Research on the Applications of Physically Unclonable Functions within the Internet of Things

Martin Deutschmann, Lejla Iriskic, Sandra-Lisa Lattacher, Mario Münzer, Felix Stornig, and Oleksandr Tomashchuk

Technikon Forschungs- und Planungsgesellschaft mbH
Burgplatz 3a, 9500 Villach, Austria
Mail: ses@technikon.com
Website: http://www.technikon.com

Introduction

The *Internet of Things* (IoT) is a technology in which everyday objects are interconnected via the internet in order to send and receive data [1]. The IoT experiences rapid growth, with an estimate of about 20.4 billion IoT devices expected to be in operation by 2020 [2]. As with all interconnected applications, the IoT is in the need of security, which can be a challenge to implement, as embedded devices in this domain are commonly resource-constrained and aimed at large-scale deployment, while keeping cost as low as possible. Within this document we aim to provide an overview of research conducted on the applications of hardware intrinsic security anchors, called *Physically Unclonable Functions* (PUFs) that can potentially facilitate the implementation of security within the IoT domain.

PUFs are a security primitive that exploits small-scale physical variations in objects that occur during the production process. A basic assumption of PUFs is that these variations cannot, within reasonable effort, be reproduced, and are thus not under the control of the manufacturer, hence the term *unclonable* [3].

Several types of PUFs exist and the kind that will be subsequently referred to are *intrinsic PUFs*, which are PUFs that are embedded on an integrated circuit (IC) [3].

The basic operation of PUFs can be formalized by a *challenge-response behavior*. When a challenge C is applied to a PUF, the same yields a desirably random response R. For this reason, every PUF possesses a set of *challenge-response pairs* (CRPs), each of which maps a challenge C to a response R, by a PUF $\mathcal{P}$, s.t. $\mathcal{P}(C) = R$.

In regards to formal properties, PUFs are [3] [4]:

Evaluable The function $\mathcal{P}$ is *easy* to evaluate given C.

Unique The function $\mathcal{P}(C)$ contains identity-related information about the physical entity that constitutes it.

Reproducible A response $R = \mathcal{P}(C)$ can be reproduced up to a correctable error.

Unclonable When given $\mathcal{P}$, it remains *hard* to construct a procedure $\mathcal{Q}$, s.t. $\mathcal{P}(C) \approx \mathcal{Q}(C)$.

Unpredictable Given only a set of CRPs $\mathcal{S}$, where $R = \mathcal{P}(C)$ and $C, R \in \mathcal{S}$, it remains *hard* to compute $R' = \mathcal{P}(C')$ and $C' \notin \mathcal{S}$.

One-Way When given R and $\mathcal{P}$, there exists no efficient algorithm to find C, s.t. $\mathcal{P}(C) = R$.

Tamper Evident When the physical entity that constitutes $\mathcal{P}$ is altered, this transforms $\mathcal{P}$ into $\mathcal{P}'$, s.t. $\mathcal{P}(C) \neq \mathcal{P}'(C)$ with high probability and also not up to a correctable error.

In the above context, the qualifier *easy* is stated by [3] to be context-dependent but typically corresponding to a variant of *in polynomial time and effort*. Likewise, the qualifier *hard* is stated by [3] to reflect physical and technical difficulties or impossibility.

An important property of PUF responses is that they are typically influenced by random noise. Thus, two responses that are obtained by two subsequent evaluations of the same PUF using the same challenge C, that is $R = \mathcal{P}(C)$ and $R' = \mathcal{P}(C)'$, will likely differ to a certain degree, which constitutes the so-called *intra-distance* of a PUF. When representing PUF responses as bit strings of length n, that means R, $R' \in \{0,1\}^n$, said intra-distance can be expressed by the number of different bits between the two responses from the same PUF, using the same challenge, that is the *Hamming distance*:

$$\mathcal{D}(R, R') \triangleq \#\{i : R_i \neq R'_i\}$$

Here, R_i and R'_i denote the i-th bit in R and R' respectively. The intra-distance, for two subsequent evaluations, is thus given by:

$$\mathcal{D}_{intra}(R, R') \triangleq \mathcal{D}(R, R')$$

The intra-distance of a PUF carries the notion of noise on the PUF responses and is desired to be as small as possible [3]. The intra-distance can also be represented as the fraction of different bits over responses of equal length n, which will subsequently be referred to as the *error-rate* of a PUF:

$$\mathcal{E}(R, R') \triangleq \frac{\mathcal{D}(R, R')}{n}$$

Another property of PUFs is the so-called *inter-distance*, which carries the notion of uniqueness. It is the distance between two PUF responses of equal length n, which means $R_\mathcal{P}, R_\mathcal{Q} \in \{0,1\}^n$ that have been obtained from two different devices using the same challenge C, that is $R_\mathcal{P} = \mathcal{P}(C)$ and $R_\mathcal{Q} = \mathcal{Q}(C)$:

$$\mathcal{D}_{inter}(R_\mathcal{P}, R_\mathcal{Q}) \triangleq \mathcal{D}(R_\mathcal{P}, R_\mathcal{Q})$$

In the same manner as the error-rate $\mathcal{E}$, the inter-distance can also be expressed as a fraction over the total length of bits:

$$\mathcal{I}(R_{\mathcal{P}}, R_{\mathcal{Q}}) \triangleq \frac{\mathcal{D}(R_{\mathcal{P}}, R_{\mathcal{Q}})}{n}$$

In this matter, the intra-distance is desired to be as low as possible, whereas the inter-distance is desired to as close to 50 % as possible [3] [4].

Also, PUF responses can be *biased* towards either 0 or 1, which means that there are more zeroes than ones or more ones than zeroes respectively. The bias of a PUF response R can be represented by the so-called *fractional Hamming weight*:

$$\mathcal{W}(R) \triangleq \frac{\#\{i : R_i \neq 0\}}{|R|}$$

Here, the fractional Hamming weight gives the percentage of ones in the response, in contrast to zeroes. As the distribution of a PUF response should desirably as even as possible, with the same amount of ones and zeroes, the fractional Hamming weight should be as close to 50 % as possible.

In this paper, the error-rate $\mathcal{E}$, the bias $\mathcal{W}$ and the fractional inter-distance $\mathcal{I}$ are established as the average value over a set of N measurements, each measurement consisting of two subsequent evaluations of either the same or two different PUFs with the same challenge, the latter being the case when measuring inter-distance. These average values will be denoted in the remainder of this paper as $\mathcal{E}_N$, $\mathcal{W}_N$ and $\mathcal{I}_N$ respectively.

If all CRPs that correspond to a PUF would be known, a function $\mathcal{Q}$ could be constructed, s.t. $\mathcal{Q}(C) \approx \mathcal{P}(C)$, by which the PUF $\mathcal{P}$ could effectively be cloned. In this matter, two different types of PUFs can be distinguished, *strong PUFs* and *weak PUFs*. Strong PUFs possess a great number of CRPs, so that even if a subset of those are in possession of an adversary, new challenges can still be found for which the latter does not know the corresponding response. Therefore, the challenge-response interface of strong PUFs can be publicly accessible as, even when an adversary is able to challenge the PUF for an extensive period of time, the PUF still cannot be cloned. The implementation of strong PUFs is quite difficult however. Weak PUFs on the other hand possess only a few or even just a single CRP. This means that the challenge-response interface of this kind of PUF must be protected from being accessed by an adversary, as the latter could potentially clone the PUF otherwise. Examples for strong PUFs are so-called *ring-oscillator PUFs* or *arbiter PUFs*, whereas so-called *SRAM PUFs* constitute a typical example of a weak PUF [3].

Because of their fundamental properties, two main fields of applications exist for PUFs, which are *Identification and Authentication*, as well as *Key Generation and Storage* [4].

PUFs can be used to identify and authenticate objects, in a similar way to what can be achieved using biometry. This is due to the fact that they are unique and reproducible [4]. An authentication scenario using PUFs usually works in two phases, which are described in [3] and briefly explained in the following. For such an authentication scenario we assume two participants, the *prover*, which is the entity who wants to authenticate itself to the second party,

called the *verifier*. In this scenario, the PUF will be embedded into the verifier's hardware. Furthermore, the prover will have a unique identifier called ID, by which it claims its identity.

Enrollment phase During this phase, a subset of the CRPs of the prover's PUF are collected in a secure environment and then stored in the database of the verifier. These database entries are indexed by the ID of the prover.

Verification phase In this phase, the prover initiates authentication to the verifier by sending its ID. The verifier, in turn, searches for the database entries that are stored under the prover's ID and selects random CRP (C, R). The challenge C that belongs to the CRP in question is then sent to the prover. The prover then proceeds with challenging the PUF using C, and sends the resulting response R' to the verifier. The verifier then checks if the received response R' is *sufficiently close* to the response R, which is stored in the database. If this is the case, the prover is successfully authenticated, otherwise the authentication fails.

PUFs also present an opportunity for the secure generation of cryptographic keys, as they fulfill two important requirements: a unique and unpredictable source of randomness, as well as a form of protected memory in which keys can be stored reliably. PUFs accomplish this by exploiting intrinsic, device-unique randomness, in the form of PUF responses, which can be used for key generation. As PUF responses can be measured repeatedly, with a desirably small intra-distance between them, and the same key can be generated again and again without the need for protected *non-volatile memory* (NVM) [3].

PUF responses by themselves, as already indicated, cannot be reliably reproduced due to their appertaining random noise, and this random noise may also be non-uniformly distributed. As cryptographic keys rely on both of these properties however, a mechanism to realiably extract uniform random data from a PUF response is needed. This can be accomplished by using *fuzzy extractors*, and such extractors are described in detail by [5]. The detailed operation of such fuzzy extractors will not be discussed in further detail at this point, for sake of brevity.

Analysis of SRAM PUFs on existing ICs

A type of PUF that is frequently found in commodity hardware is the aforementioned *SRAM PUF* that works by exploiting manufacturing variations in *Static Random Access Memory* (SRAM) components. On power-up, the memory cells of the SRAM, each holding a binary digit, flip into a random state of either 0 or 1. This state is dependent on physical asymmetries in the SRAM cell. SRAM PUFs have the advantage that SRAM components are already present in a wide range of existing hardware, without the need to be separately introduced into the architecture.

In order to use an SRAM component for the implementation of an SRAM PUF, it is necessary to asses whether the SRAM component in question exhibits the desired quality in terms of error-rate, bias and inter-distance. In [6] and [7], the SRAM components present in several microcontroller units (MCUs) from different vendors have been analyzed regarding these properties. An overview

of the results of this analysis, in terms of error-rate and bias, is shown in table 1 below. For some models, more than one device was analyzed. Likewise, the number of responses taken from the several devices varies due to technical reasons. In both cases, the indicated values mean the average over the given quantity of devices and number of responses.

Platform	**Quantity**	N	$\mathcal{E}_N$	$\mathcal{W}_N$
Arduino Mega	1	10	8.240	69.140
Arduino Uno	4	10	6.467	68.945
ESP32	2	100	6.501	63.562
ESP8266	2	20	6.225	50.019
Genuino 101	1	10	7.118	33.437
Genuino MKR1000	1	10	3.906	55.078
Nordic nRF51	1	10	1.822	54.609

Table 1: Properties of different SRAM PUF responses

From the above values it can be seen that the bias of all Arduino devices, as well as that of the Genuino 100 and the ESP32 is fairly high and needs to be improved, whereas other devices have a fairly even distribution of zeroes and ones, with the ESP8226 being very close to 50 %. The intra-distances of all devices are fairly low, with the Nordic nRF51 exhibiting an intra-distance as low as 1.822 % and as high as 8.240 % for the Arduino Mega.

The average fractional inter-distance $\mathcal{I}_N$ between different devices has also been analyzed. A detailed listing of these values is not given for sake of brevity however, and it is deemed sufficient to say that, on average, an inter-distance $\mathcal{I}_N$ of about 50.598 % can be established between all the above devices, with an upper bound of 60.859 % and a lower bound of 42.578 %. This is fairly close to the desired value of 50 %.

These results indicate that potential for PUF applications indeed exists on several analyzed platforms that represent commodity hardware and are in line with other research performed in this domain, e.g. in [8].

As SRAM PUFs are weak PUFs however, it has been concluded in [6] that this kind of PUF, although easily available, suffer from the issue of how to properly protect the PUF response. Such protection is not found to be given in most analyzed device models, which means that the PUF cannot be used for security critical applications *as is* and special precautions must be put into effect, in order to protect only available response from being read out by an adversary. Another potential issue with weak PUFs that has been found in [6], is that measures which introduce a level of security into the system that is sufficient to protect the PUF response, can potentially eliminate the advantages of a PUF, as the respective functionality could then also be achieved by other means. The PUF response, as well as the non-volatile memory (NVM) present in a device could, for example, be protected by preventing the reprogramming of an IoT device, after the required software has been flashed. In such a case it would be possible to store a cryptographic key in NVM and not to use a PUF at all. Also, research into practical strong PUFs is thus encouraged, as such

would enable to exploit the full potential of PUF technology. As stated earlier however, this may be a difficult task to accomplish.

Identification and Authentication using PUFs

As mentioned before, the identification and authentication of objects are a main field of application for PUF technology. In the following, an authentication scheme for resource-constrained embedded devices that has been developed in [9] will be presented, which is especially designed to be employed in the domain of the *Internet of Things* (IoT). In line with what has been described in the introduction, the authentication scheme consists of two entities, the verifier and the prover. In this case, the target IoT device is the prover that wants to authenticate itself to an *authentication authority* (AA), which represents the verifier.

In the course of this authentication scheme, not only the authenticity of an IoT device is proven to the AA, but also the integrity of the device's bootloader. The section in the NVM of a device that contains the code of the bootloader will in the following be called *bootloader section* (BLS). For the authentication scheme, the AA is equipped with a database, in which hash values of a device's BLS can be stored. In the following, it is assumed that the AA, prior to the enrollment phase, already possesses the respective hash values of all relevant devices.

During the enrollment phase the IoT device evaluates its SRAM PUF and sends the resulting PUF response R to the AA. In turn, the AA sends its public key PK to the IoT device. The AA stores the received PUF response in its database and the IoT device stores the received public key in NVM. As previously stated, this process has to take place in a secure environment and is illustrated in figure 1.

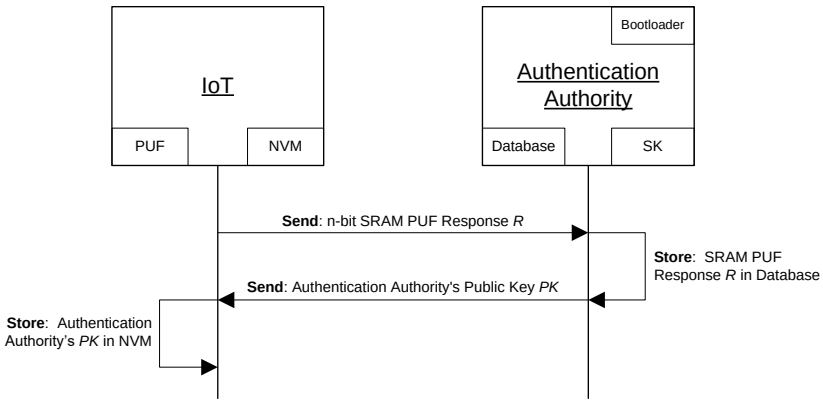

Figure 1: Enrollment in Secure Environment

During the evaluation phase the IoT device computes a so-called *PUF-based Device Identifier* or $PbDI$. This is done by computing a hash of its BLS and concatenating it with a newly obtained PUF response R' and by encrypting the result with the AA's public key PK. The $PbDI$ is then sent to the AA, which

decrypts it using its secret key SK. The AA then compares the received PUF response R' to the PUF response R that is stored in its database and decides if they match. As PUF responses are subject to noise, this match will most likely never be exact and the decision is rather based on whether the responses are sufficiently similar. Whether this is the case or not can be determined by defining an upper limit to the Hamming distance between the two responses, t, such that $\mathcal{D}(R, R') \leq t$ must hold true. The two PUF responses match if this check succeeds and mismatch otherwise. In addition, the hash of the BLS that was received from the IoT device is checked against the one stored in the database of the AA, which must be equal. This is done in order to evaluated the integrity of the device's bootloader. If both conditions are met the device is successfully authenticated. If one of the two conditions does not hold, the authentication fails. This process is shown below in figure 2.

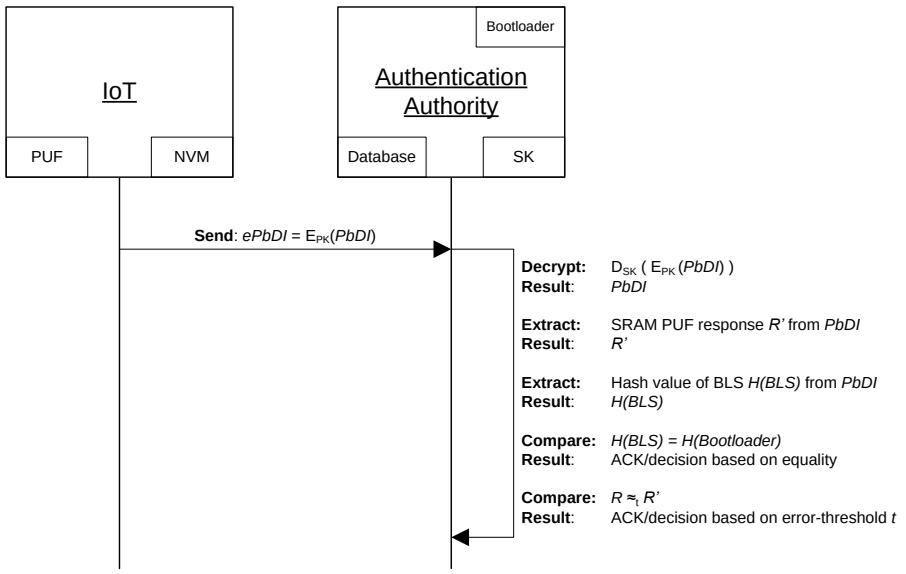

Figure 2: Evaluation in the Field

In addition, the process above is executed solely in the boot stage of the device and it is assumed that no other instructions can be executed parallel to those in the BLS. At the end of the boot stage the cells of the SRAM are initialized and the hash value of the BLS is erased from NVM. Because of this, the response of the SRAM PUF and other sensitive information are no longer present on the device after the boot stage. This protects the PUF response from being read out by an attacker, unless modifications are introduced into the BLS to change the behavior of the bootloader. In this case however, the integrity check of the bootloader that is performed during the evaluation phase must fail.

Alternatively, the bootloader can be written to *One-Time Programmablee* (OTP) memory, which can be written exactly once. This is done during the enrollment phase, and the bootloader cannot be modified afterwards in a meaningful way by neither an adversary nor a legitimate party. The evaluation phase differs in this case as the $ePbDI$ only consists of the encrypted PUF response and no hashing of the BLS or integrity check for the same is required.

Another variation of this scheme assumes that a target IoT device is capable of operating without a flashed bootloader. In this matter, the bootloader is only flashed to the device when authentication is needed and erased afterwards. During the evaluation phase, the NVM contents of the device are erased and the bootloader is flashed to the device by the AA. During the boot stage the *ePbDI* is computed as above and sent to the AA. In this case also no hash of the BLS is required.

The developed scheme comes with several benefits, such as economic advantages. It can be issued as a patch or update to existing devices. Another benefit for manufacturers of IoT devices is that they do not have to deal with the generation of unique identifiers. This eases the potential for security issues at the site of manufacturing. Furthermore, the scheme enables intrusion detection to some extent, as modifications to the bootloader or to the public key of the AA will cause the authentication process to fail. It has been concluded that all variations of the scheme provide resilience against eavesdropping, modification of hardware and against the cloning of identifiers, as well as against the injection of malicious instructions into the bootloader. The variations of this scheme that employ hashes or OTP memory respectively, additionally provide protection against spoofing, in which an attacker attempts to impose the AA. The scheme is not without weak points however, as it does not protect against playback attacks for instance. In such a case, an attacker intercepts the *ePbDI* during the authentication process and exploits it to successfully authenticate itself to the AA, despite being a completely different entity. However, this attack can be coped with, for example, by introducing time stamps into the scheme, such that the *ePbDI* is unique every time it is sent.

It should be noted that other security anchors such as *Trust Platform Modules* (TPMs) or *Hardware Security Modules* (HSMs) exist which potentially provide a higher level of security than the proposed scheme. Such security anchors raise the complexity and cost of the product however, and the proposed scheme has been designed with low cost solutions in mind that would otherwise have no or very little security at all.

Conclusion

In this document, a brief overview has been provided over research activities in the field of Physically Unclonable Functions (PUFs) that reach into the domain of the Internet of Things (IoT) and their outcomes have been summarized. It has been concluded that the practicality of PUF technology, especially when it comes to SRAM PUFs, is still an open topic and enabling the sophisticated application of the former in the IoT sector still needs further research. In the future, further implementations are going to be analyzed and resource constrained devices will be tested in order to assess the potential applications of PUFs within the IoT. Furthermore, research will be conducted on the the application of PUFs for the protection of intellectual property (IP), as well as the combination of PUFs and block chain technology.

Acknowledgments

This work was partially supported by the European ECSEL Joint Undertaking under the grant agreement 661796 ADMONT "Advanced Distributed Pilot Line for More-than-Moore Technologies" and by the Austrian Ministry for Transport, Innovation and Technology under the framework of "IKT der Zukunft" with the FFG grant agreement project 848732. Furthermore, this work was also partially supported by the European Union's Horizon 2020 research and innovation programme under the grant agreement 700002 ALFA "Advanced Low Flying Aircrafts Detection and Tracking".

References

[1] Definition of Internet of Things in English by Oxford Dictionaries. `https://en.oxforddictionaries.com/definition/internet_of_things`. (Accessed on 09.07.2018).

[2] Gartner, Inc. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. `https://www.gartner.com/newsroom/id/3598917`, February 2017. Accessed on 09.07.2018.

[3] Roel Maes. Physically Unclonable Functions: Concept and Constructions, pages 11–48. 01 2013.

[4] Roel Maes and Ingrid Verbauwhede. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions, pages 3–37. 10 2010.

[5] 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, March 2008.

[6] Felix Stornig. Security for the Internet of Things by Physically Unclonable Functions. Master's thesis, Alpen-Adria-Universität Klagenfurt, May 2018.

[7] Oleksandr Tomashchuk. Feasibility Study of SRAM PUF Implementation on IoT Devices. Master's thesis, Carinthia University of Applied Sciences, June 2017.

[8] Vincent van der Leest, Ruben Niederhagen, et al. PUFFIN D2.1. `http://puffin.eu.org/PUFFIN_D2.1.pdf`, September 2013. (Accessed on 09.07.2018).

[9] Martin Deutschmann, Lejla Iriskic, Sandra-Lisa Lattacher, Mario Münzer, and Oleksandr Tomshchuk. A PUF Based Hardware Authentication Scheme for Embedded Devices. Technical report, Technikon Forschungs- und Planungsgesellschaft mbH, Burgplatz 3a, 9500 Villach, Austria, August 2018.