns output to $0$) or preset (turns output to $1$). Like SRAM cells, butterfly cells have only two stable states. To read out the PUF, one of the latches is cleared and simultaneously the other one is preset. This brings the BPUF into an unstable condition. The butterfly-cell falls back into one of the stable states depending on the circuit delays, which are were determined by the manufacturing process. Thus, BPUFs are very similar to SRAM PUFs beside the fact that they do not need any power-up for evaluation. Unlike SRAM PUFs, BPUFs are suitable for all types of FPGAs. Similarly, Gora et al. [13] and Atallah et al. [1] proposed the use of PUFs in binding software against specific hardware.

### 2.4 Tamper-Evidence – Coating PUFs

Another approach to obtain a stronger PUF is an "active coating" – a covering that is applied to the surface of an object. Posch [26] suggests to protect a device by embedding a unique signature into the coating material used in smart cards. Tuyls et al. [32, 36] apply this idea to PUFs and introduce the concept of *Coating Physical Uncloneable Functions* (COPUF). A coating PUF employs a protective coating, covering an integrated circuit. The opaque coating material is doped with dielectric particles, having random properties concerning their size, shape, and location. Below the coating layer, a comb structure of metal wire sensors is used to measure the local capacitance of the coating. The measured values, which are random due to the randomness present in the coating, form the responses to challenges, each of them specified by a voltage of a certain frequency and amplitude, applied to a region of the sensor array.

Because of the coating the PUF is physically uncloneable since it is very hard to produce a second PUF where all sensors produce the same measurements as the original PUF. However, the coating PUF is unfortunately easy to be modeled and supports only a limited number of CRPs. Since the characterization of the coating is very difficult, coating PUFs can be used e.g. for RFID-tags or key extraction. As for key extraction, [32] succeed to generate on the average $45$ uniformly distributed bits by using $30$ sensors.

The advantage of coating PUFs is that their production price is very low. Moreover, a benefit of coating PUF is that they are suitable for detecting a certain level of physical tampering. If a device is physically attacked, its response behavior is likely to change; thus, tampering can be uncovered by measuring the PUF with specific challenges. Due to tampering, the responses usually change only locally, which could even allow the determination of specific attack positions on the chip surface.

## 3 Conclusion

In this paper, we have summarized the history of PUFs by studying PUF approaches in literature. Many constructions have been called *Physical Unloneable Function*, however, it is difficult to come up with a consistent definition. Indeed, we can deduce some "requirements" for PUFs, such as *unclonability* or *unpredictability*, but in the end the question "what a PUF is", remains difficult. Even the question ”which PUF is more suited", is not easy to answer. Current constructions depend heavily on their application and thus follow different, and sometimes contradicting goals (see Section 2.2). In conclusion, *Physical Uncloneable Functions* are a young research area where many interesting problems are open. We believe that PUFs are a promising technology that benefit from many applications.

*Acknowledgments* We thank the anonymous reviewers for valuable comments. This work was supported by CASED (`www.cased.de`). Sion and Sotáková were supported by the U.S. National Science Foundation grants CCF 0937833, CAREER CNS 0845192, CNS 0708025, IIS 0803197, and CNS 0716608, as well as by grants from Xerox, IBM and Microsoft Research.

## References

[1] Mikhail J. Atallah, Eric Bryant, John T. Korb, and John R. Rice. Binding software to specific native hardware in a vm environment: the puf challenge and opportunity. In *VMSec*, pages 45–48. ACM, 2008.

[2] D. W. Bauder. An anti-counterfeiting concept for currency systems. Research report PTK-11990. Sandia National Labs. Albuquerque, NM, 1983.

[3] Leonid Bolotnyy and Gabriel Robins. Physically unclonable function-based security and privacy in rfid systems. In *PerCom*, pages 211–220. IEEE Computer Society, 2007.

[4] Duane S. Boning and Sani Nassif. Models of process variations in device and interconnect. In *Design of High Performance Microprocessor Circuits*. IEEE Press, 2000.

[5] Heike Busch, Stefan Katzenbeisser, and Paul Baecher. Puf-based authentication protocols - revisited. In *WISA*, pages 296–308. Springer, 2009.

[6] David G. Chinnery and Kurt Keutzer. Closing the gap between asic and custom: an asic perspective. In *DAC*, pages 637–642, 2000.

[7] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. 38.1:97–139, 2008.

[8] Keith B. Frikken, Marina Blanton, and Mikhail J. Atallah. Robust authentication using physically unclonable functions. In *ISC*, pages 262–277. Springer, 2009.

[9] Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Silicon physical random functions. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 148–160. ACM, 2002.

[10] Blaise Gassend, Dwaine E. Clarke, Marten van Dijk, and Srinivas Devadas. Delay-based circuit authentication and applications. In *SAC*, pages 294–301. ACM, 2003.

[11] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Srinivas Devadas, and Marten van Dijk. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience*, 16(11):1077–1098, 2004.

[12] Blaise L.P. Gassend. Physical random functions. Master thesis, Massachusetts Institute of Technology, Massachusetts Institute of Technology, 2003.

[13] M. Gora, A. Maiti, and P. Schaumont. A flexible design flow for software ip binding in commodity fpga. In *SIES 2009*, pages 211–218, 2009.

[14] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. Fpga intrinsic pufs and their use for ip protection. In *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.

[15] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. Brand and ip protection with physical unclonable functions. In *ISCAS*, pages 3186–3189. IEEE, 2008.

[16] Ghaith Hammouri and Berk Sunar. Puf-hb: A tamper-resilient hb based authentication protocol. In *ACNS*, pages 346–365, 2008.

[17] Tanya Ignatenko, Geert-Jan Schrijen, Boris Škorić, Pim Tuyls, and Frans M. J. Willems. Estimating the secrecy rate of physical uncloneable functions with the context-tree weighting method. In *Proc. IEEE International Symposium on Information Theory 2006*, pages 499–503. IEEE Press, 2006.

[18] IntrinsicID.com. Intrinsic-ID Netherlands. Online at `http://www.intrinsic-id.com`.

[19] Sandeep S. Kumar, Jorge Guajardo, Roel Maes, Geert Jan Schrijen, and Pim Tuyls. The butterfly puf: Protecting ip on every fpga. In *HOST*, pages 67–70. IEEE Computer Society, 2008.

[20] J. W. Lee, Daihyun Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Proc. of the IEEE VLSI Circuits Symposium*, pages 176–179. IEEE Press, 2004.

[21] Frank Thomson Leighton and Silvio Micali. Secret-key agreement without public-key cryptography. In *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 456–479. Springer, 1993.

[22] 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, 2005.

[23] Daihyun Lim. Extracting secret keys from integrated circuits. Master thesis, Massachusetts Institute of Technology, Massachusetts Institute of Technology, 2004.

[24] Ravikanth Srinivasa Pappu. *Physical One-Way Functions*. Phd thesis, Massachusetts Institute of Technology, Massachusetts Institute of Technology, March 2001.

[25] Ravikanth Srinivasa Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.

[26] Reinhard Posch. Protecting devices by active coating. *J. UCS*, 4(7):652–668, 1998.

[27] Eric Simpson and Patrick Schaumont. Offline hardware/software authentication for reconfigurable platforms. In *CHES*, pages 311–323, 2006.

[28] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *DAC*, pages 9–14. IEEE, 2007.

[29] G. Edward Suh, Charles W. O'Donnell, and Srinivas Devadas. Aegis: A single-chip secure processor. *IEEE Design & Test of Computers*, 24(6):570–580, 2007.

[30] G. Edward Suh, Charles W. O'Donnell, Ishan Sachdev, and Srinivas Devadas. Design and implementation of the aegis single-chip secure processor using physical random functions. In *ISCA*, pages 25–36. IEEE Computer Society, 2005.

[31] Adrian Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *ICES*, volume 1259, pages 390–405. Springer, 1996.

[32] Pim Tuyls, Geert Jan Schrijen, Boris Skoric, Jan van Geloven, Nynke Verhaegh, and Rob Wolters. Read-proof hardware from protective coatings. In *CHES*, pages 369–383, 2006.

[33] Pim Tuyls and Boris Škorić. Strong Authentication with Physical Unclonable Functions. *Security, Privacy, and Trust in Modern Data Management*, page 133, 2007.

[34] Pim Tuyls, Boris Škorić, Sjoerd Stallinga, A.H.M. Akkermans, and Wil Ophey. Information-theoretic security analysis of physical uncloneable functions. In *Financial Cryptography*, volume 3570, pages 141–155. Springer, 2005.

[35] Verayo Inc. PUF Technology. Online at `http://www.verayo.com/`.

[36] Boris Škorić, Stefan Maubach, Tom Kevenaar, and Pim Tuyls. Information-theoretic analysis of capacitive physical unclonable functions. *Journal of Applied physics*, 100, 2006.

[37] Boris Škorić, Pim Tuyls, and W. Ophey. Robust key extraction from physical unclonable functions. In *ACNS*, volume 3531, pages 407–422, 2005.