A PUF Based Hardware Authentication Scheme for Embedded Devices

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

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

Abstract. This paper describes an offline authentication scheme for resource-constrained embedded devices designated for applications in the Internet of Things (IoT). The novelty of the work is the combination of SRAM based PUF structures and hash values of the bootloader to derive a device unique fingerprint. We propose the usage of asymmetric encryption to secure the communication path. On top of the description of the generic concepts, we also present alternative variations using additional components such as one-time programmable (OTP) memory. The schemes are thoroughly evaluated against a set of relevant attack scenarios.

Keywords: PUF, bootloader, IoT, hash, authentication, attacks

1 Introduction

The Internet of Things (IoT) is going to be one of the primary drivers of the digital transformation that society will undergo in the coming years. The Oxford dictionary defines the IoT as an interconnection via the internet of computing devices embedded in everyday objects, enabling them to send and receive data[1]. This definition clarifies the fact that embedded devices are one of the biggest fundamentals in the IoT architecture. The main goal of IoT systems is to improve productivity, efficiency and customer satisfaction while keeping all the data and information secure. Security is thus one of the most challenging areas when it comes to designing IoT systems. As IoT devices can be subject to substantial constraints, such as low processing and battery power, small memory, as well as the lack of a graphical user interface, the implementation of sophisticated security measures may pose a significant challenge.
There exist various types of protocols to achieve entity authentication, mutual authentication, device identification and similar. It is however essential to have a clear understanding of the practical requirements, the use case scenario and objectives of each involved party, before one can attempt to effectively develop a suitable security solution.

[1] https://en.oxforddictionaries.com/definition/internet_of_things

1.1 Our Contribution and Structure of the Paper

In this paper we propose an Static Random Access Memory (SRAM) Physically Unclonable Function (PUF) based "offline" authentication scheme with the goal to verify if an IoT target device is genuine, i.e. not a counterfeited product. Offline means in this context that the verifier has physical access to the device and thus exerts control over its power supply and is able to directly connect to its wired interfaces. On top of the general description of our scheme we also propose suitable alternative implementations. The developed protocols take into account limitations due to constrained IoT environments and are evaluated against most relevant attacks.

Within this chapter, a short introduction as well as a summary of related work is given. In Chapter 2 we provide an overview of the relevant background and the most important notations and definitions. The core part of this paper is presented in Chapter 3, where we provide an overview of the proposed scheme and describe possible alternatives. In Chapter 4 we highlight the resistance of our scheme against relevant attacks. The paper ends with a conclusion and summary in Chapter 5. In addition we provide a set of relevant definitions in the Appendix.

1.2 Related Work

2001 Pappu [10] introduced so called Physical One-Way Functions in his PhD thesis. The idea was based on obtaining an optical response pattern from shining a laser on an epoxy wafer. Later, in 2002, it was Gassend et. al [3] who took up the concept and translated it into Silicon Physical Random Functions. He made use of random variations caused by the manufacturing process to uniquely identify integrated circuits (ICs). Lee et al. [7] described in 2004 the Arbiter PUF and in 2007 Guajardo et al. [4] mentioned the first time so-called SRAM PUFs. Later various other concepts where introduced, such as the Butterfly, D flip-flop or Buskeeper PUFs. For an elaborate overview of PUF types and concepts please be referred to [9]. For other state-of-the-art research on PUF notations, comparison of post-processing and PUF related schemes please read the PhD theses of Roel Maes [8] and Jeroen Delvaux [2], which we consider as very complete.
Many PUF based authentication schemes published in literature consider strong PUFs as suitable candidates. Jeroen Delvaux pointed out in his PhD thesis [2] that those schemes are being proposed quite loose and unconnected. Koeberl et al. describe in [5] a device authentication scheme based on SRAM PUFs. The paper is based on a predecessor paper, namely "Practical device authentication scheme using SRAM PUFs" [6]. Both papers share the goal to build a simple device authentication scheme utilizing SRAM PUFs. There are however also clear differences to our scheme. First we do not perform any post-processing (e.g. error correction) on the constrained device, as such processes can be quite expensive. This step is foreseen on the more powerful evaluator side. Further, there is no resistance against physical attacks provided in the papers of Koeberl et al. In

our paper we clearly assess the robustness of the scheme.
Intrinsic ID[2] published different white papers and product briefs, such as *Quidikey*, *Broadkey* or *Citadel* where they utilize SRAM PUFs for device authentication. Very often practical details are however not exposed, which makes it hard to thoroughly verify their schemes.

2 Background and Notation

A PUF provides, when queried with a challenge C a response R. In this paper we only focus on digital PUFs, i.e. C and R can be understood as binary strings of length n. In literature there is a differentiation between so-called *strong* and *weak* PUFs. Guajardo et al. define a PUF being *strong* in [4] if the number of challenge-response pairs (CRPs) is exponentially large. In other words, the PUF has so many CRPs such that an attack (performed during a limited amount of time) based on exhaustively measuring the CRPs only has a negligible probability of success [4]. If the number of CRPs is rather small the PUF is called *weak*. Within this paper we focus on weak PUFs only.

Due to their physical nature, PUFs are inherently noisy. The consequence is that a PUF output R measured multiple times on the same device, will differ in its value slightly, depending on the technology and framework conditions. This distance, expressed by the relative Hamming distance is referred to as the *intra-device distance*.

One essential property of PUFs is that their responses should not be biased, as this has a negative impact on the overall entropy. In other words, the probability of the occurrence of zeros and ones should be equal. A common approach to determining this property is the calculation of the *Hamming weight*, defined as the number of bits that are non-zero. Often the fractional Hamming weight is used, where the weight is divided by the number of bits.

For a particular challenge, the *inter-device distance* between two different PUF instantiations is defined as the distance between the two responses, resulting from applying this challenge simultaneously to both PUFs. It is therefore a measure of the uniqueness between PUFs, which indicates how easy it is to distinguish or identify different devices. For uniqueness, it is desirable to have an (relative) inter-device distance close to 50%, which means that on average half the bits prefer a different response bit value.

For many applications and in particular for PUF-based key generation it is important to accurately estimate the entropy of a PUF response. Entropy is a function of the distribution of a random variable and expresses the amount of uncertainty one has about the outcome of the random variable. In this paper we

[2] `https://www.intrinsic-id.com/`

do not dig into the aspects of post processing (privacy amplification and information reconciliation) and entropy estimation. We acknowledge the importance of such assessments when it comes to implementations, our paper is however focusing purely on the conceptional view and not considering implementation details. On top of the PUF response R we use also the hash value H of the bootloader section (BLS) to derive a device specific fingerprint. Throughout this paper, we refer to the PUF response R, in combination with the bootloader hash value H (in case of hash-based variation), as the PUF-based Device Identifier, short the $PbDI$.

3 Hardware Authentication Using SRAM PUFs

Within this chapter, we first describe potential applications for our solution, as well as the authentication scheme. The primary field of application is the medical sector, including appliances such as body-worn vital sign monitors but also smart homes. Beside the solution's field of application, important environmental conditions and assumptions are presented in order to make a clear statement about the premises the authentication scheme is built on. Afterwards, the scheme's authentication procedure is described in detail. In addition to the primary hash-based variation, two alternatives are described as well. This chapter concludes with a summary of the key aspects of our solution.

3.1 Field of Application

Industry is the main driving force of the Internet of Things. First IoT solutions were implemented in this sector in the form of distributed sensor networks as a next step after Machine to Machine (M2M) communications. Due to its use case scenarios, it is an obvious fact that IoT security requires a multi-layered approach. From a device point of view, it should be considered at the design phase that keeps hardware, software, and data secure through their entire life cycle.
IoT systems can be found among others in the healthcare domain, such as presented in [1]. Commonly, the main idea of such systems includes simple and centralized monitoring of patients independently of their location. With the help of a discrete, wireless, non-intrusive body-worn monitor, healthcare providers, physicians and authorised personnel are allowed to monitor the key biometrics outside the usual clinical setting, while patients go about their daily lives. On top of the medical usage of IoT, a typical example of IoT implementations in real life is the smart home. There is no exact definition what a smart home is, but in general, this is a combination of connected devices with some kind of actuators. These devices can be controlled remotely (e.g. by a smartphone) or they can have a pre-programmed instructions.
However, due to the fact that uncontrolled access to these systems by third parties - no matter if within healthcare or the smart home domain - can lead to data leakage, the security and data preserving has great importance. The proposed

use case within this paper is an offline authentication scheme for IoT devices. Within the context of this use case we consider the resources of these IoT devices to be limited and it is assumed that attackers can obtain physical access to them, as well as to their interfaces. In this matter, a trusted party aims to evaluate, whether an IoT device is legitimate and still behaves in it's originally intended manner.

3.2 Environmental Conditions and Assumptions

The scheme proposed in the next section was designed to be used in environments where offline authentication can take place and attackers have the capability to physically tamper with the device. It is necessary to mention that we do not consider security of the evaluator's side, which is referred to in the following chapters as the Authentication Authority, short the AA. The evaluation of security of databases, decision taking mechanisms, etc. have to be done separately as they heavily depend on implementation-specific properties. Also, this scheme does not cover security of bootloader/firmware/software over-the-air (OTA) updates. In case OTA updates are required, their security, as well as influence on the proposed scheme, should be assessed separately.
The scheme is based on the following assumptions:

- No other instructions can be executed in parallel to the bootloader instructions;
- Bootloader instructions are the very first to be executed after power on;
- Reading-out data from NVM (e.g. memory dump) is only possible for the attacker if one has physical access to a target device.

3.3 Proposed Scheme

In order to avoid the malicious modification of hardware, we use resources, such as SRAM that are already built into most commodity embedded devices. The PUF-based identity is defined as the start-up values of the SRAM cells, which are obtained during the boot stage. In addition to the PUF-based identity, the hash value of the non-volatile memory sections, in which the bootloader program code is stored, a so-called bootloader section (BLS), is required. By computing the BLS hash value during boot of the IoT device, the genuineness and integrity of the bootloader program code can be verified.
Overall, the scheme is composed of two entities, namely the target IoT device itself and the authentication authority, where the evaluation and verification of the IoT device happens. Therefore, at least the resource-intensive post-processing of the sensitive SRAM PUF output is outsourced from the target IoT device, whilst reducing the attack capabilities and saving in turn resources of the IoT device.
As mentioned above, the scheme itself is divided into two phases, the enrolment and the evaluation phase. The enrolment phase is only performed once in a secure environment before the IoT device is deployed in the field. The evaluation phase is performed in the field, whenever a verification is needed. Both phases

are described in detail within the subsequent paragraphs.

The **Enrolment Phase** consists of two main tasks, the provision of the start-up values of the target IoT device's SRAM cells (SRAM PUF response) to the authentication authority and the transmission of the authority's public key PK to the IoT device stored in the NVM. Therefore, the enrolment process has to be done in a secure environment. Since all the resource-intensive procedures are done on the AA's side, the solution is more lightweight for low-resource IoT devices and contributes to wide usage opportunities. Besides holding the n-bit SRAM PUF response, the authority already possesses the original bootloader program code as well as its hash, hence both do not have to be transmitted during the enrolment phase.

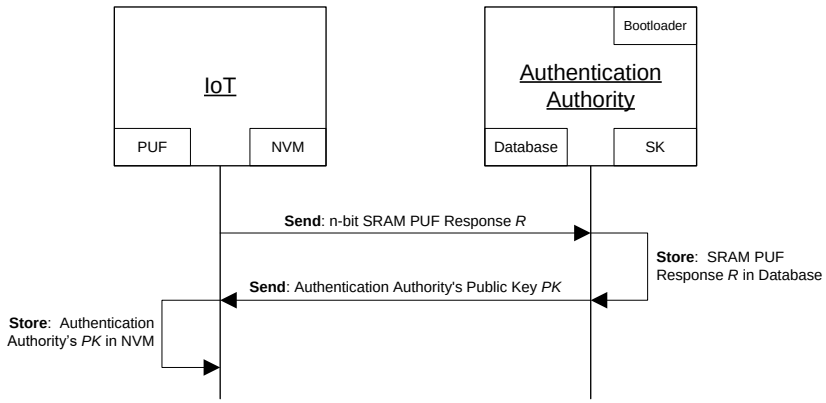

Fig. 1: Enrolment in Secure Environment

In detail, the Enrolment Phase comprises of the following steps:

Enrolment Phase

1: **Boot** target IoT device.
2: **Send** n**-bit response** R to authentication authority.
3: **Store received response** R in authority's database.
4: **Send authority's public key** PK to IoT device.
5: **Store received public key** PK of authority in IoT's non-volatile memory (NVM).

During the **Evaluation Phase**, the PUF-based Device Identifier, the $PbDI$, of the target IoT device is provided to the authentication authority. As mentioned before, the $PbDI$ is composed of the SRAM PUF response R' and the hash value of the bootloader memory section $H(BLS)$. For a secure transmission, $PbDI$ is encrypted with the help of the AA's public key PK, which was stored during the enrolment phase. We denote the encrypted $PbDI$ as $ePbDI$. In order to start the verification procedure and evaluate the integrity and original behaviour of the target IoT device, the AA has to decrypt the $ePbDI$ first by using its secret key

SK. Afterwards, the AA has to extract R' as well as the hash value $H(BLS)$ in order to perform comparison checks between the entries of the response-database and the hash value $H(Bootloader)$ of the genuine bootloader program code. When comparing R' against R, a certain number of bits in R' will not match the ones in R, due to the noise occurring in R'. We describe the number of bit errors present as the Hamming Distance between R and R', $HD(R, R')$. Furthermore, a threshold t is defined, by which the evaluation phase will either succeed or fail if more than t errors are detected, thus resulting in $HD(R, R') > t$. We subsequently note $HD(R, R') \leq t$ as $R \approx_t R'$. Beside the comparison of the responses, $H(BLS)$ is compared against $H(Bootloader)$ in order to check for equality, which is subsequently noted as $H(BLS) = H(Bootloader)$. As a result of both comparison checks, the genuineness and integrity of the target IoT device will be proven in case of a positive result, otherwise these properties will be invalidated. It should be kept in mind, that the procedure is purely done in the boot stage of the target IoT device and it is required to reboot the device, if it is in power-on state, or boot it for running the evaluation phase.

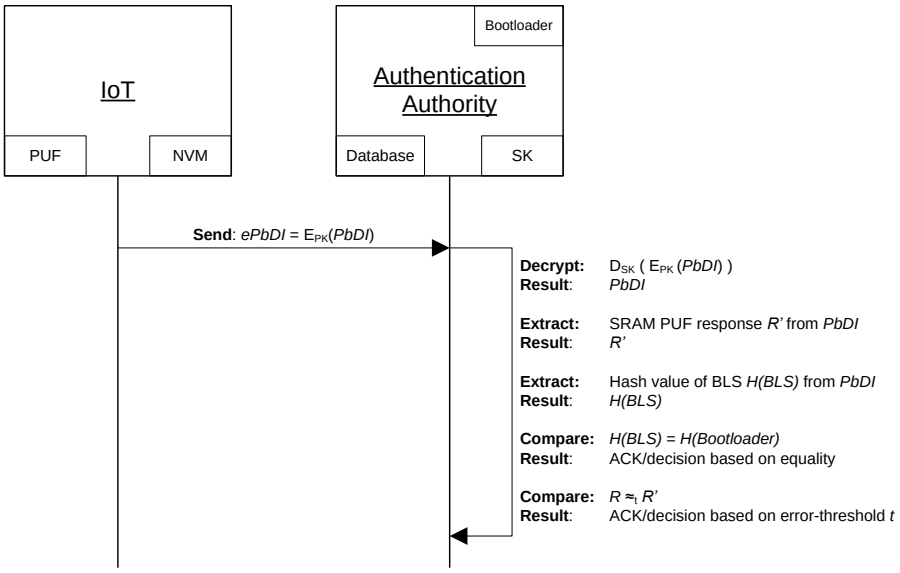

Fig. 2: Evaluation in the Field

In detail, the Evaluation Phase comprises of the following steps:

Evaluation Phase

1: **Establish connection** via suitable interface and initiate boot/reboot of device
2: **Compute PUF-based Device Identifier** $PbDI$, composed of SRAM PUF response R' and hash value of BLS:
   – Compute hash value of bootloader section (BLS).
   – Keep $H(BLS)$ in SRAM cache, which is not used for PUF response generation.
   – Get SRAM PUF response R'.
3: **Encrypt** $PbDI$ with the AA's public key PK, resulting in $E_{PK}(PbDI)$, respectively in $ePbDI$.
4: **Send** $ePbDI$ to authentication authority.
5: **Process** $ePbDI$ on authority side:
   – Retrieve and decrypt $ePbDI$ by means of the AA's secret key SK.
   – Retrieve/extract actual SRAM PUF response R' and hash value $H(BLS)$ of the BLS.
6: **Perform comparison checks** of original SRAM PUF response R and actual provided SRAM PUF response R' by using corresponding response-database entries.
7: **Perform comparison checks** of hash value of the original bootloader program code and provided hash value of the BLS.
8: **Decide**, whether the device is authentic, legitimate and unaltered or not.

Right after the boot stage, the initialization of the SRAM cells has to be performed as a part of the bootloader before the starting of the firmware and/or the main software. As a result, the sensitive information, respectively the PUF response, is no longer available and cannot be exploited by malicious firmware or software. Besides the initialization of the PUF-related SRAM cells, also the hash value of the BLS has to be deleted to limit the attacker's capabilities. For either of the two additional variations introduced in the following, the explanations as well as steps of the verification procedure are closely related to our primary variation above, however there are slight differences, which are pointed out in detail below.

3.4 Alternative Variations of the Scheme

It is crucial for the verifier to know whether the bootloader of a device is genuine and trusted as it is dealing with the fundamental security anchors of the target device. One of the key elements of the scheme described above is indeed the computation of the hash value of the corresponding bootloader section, which serves as a proof of genuineness and integrity for the bootloader. However, there are alternative solutions which can provide these proofs as well, and they are going to be described in the next sections.

OTP-based alternative variation: In this variation the bootloader is written to an OTP in order to provide a proof of bootloader genuineness and integrity. The additional requirement for this variation of the scheme is that a target device has OTP memory. Information can be written to this memory only once,

which is done in the enrolment phase, but "read" operations can be performed without limitations. In case that this requirement is satisfied, the bootloader will be flashed to the target IoT device in a secure environment before sending the latter to end users. This guarantees that neither attacker nor verifier can modify it in a meaningful way. The steps of evaluation phase for this variation of the scheme are as follows:

OTP-based variation of the scheme

1: **Establish connection** via suitable interface and initiate boot or reboot of device;
2: **Device enters bootloader stage** and performs the following operations:
   – Get PUF response R';
   – Obtain $ePbDI$ via encryption of R' with public key PK;
   – Send $ePbDI$ to verifier;
   – Delete $ePbDI$ as well as R' from the memory of device;
   – Initialize SRAM section which was used for R' generation;
   – Proceed with starting firmware and/or software;
3: **Process** $ePbDI$ on AA side:
   – Receive and decrypt $ePbDI$;
   – Perform comparison checks of original SRAM PUF response R and actual provided SRAM PUF response R' by using response-database entries;
4: **Make threshold-based decision**, whether the device is authentic, legitimate and unaltered or not.

One of the most important properties of this variation of the scheme is that it cannot be applied to devices which do not offer programmable OTP. Furthermore, using this scheme does not allow for bootloader updates to occur. Nevertheless, firmware and software updates are still possible. If there is a strong need in bootloader updates, they can potentially be enabled by splitting the bootloader into two or more stages. In this case, the first bootloader stage is responsible for all processes related to authentication and subsequent stages for all the other processes. Here, the first stage has to be written to OTP. All instructions of subsequent bootloader stages may then be stored in NVM.

Flashing-based alternative variation: This variation of the scheme is based on the requirement, that applications of the target IoT devices can operate normally without a flashed bootloader. This requirement is easier to satisfy, than the ones from the previously described variations, as it neither does not need additional hardware nor software libraries for normal functioning. When authentication needs to take place, the verifier erases all the content of the memory of the target device, verifies it via flashing the bootloader and performing its instructions, erases the bootloader and, as the verifier is in possession of the application code - flashes the application in order to bring the device back to normal functioning. The steps of the evaluation phase for this variation have the following structure:

Flashing-based variation of the scheme

1: **Establish connection** via suitable interface, erase all content of memory of the device and flash bootloader;
2: **Device starts bootloader stage** and performs the following operations:
   – Get PUF response R';
   – Obtain $ePbDI$ via encryption of R' with public key PK;
   – Send $ePbDI$ to verifier;
   – Delete $ePbDI$ as well as R' from memory of the device;
   – Initialize SRAM section which was used for generation of R';
   – Delete bootloader;
   – Flash application to bring device back to normal operation;
3: **Process** $ePbDI$ on AA side:
   – Receive and decrypt $ePbDI$;
   – Perform comparison checks of original SRAM PUF response R and actual provided SRAM PUF response R' by using response-database entries;
4: **Make threshold-based decision**, whether the device is authentic, legitimate and unaltered or not.

An interesting point is that this variation of the scheme is compatible with bootloader, firmware and software updates and can be issued as a software update for existing devices which meet all necessary requirements.
Another property of this variation is that an attacker unnoticeably can obtain an SRAM PUF response of the target device. It can happen via flashing a malicious bootloader for reading-out the PUF response and bringing the device back to normal functioning afterwards. This is not a critical point, as even if an obtained PUF response is used for cloning, such a clone will always send a PUF response which belongs to its SRAM (if available). Such behavior is possible due to the fact that all the software which may contribute to counterfeiting is deleted on the first stage of the evaluation phase.

3.5 Key Aspects

The availability of the SRAM PUF and genuineness as well as integrity of the bootloader represent the foundation of the scheme. Besides this, we identify three key aspects of the scheme in respect to security, which are listed below:

Key Aspect 1: Proof of the genuineness of the bootloader, which is specific for every variation of the scheme (hash-, OTP- and flashing-based variation). It is required for ensuring the genuineness of the bootloader and assuring that the device sends its original SRAM PUF response to the verifier.

Key Aspect 2: Encryption of the data with the authentication authority's public key during the execution of the bootloader. It is needed for hardening security of the SRAM PUF as well as for adding resistance to cloning attacks as eavesdropped data may be easily exploited. In the proposed scheme, asymmetric encryption is used in order to reduce the number of potentially

successful attack scenarios as well as to support simpler key management. In case the security of the communication channel is guaranteed in another way, it is not obligatory to use encryption in this scheme.

Key Aspect 3: Sending of the $ePbDI$ during the execution of the bootloader. In case the $ePbDI$ is not sent during the bootloader stage, it cannot be deleted before the subsequent stages commence. This may present an opportunity for the attacker to exploit the $ePbDI$ during the execution of firmware and/or software.

As a result, all the critical steps of the evaluation phase are done in the bootloader stage of the target IoT device. Therefore, its instructions cannot be manipulated by other software. Even if the firmware or software is infected by malicious code, it cannot influence the bootloader as they are being executed during later stages. In case it is not possible to put trust into bootloader stage, the added value of entire scheme is questionable.

4 Assessment and Discussion

In this chapter we aim to review the security requirements and how our scheme contributes to them. Any variation of the scheme provides protection from modification of hardware and eavesdropping attacks, as any device, which is applying our concept, due to the guaranteed integrity of the bootloader, will send its original SRAM PUF response in an encrypted form.
The scheme's ability of offline authentication contributes to the verifier's awareness of the very first step of the procedure - the reboot of the target device. In case of online authentication it is hard for the verifier to be sure whether the device has rebooted or not. This is not the case if physical access is granted, since the verifier can control the power supply of the device and improve assurance whether authentication takes place during the execution of bootloader instructions. In case an additional energy source is connected to the device, it does not remain undetected as the verifier has physical access. If one can solve a problem of awareness regarding the reboot process, the described scheme can be used in online cases as well.

The main motivation in using asymmetric cryptography in this scheme is to protect the data, which is transmitted to the AA, from eavesdropping[3] and spoofing, as only the AA can decrypt the $ePbDI$. The benefit of this kind of cryptography is that secure key storage, which cannot be found in most of the resource constrained embedded devices, is not an obligatory requirement. From the practical point of view, we recommend to use lightweight asymmetric cryptography (e.g. NTRU) in order to keep resource usage of devices on a low level. Symmetric cryptography may still be used in such conditions, but only in case the security of key storage is guaranteed. Also, in this case the overall added value of the scheme may become questionable as the unique identifier may be secured the same way as the symmetric key.
The way of accessing the device is also crucial when distinguishing attackers who intend to manipulate the device. As a result, they can be divided into two groups: those who have physical access and those who do not have it. The difference between the two groups of attackers lies in the types of attacks which they can perform. The absence of attackers' physical access provides protection from identifier cloning, modification of hardware and similar attacks. In combination with the added value of the scheme, specific requirements can be met in various conditions. Table 1 clearly depicts the discussed variations as well as the case of identifier (ID) storage in NVM, and their resistance (✓), weakness (✗) or inapplicability (❍) to several feasible attacks, whilst considering the attacker's device access method. Storage of the identifier within the NVM was considered the non-secured case and therefore this option is shaded in grey in Table 1.

[3] The definition of eavesdropping as well as other attacks are introduced in the Appendix.

Table 1: Protection from Attacks

	Attacker **has** physical access				Attacker **has not** physical access			
	Store unique ID in NVM	**Hash-based variation**	**OTP-based variation**	**Flashing-based var.**	Store unique ID in NVM	**Hash-based variation**	**OTP-based variation**	**Flashing-based var.**
Eavesdropping	×	✓	✓	✓	×	✓	✓	✓
Malware Injection into Bootloader Instructions	×	✓	✓	❍	✓	✓	✓	❍
Bootloader Removal	×	×	✓	❍	✓	✓	✓	❍
Spoofing	×	✓	✓	×	×	✓	✓	✓
Playback Attack	×	×	×	×	×	×	×	×
Modification of Hardware	×	✓	✓	✓	❍	❍	❍	❍
Identifier Cloning	×	✓	✓	✓	❍	❍	❍	❍

Table 1 illustrates the additional protection of our scheme against relevant attacks in comparison to storage of sensitive information in NVM. In our case, the sensitive data is protected by means of a device identifier based on a PUF and the bootloader. It strongly improves the security of the embedded device and significantly hardens potentially successful attack scenarios as common techniques of hijacking content from NVM cannot be applied. Nevertheless, there are even more advantages that are provided by our scheme:

- The scheme brings economic advantages as it can be issued as a patch or update for existing devices (OTP variation may require additional hardware);
- Manufacturers do not have to deal with the generation of unique identifiers which helps to prevent potential security issues that arise by processing sensitive information at manufacturing sites;
- It offers some degree of intrusion detection, as in case the public key is modified or removed, communication with the AA will be affected which will automatically lead to an unsuccessful authentication.

Another element of intrusion detection arises from the computation of hashes within the bootloader stage. If this computation is possible for a target device, firmware or software hashes may be used additionally in any variation of the scheme for contributing to overall security improvements. However, this potential advantage heavily depends on the implementation scenario.

When designing a cryptographic scheme, it is unavoidable to have a couple of weak spots, which have to be kept in mind while designing real life implementations. Situations in which an attacker can access the NVM of a target device may occur and one may be in a position to manipulate the bootloader. It is not critical for the authentication process if the attacker manipulates firmware or software as both execute after the bootloader stage. If the bootloader is modified with malicious intent, the hash value of the BLS will not match with the hash value of the genuine bootloader which is stored on the AA's side, which will cause the authentication to fail. Nevertheless, if an attacker has deep understanding of the implementation-specific architecture and bootloader instructions, he or she can perform a specific attack which may lead to sensitive data leakage. An attacker needs to make a memory dump for obtaining the BLS hash H, erase the bootloader and re-install the modified one. Afterwards the attacker can obtain R' and return the device to pre-attack stage. By doing this, it may be possible to make a clone of the target device in question. Such clone can then be used for a playback attack based on values of a BLS hash H and PUF response R', via writing a software for another device to provide stolen data as an output at the request of the verifier. It is to be noted that this attack is not feasible for the OTP-based variation.

It may also be possible to perform a playback attack by using cloning, based on a man-in-the-middle attack, without knowing the H and/or R'. The attacker may have a chance for intercepting the $ePbDI$ and exploit it for cloning of the device without knowing the $PbDI$. One may be in a position to program a device in such way that it sends the intercepted packet in encrypted format after the reboot. In such a case, the verifier will get a packet of the original device and successful authentication may take place despite the fact that it was sent by a completely different device. In order to cope with this weak spot, every packet in real life implementation should be unique. There are several ways to generate uniqueness, however we recommend to use timestamps for this kind of purposes. Also, it may be possible to use noise-based properties of SRAM PUF for this goal, but further evaluation of them is needed as this anchor has a very probabilistic nature.

Another weak spot arises from the field of public key management. It may be possible to force a device to use a malicious public key instead of the one originally provided by the AA via injecting it into NVM. This may enable stealing the H and R', and may put an attacker in the position to make a clone which sends H and R' of the previously attacked device instead of its own. This problem does not cancel the added value of the scheme as it is much harder to perform the previously described manipulations than to copy a unique device identifier from the NVM of a target device to the NVM of a clone. Also, this problem is out of scope in case a public key is written to OTP, which cannot be removed or modified (OTP-based variation). This negates the opportunities attackers have, in order to get the $PbDI$ in unencrypted form, but makes public key updates almost impossible.

When evaluating added value as well as possible application scenarios, it should be taken into account that there may exist security anchors which fit high security applications much better than the proposed scheme. Some of them, like Trusted Platform Module (TPM) and Hardware Security Module (HSM), can guarantee a higher level of security if they are properly integrated. However, the proposed scheme was designed to be used in solutions which have very poor or no security at all, and for which anchors like TPM or HSM are infeasible or uneconomic. It is important to note that we position the proposed scheme as a low cost, intermediate solution between almost no security and different, more expensive sources of security like those mentioned above.

5 Conclusion and Summary

The employment of PUF technology in security critical applications has been a heavily debated topic in the past years. Supporters praise the outstanding properties allowing for secure key storage, especially in areas where the resources are constrained. Critics point out the instability due the physical nature as well as the problem of PUF evaluability and confidence in the technology. In this paper we present a simple authentication scheme using SRAM based PUFs. The novelty of our solution can be clearly seen from the key aspects which are introduced in section 3.5. In fact, it is based on the combination of public key encryption with weak PUFs, as well as the attestation of bootloader integrity by using hash values (or its alternatives), to realize a lightweight authentication scheme. This way we aim to provide an answer to if and how PUF technology could be employed within the authentication process of IoT applications. We do not only provide a conceptual view, but also analyse against which attacks our scheme would be resistant and against which this is not the case. We do see niche areas where there are advantages in using PUFs. When it comes to high security applications and attackers with extensive resources, we however take the position that sophisticated measures, like the usage of hardware security modules may be more appropriate.
We will continue to move forward with the analysis of further implementations and will also test our solution on common resource-constrained devices, such as the *Microchip AVR*[4] micro-controllers and their well-known SoC product range *ATmega*, as for instance.

[4] Formerly known as two independent brands, namely *Microchip* and *Atmel AVR*, before company merger in 2016

References

1. BodyGuardian Heart, Preventice Technologies. http://www.preventicesolutions.com/technologies/body-guardian-heart.html, Accessed: 2017-03-21
2. Delvaux, J.: Security Analysis of PUF-Based Key Generation and Entity Authentication. Ph.D. thesis, Katholieke Universiteit Leuven, Belgium (2017), https://securewww.esat.kuleuven.be/cosic/publications/thesis-290.pdf
3. Gassend, B., Clarke, D., van Dijk, M., Devadas, S.: Silicon physical random functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. pp. 148–160. CCS '02, ACM, New York, NY, USA (2002), http://doi.acm.org/10.1145/586110.586132
4. Guajardo, J., Kumar, S.S., Schrijen, G.J., Tuyls, P.: Fpga intrinsic pufs and their use for ip protection. In: Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems. pp. 63–80. CHES '07, Springer-Verlag, Berlin, Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-74735-2-5
5. Koeberl, P., Li, J., Maes, R., Rajan, A., Vishik, C., Wójcik, M.: Evaluation of a PUF Device Authentication Scheme on a Discrete 0.13um SRAM, pp. 271–288. Springer Berlin Heidelberg, Berlin, Heidelberg (2012), https://doi.org/10.1007/978-3-642-32298-3-18
6. Koeberl, P., Li, J., Maes, R., Rajan, A., Vishik, C., Wójcik, M., Wu, W.: A practical device authentication scheme using sram pufs. Journal of Cryptographic Engineering 2(4), 255–269 (Nov 2012), https://doi.org/10.1007/s13389-012-0043-1
7. Lee, J.W., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for identification and authentication applications. In: 2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525). pp. 176–179 (June 2004)
8. Maes, R.: Physically Unclonable Functions: Constructions, Properties and Applications (Fysisch onkloonbare functies: constructies, eigenschappen en toepassingen). Ph.D. thesis, Katholieke Universiteit Leuven, Belgium (2012), https://lirias.kuleuven.be/handle/123456789/353455
9. Maes, R., Verbauwhede, I.: Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions, pp. 3–37. Springer Berlin Heidelberg, Berlin, Heidelberg (2010), https://doi.org/10.1007/978-3-642-14452-3
10. Ravikanth, P.S.: Physical One-way Functions. Ph.D. thesis, Massachusetts Institute of Technology Cambridge, MA, USA, Cambridge, MA, USA (2001), aAI0803255

List of Abbreviations

AA	Authentication Authority
BLS	Bootloader Section
CRP	Challenge-Response Pairs
ePbDI	Encrypted PbDI
HSM	Hardware Security Module
ID	Identifier
IoT	Internet of Things
M2M	Machine to Machine
NVM	Non-Volatile Memory
NTRU	Nth Degree Truncated Polynomial Ring Units
OTA	Over-the-Air Programming
OTP	One-Time Programmable NVM
PbDI	PUF-based Device Identifier
PK	Public Key
PUF	Physical Unclonable Function
SK	Secret Key
SoC	System on a Chip
SRAM	Static Random-Access Memory
TPM	Trusted Platform Module

APPENDIX

Below we list a set of definitions which are helpful for a better understanding of the authentication scheme itself as well as the chapter on attacks and evaluation.

Bootloader	A set of instructions which is performed after the unmodifiable device-specific processes which take place right after the moment when the power is switched to "ON".
Hamming distance	In the PUF context the Hamming distance between two responses, interpreted as bit strings, is a count of the number of disagreeing bits.
Intra-device-distance	The Hamming distance between the two responses, resulting from applying the same challenge twice to one PUF.
Inter-device distance	The Hamming distance between the two responses, resulting from applying a challenge simultaneously to two different PUFs.
Hamming weight	The number of bits in a string that are non-zero.
Physical access	A condition when a verifier or an attacker is in a position to control the power supply of a target device and to directly connect to its wired interfaces.
Offline authentication	Authentication which takes place in conditions when a verifier has physical access to the target device.
Online authentication	Authentication, which takes place in conditions when a verifier does not have physical access to the target device.
Target device	A device which undergoes authentication procedures by a verifier and may undergo attack procedures from an attackers' side.
Eavesdropping	An attack in which attacker tries to steal information that a target device transmits over a network.
Malware injection into bootloader	An attack in which malicious instructions are added to existing instructions of bootloader.
Bootloader removal	An attack in which an attacker by some means removes the bootloader of a target device in order to prevent it from functioning as originally intended.
Spoofing	An attack in which an attacker acts as verifier in order to obtain protected data.
Playback attack	An attack which exploits the replay of messages from some context within the intended context, thereby fooling participating sides into falsely assuming that expected procedure has been successfully completed. In our specific context, it is based on two preparatory steps: unique identifier hijacking and development of software-based clone.
Unique identifier hijacking	An attack in which an attacker obtains access to the unencrypted identifier and exploits it for malicious purposes.
Modification of hardware	An attack in which a modification or substitution of hardware components is done for malicious reasons.
Identifier Cloning	Extracting the unique identifier from a target device and injecting it into a counterfeited device instead of the original identifier of the latter.