Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT

KARIM LOUNIS and MOHAMMAD ZULKERNINE, Queen’s Reliable Software Technology (QRST) Lab, School of Computing, Queen’s University, CANADA.

The service of authentication constitutes the spine of all security properties. It is the phase where entities prove their identities to each other and generally establish and derive cryptographic keys to provide confidentiality, data integrity, non-repudiation, and availability. Due to the heterogeneity and the particular security requirements of IoT (Internet of Things), developing secure, low-cost, and lightweight authentication protocols has become a serious challenge. This has excited the research community to design and develop new authentication protocols that meet IoT requirements. An interesting hardware technology, called PUFs (Physical Unclonable Functions), has been the subject of many subsequent publications on lightweight, low-cost, and secure-by-design authentication protocols. This has turned our attention to investigate the most recent PUF-based authentication protocols for IoT. In this paper, we review the security of these protocols. We first provide the necessary background on PUFs, their types, and related attacks. Also, we discuss how PUFs are used for authentication. Then, we analyze the security of PUF-based authentication protocols to identify and report common security issues and design flaws, as well as to provide recommendations for future authentication protocol designers.

CCS Concepts: · IoT security → Authentication protocols; · Authentication protocols → lightweight authentication protocols; · Lightweight authentication protocols → PUF-based authentication protocols.

Additional Key Words and Phrases: PUFs, IoT Security, Thing-to-Thing authentication, PUF-based authentication.

1 INTRODUCTION

IoT (Internet of Things) is an emerging networking paradigm that connects an exponentially increasing number of heterogeneous devices (Things), to the Internet. It actually expands the Internet from a Machine-to-Machine (M2M) communication system to a Things-to-Machine (T2M) and Things-to-Things (T2T) communication system. This has allowed the emergence of various smart applications in many fields (e.g., smart transportation, smart agriculture, and smart healthcare). Nevertheless, this remarkable evolution of the Internet has invited cybercriminals to exploit the new infrastructure and mount devastating large-scale distributed as well as local-scale cyberattacks [1–7]. These cyberattacks have demonstrated to the entire world how catastrophic and diversified cybercrimes could be in a world where everything is perceived as a connected computer [8].

Furthermore, most of these cyberattacks, if not all, are due to vulnerabilities in the adopted authentication protocols. Indeed, as the service of authentication constitutes the spine of all security properties\(^1\), any vulnerability residing on the adopted authentication protocol will be exploited to generate security attacks that may breach

\(^1\)In most, if not all, security mechanisms, and during the execution of the authentication protocol, cryptographic keys are generally established and derived to provide confidentiality, data integrity, non-repudiation, and availability. A vulnerable authentication protocol would result in security breaches on the confidentiality, data integrity, and availability.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2022 Association for Computing Machinery.
2576-5337/2022/2-ART $15.00
https://doi.org/10.1145/3487060

Digit. Threat. Res. Pract.
the authenticity, confidentiality, integrity, and availability of the system. Moreover, the impact of cyberattacks on IoT has become more serious than it used to be in classical information systems. In fact, in the classical scenario, cyberattacks used to be an unlawful way for stealing confidential and private information and asking for ransoms by threatening their owners to reveal the information in the case there is no payment. Nowadays, the loss of human lives is becoming a common consequence of cyberattacks. For example, a cybercriminal could bypass the authentication mechanism of a hospital, gain access to a patient’s medical record, and reveal confidential information, which just leads to the patient’s privacy violation. Nevertheless, if the cybercriminal gains access to the medical record of the patient and modifies the patient’s blood type, then it becomes serious as the latter cybercrime would lead to the death of the patient. Therefore, more research is required to develop new authentication protocols that are secure and reliable-by-design, and more importantly, conform with IoT heterogeneity, and with IoT security and performance requirements.

Developing secure and reliable authentication protocols for IoT is a mature and well-known research challenge. Many research works have been conducted for this purpose. In fact, ten years after the emergence of IoT (i.e., early 2010), a large number of authentication protocols, known as lightweight protocols (or ultra-lightweight), were designed and proposed for various IoT wired or wireless infrastructures (e.g., WSNs, VANETs, and Cloud) and IoT applications [109]. These protocols relied on the application of classical cryptographic mechanisms, where some of them adopted ECC (Elliptic Curve Cryptography) [110], whereas others employed lightweight cryptographic operations, such as simple hash functions and bitwise logical operators. Notwithstanding, these protocols suffered from known security vulnerabilities that are related to the incorrect usage of classical cryptographic mechanisms, which resulted in the re-occurrence of known cyberattacks but with an IoT flavor.

There has been a remarkable attraction and convergence from the research community and the industry to adopt PUFs (Physical Unclonable Functions) as a prominent physical security technology. Important industrial cores, such as NXP, Microsemi, Intel, and Xilinx, have already implemented the technology to develop secure integrated circuits [48–51]. In the meantime, researchers have turned their attention to PUF technology to develop lightweight and secure-by-design authentication protocols for IoT applications [9–24].

PUFs are physical one-way functions constructed from the unique nanoscopic-structure of physical objects (e.g., integrated circuits, crystals, magnets, lenses, solar cells, or papers) and their reaction to random events. This intrinsic uniqueness in the structure and reaction is due to the idiosyncrasies in the manufacturing process of the objects. It allows not only the unique identification of an object but also its authentication. Additionally, it is assumed to be impossible to clone the PUF of an object (and hence the object itself), which can be perceived as a security-by-design that will prevent any possible impersonation and cloning attacks. Therefore, PUFs are considered as a novel, reliable, and prominent physical security technology to develop secure integrated circuits and lightweight authentication protocols for IoT applications.

The motivation to adopt PUFs as a reliable physical security technology by a large number of researchers and industrial cores, is based on the following facts:

- Resource-constrained devices, such as RFID-tags, can only support 3K to 5K logic gates for implementing security algorithms [100, 101]. Nevertheless, classical cryptographic algorithms require between 12K to 22K logic gates [102], which makes them not suitable for this kind of devices. In the contrary, PUFs have a relatively lower hardware cost compared to other solutions, which makes them one of the perfect candidates for implementing security on resource-constrained devices.

- Many resource-constrained devices are low-cost devices that have no physical security (i.e., vulnerable to invasion attacks). Adding physical security to such devices would increase the cost. PUFs allow implementing physical security by using the device’s own circuit without adding any new component.
Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT

Traditionally, cryptographic keys are stored on devices’ non-volatile memories (e.g., ROM, fuse, or EEPROM). These keys are generally used for authentication. This has a consequent cost for securing the non-volatile memory. With PUFs, authentication keys are no longer stored on the device. They are created and extracted on-the-fly from the device circuit.

PUFs allow conceiving unclonable circuits, which implies unclonable devices. Moreover, as cryptographic keys are not stored on the devices, the latter are deemed to be tamper-resistant.

Last but not least, PUFs have been accepted as a novel and promising candidate for implementing low-cost security of resource-constrained devices by a large number of multidisciplinary researchers and industrial cores.

The rapid convergence of adopting PUFs for authentication by the research community has resulted in many subsequent research work on lightweight, low-cost, and secure-by-design authentication protocols [9-24]. This has turned our attention to devote this research work for investigating the security of the most recent PUF-based authentication protocols.

In this paper, we analyze the security of recent PUF-based authentication protocols (published between the year of 2016 and 2020). We first provide the necessary background on PUFs, their types, and related attacks, as well as discuss how PUFs are used for authentication. Then, we analyze the security of existing PUF-based authentication protocols to identify possible attacks. Based on the identified attacks, we discuss possible security lessons to be considered by future authentication protocol designers so that newer protocol would be immune from the identified attacks.

To summarize, the contributions of this paper are twofold:

1. We analyze the security of PUF-based authentication protocols and identify possible attacks.
2. We draw lessons from the design flaws of the existing protocols and provide security recommendations for future authentication protocol designers.

The remainder of the paper is organized as follows. In Section 2, we provide an overview of PUFs, their types, and their related attacks, as well as discuss how PUFs are used for authentication. This would provide the necessary background needed for the remaining part of the paper. Section 3 reviews the design and the security of PUF-based authentication protocols. We draw lessons and recommendations for future PUF-based authentication protocol development. In Section 4, we summarize the lessons learned. We conclude the paper in Section 5.

2 PUFs (PHYSICAL UNCLONABLE FUNCTIONS)

2.1 PUFs Overview

PUFs (Physical Unclonable Functions), also known as POWFs (Physical One-Way Functions) [25] or PRFs (Physical Random Functions) [26], are physical one-way functions constructed from the unique nanoscopic-structure of physical objects (e.g., integrated circuits, crystals, magnets, lens, solar cells, or papers) and their reaction to random events. In fact, when a PUF is excited with a random event, called stimulus, the function returns a unique, unpredictable, and reproducible output. This unique output represents the actual fingerprint of a particular PUF and allows its distinction among others. In the field of information technology security (ITS), the stimulus is known as the challenge, whereas the output is known as the response. The set of all possible challenges and their corresponding responses is often referred to as the CRPs (Challenge-Response Pairs).

In the electronics and nanotechnology disciplines, PUFs are constructed from the idiosyncrasies in the manufacturing process of semiconductors [27-29] (e.g., inherent delay characteristics of wires and transistors [30]) to assign uniquely distinct fingerprints to semiconductor pieces [31]. Thus, physical cloning of a piece of semiconductor becomes very difficult. This for example, allows distinguishing a genuine integrated circuit from a...

\(^2\)When a PUF is constructed from a semiconductor, which is generally made of Silicon, the PUF is categorized as a Silicon-PUF (SPUF).
counterfeit one (including exact copies produced during the same manufacturing process [31]). Nowadays, PUFs are considered as a low-cost and a more sophisticated integrated-circuits-tagging technology than hard-printed serial numbers, bar codes, and holograms, which are readily reproducible and consequently not secure.

Formally, we can define a physical unclonable function as a function \( \Psi : C \rightarrow R \), where \( C \) is the set of challenges and \( R \) is the set of the corresponding responses. Also, the function \( \Psi(\cdot) \) should have the following properties:

- **Reliable.** At any time, the same challenge should return the same response, i.e., \( \forall c_i \in C, \Psi(c_i) = \Psi'(c_i) \), where \( t' > t \) (at different instants of time, \( t \) and \( t' \), such that \( t' \) is greater than \( t \), the response is the same for a given challenge \( c_i \in C \)).

- **Unique.** The function \( \Psi(\cdot) \) cannot have an equivalent function, i.e., \( \forall c_i \in C, \Psi(c_i) = \Psi'(c_i) \), then \( \forall \Psi(\cdot) = \Psi'(\cdot) \).

- **One-wayness.** The function \( \Psi(\cdot) \) is a one-way function and it is not invertible, i.e., \( \Psi \circ \Psi^{-1} \neq \text{id}_C \), where \( \text{id}_C \) is the identity function on the set of challenges \( C \).

- **Easy to evaluate.** For any challenge, the function \( \Psi(\cdot) \) is easy to evaluate, i.e., \( \forall c_i \in C, O(\Psi(c_i)) \leq O(1) \) (the temporal complexity for evaluating a PUF is constant).

- **Difficult to replicate.** Function \( \Psi(\cdot) \) cannot be copied or cloned, i.e., \( \forall c_i \in C, \Psi(\cdot) = \Psi'(\cdot) \), then \( O(\Psi'(\cdot)) \approx \infty \) (the temporal complexity of building a clone tends to be infinite).

- **Difficult to predict.** The responses of the function \( \Psi(\cdot) \) are difficult, or impossible, to guess or predict, i.e., \( \forall k \geq 0 = \{(c_0, \Psi(c_0)), \ldots, (c_k, \Psi(c_k)) \}, O(\Phi_{k}(\{(c_{n>k}, \Psi(c_{n>k}))\}) \approx \infty \), where \( \Phi(\cdot) \) is a prediction function based on the set \( \Gamma \geq 0 \) (the temporal complexity of predicting a response, knowing a set of CRPs, tends to be infinite).

When the cardinality of the set of challenge and response pairs (CRPs) is large, the function \( \Psi(\cdot) \) is called a **strong** PUF. Otherwise, the PUF is called a **weak** PUF [32]. Additionally, by increasing the size of the CRPs set, the size of the PUF circuit increases as well. Hence, there exists a proportional relationship between the size of the circuit implementing the PUF and the cardinality of its CRPs set (strength of the PUF). Principally, if the number of CRPs of a given PUF increases exponentially with the size of the PUF circuit, the function is considered strong, whereas if it scales in a linear or polynomial manner, the function is then considered weak [33]. In general, the output of a PUF is subject to noise, which may slightly change the output value for a given challenge. Thus, to generate the correct output for a given input in the presence of noise, a fuzzy extractor is commonly used [34, 35].

For the past twenty years, PUFs have attracted the attention of a large number of researchers. They are recognized as reliable and promising security tools for implementing future security solutions. The attraction is due to the following assets: (1) PUFs have a relatively lower hardware overhead compared to other hardware solutions. (2) PUFs provide higher physical security as they can be used to extract secret keys (volatile-secret keys) from a physical system instead of generating them using software and storing the keys in non-volatile memories (e.g., ROM, fuses, or EEPROM) that need to be secured as well (which is difficult and expensive). This makes them resilient to known physical invasive attacks. (3) They can be used as a hardware random number generators to generate purely random numbers. (4) PUFs do not require special manufacturing processes and treatments (programming and testing steps), which makes them low-cost security solutions. (5) PUFs are practically unclonable. They can be used to design and manufacture low-cost tamper-resistant circuits and devices. (6) PUFs can be used for key agreement between two resource-constrained devices that have never met with each other (no prior key sharing). (7) PUFs are multidisciplinary technologies, where researchers from different fields are contributing to their development.

\[\text{∞ means a very large value that can be considered as impractical to realize.}\]

\[\text{Strong PUFs, such as arbiter-PUF, are generally used for device authentication, whereas weak PUFs are mainly used in key generation.}\]
2.2 Types of PUFs
Since the emergence of PUFs, researchers have worked on designing PUFs by exploiting the physical characteristics of different materials, including Silicon, crystals, magnets, lens, solar cells, and papers. This has resulted in a large variety of PUFs, including but not limited to, delay-based PUFs (e.g., Arbiter-PUFs [26], ring oscillator-PUFs [36], and glitch-PUFs [37]), memory-based PUFs (e.g., SRAM-PUFs [38], DRAM-PUFs [39], SR Latch PUF [40], and Rowhammer-PUFs [41]), acoustical-PUFs [42], coating-PUFs [43], optical-PUFs (e.g., paper-PUFs [44], Compact Disc-PUF [45], and liquid crystal-PUF [46]), and magnetic-PUFs [47]. These PUFs differ from each other in their environmental application, source of randomness (e.g., semiconductors, lens, crystals, magnets, or solar cells), excitation mechanisms (e.g., electronic signals, beam of light or laser, or electromagnetic waves), or other parameters. Interested readers are referred to the work of McGrath et al. [33], where a comprehensive taxonomy of existing PUFs was extensively elaborated.

Despite the existence of various types of PUFs, the Silicon-PUFs (a.k.a., delay-based PUFs) are the most commonly researched and adopted in the electronics and nanotechnology disciplines. This type of PUFs relies on the timing and delay information retrieved from semiconductors (i.e., electronic components made of silicon material). Moreover, similar to delay-based PUFs, memory-based PUFs are becoming more and more popular in the research field as well as in the industrial field. Nowadays, most of the latest secure ICs adopt SRAM-based PUFs for secure storage. For example, the NXP company (the semiconductors division of Philips) has embedded an SRAM-based PUF on its SmartMX2 960 microcontroller and LPC54S0xx family of microcontrollers [48]. Also, Microsemi company has adopted SRAM-PUFs on their SmartFusion2 System-on-Chip FPGA devices [49]. Other hardware from various companies have adopted SRAM-PUFs to implement physical security, such as Coherent Logix HyperX, Intel Altera Stratix 10 FPGA [50], Redpine Signals WyzBee, and the Xilinx Zynq Ultrascale+ [51]. This attraction to SRAM-PUFs is due to the fact that this type of PUF is pervasively embedded within commodity electronics and hence no additional manufacturing is required. Additionally, SRAM-PUFs are considered more resilient to temperature variations and more compact than many other memory-based PUFs.

It is important to note that despite the type of the PUF that is being adopted, the reliability of PUFs, in general, is affected by environmental fluctuations, which include temperature variations, voltage variation, and ambient noise [53–57]. In the case of delay-based PUFs (Silicon-PUFs), the delay of wires and transistors strongly depend on these environmental fluctuations. This, for instance, could result in generating two different responses for a given challenge at two different times or under two different environmental conditions. Therefore, to improve the reliability of PUFs (as well as other properties, e.g., uniqueness), many researchers have proposed different solutions to guarantee a certain level of reliability. For example, fuzzy-extractors, error-correcting codes, and assisting computed helper data are usually applied to generate correct PUF responses.

2.3 PUFs for Authentication
One of the substantially explored security applications of PUFs is the development of on-the-fly and low-cost authentication protocols. These protocols would provide security for resource-constrained devices without having the devices to store any credentials on their limited non-volatile memories. This makes them resilient against physical invasive attacks [58, 59].

---

5Delay-based and memory-based PUFs are often classified as electronic CMOS (Complementary Metal Oxide Semiconductor)-PUFs.
6In some literature, coating-based and optical-based PUFs are often classified as MEMS (Micro-Electro-Mechanical Systems)-based PUFs.
7The reliability of a PUF is a performance metric that measures its reproducibility. A reliable PUF produces the same response for a given challenge at different times and under different environmental conditions [52].
To exploit the advantages of PUFs for integrated circuit authentication in general, and device authentication in particular, a typical challenge-response-based protocol is adopted [26]. The protocol consists of two main phases, the registration phase (a.k.a., enrollment phase) and the verification phase (a.k.a., authentication phase). During the registration phase, a device is enrolled in a database (a trust center) by registering pairs of challenges and responses (CRPs), generated from the PUF that is embedded in the device’s circuit. Therefore, each registered device \( d_{\text{id}} \) will have its proper set of challenge and response pairs in the database \( \Gamma_{\text{id}} = \{(c_0, r_0), \ldots, (c_n, r_n)\} \).

The authentication of a registered device \( d_{\text{id}} \) consists of randomly selecting a pair of registered challenge and response \( (c'_i, r'_i) \in \Gamma_{\text{id}} \) and interrogating the device \( d_{\text{id}} \) by sending the challenge value \( c'_i \) and obtaining the corresponding response \( r''_i \) from the device. Once the response \( r''_i \) is received, it is compared with the registered response \( r_i \) in the database. If both responses are identical (or closely identical), i.e., \( r'_i \approx r''_i \), the device \( d_{\text{id}} \) is authenticated, otherwise, it is rejected.

In general, the device is referred to as the prover \( v \), whereas the authenticator party (which could be the trust center or another device), is referred to as the verifier \( v \). This process is illustrated in the MSC\(^8\) of Fig. 1, where \( \Psi_{\text{p}}(\cdot) \) denotes the prover’s PUF. We note that for higher security, the challenge-response pass during the authentication phase can be run multiple times (i.e., rounds). Moreover, after an authentication round, the used CRP \( (c_i, r_i) \) is sometimes deleted from the verifier’s database to prevent replay attacks [60].

Finally, besides low-cost authentication, PUFs are used in many other applications such as, for the protection of intellectual property (IP) of hardware components (e.g., smart cards [61] and FPGA circuits [62–65]), software components (e.g., software IP binding [66] and images [67]), and embedded devices’ firmware integrity [68]. They are also employed for generating cryptographic keys and random numbers [69–73], and securing memories and processors.

2.4 Attacks on PUFs

In this subsection, we present some attacks on PUFs.

**PUF Invalidation Attack.** This attack assumes that the attacker has physical access to a device’s PUF. The attacker randomly tampers with the PUF circuit in such a way so that the behavior (i.e., output) of the PUF circuit changes and behaves as another PUF. For example, the attacker can spray the PUF circuit with some chemical substances so that the conductivity of the circuit gets affected. Also, the attacker can place some electromagnetic devices near the PUF circuit so that the PUF’s output gets altered. Therefore, any future authentication attempt

\(^8\)MSC (Message Sequence Chart) is a graphical language for the description of the interaction between different components of a system. This language is standardized by the ITU (International Telecommunication Union) [114].
### Table 1

This table summaries the differences between the reviewed PUF-based authentication protocols for T2M and T2T authentication schemes. We have entitled some columns with a numbered letter $P_i$, where $P_1$: Mutual authentication using PUFs, $P_2$: Lightweight (no asymmetric cryptography and no preshared keys), $P_3$: Scalability, $P_4$: Smart, $P_5$: Key establishment (for message authenticity, confidentiality, and integrity), $P_6$: Security evaluation, $P_7$: Performance evaluation, $P_8$: PUF-circuit size evaluation. The symbol ✓, ✗, and • indicate Yes, No, and Possible, respectively.

| Scheme         | Protocol       | Used PUF | Protocol properties | Evaluation | Implementation platform | Possible attacks on the protocol |
|----------------|----------------|----------|---------------------|------------|-------------------------|----------------------------------|
| T2M            |                |          |                     |            |                         |                                  |
| Boyapally et al., [9] (cf., Section 3.1.1) | 5-4 DAPUF | ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | | Digilent Nexys-4 FPGA | Spoofing attacks, Replay attack |
| Qureshi et al., [11] (cf., Section 3.1.2) | Modified APUF | ✓ ✓ ✓ ✓ ✓ ✓ | | 7000 SoC Xilinx ZC-706 | Spoofing attacks, Replay attack, DoS attacks |
| Qureshi et al., [11] (cf., Section 3.1.2) | Modified APUF | ✓ ✓ ✓ ✓ ✓ ✓ | | 7000 SoC Xilinx ZC-706 | Spoofing attacks, Replay attack, DoS attacks |
| Yilmaz et al., [22] (cf., Section 3.1.9) | Unspecified | ✓ ✓ ✓ ✓ ✓ ✓ | | Zolertia RE-Mote | Spoofing attacks, CRPs disclosure, DoS attacks |
| T2T            |                |          |                     |            |                         |                                  |
| Bansal et al., [10] (cf., Section 3.2.1) | Unspecified | ✓ ✓ ✓ ✓ ✓ ✓ ✓ | | STM32F4-MCU | Spoofing attacks, Key disclosure, DoS attacks |
| Barbareschi et al., [19] (cf., Section 3.2.2) | SRAM-PUF | ✓ ✓ ✓ ✓ ✓ ✓ | | STM32F7-MCU Cubietruck-board | Desynchronization, CRPs disclosure, DoS attacks |
| Mahalat et al., [17] (cf., Section 3.2.3) | Unspecified | ✓ ✓ ✓ ✓ ✓ ✓ ✓ | | – | – |
| Feng et al., [21] (cf., Section 3.2.4) | SRAM-PUF | ✓ ✓ ✓ ✓ ✓ ✓ | | MSP-EXP430G2 & MSP430G2553 | Spoofing attacks, CRPs disclosure, DoS attacks |
| Clupek et al., [24] (cf., Section 3.2.5) | Unspecified | ✓ ✓ ✓ ✓ ✓ ✓ | | – | Message forging, CRPs disclosure, DoS attacks |
with the device's PUF will fail as the PUF has changed. To mitigate this attack, we need to physically secure the PUF circuit so that it does not get easily affected by any tampering action or environmental variation.

**PUF Spoofing Attack.** In this attack, a malicious insider has access to all the CRPs of another device. This could happen when the authenticator, generally, the trust center server (a.k.a., verifier), is compromised. In this case, the attacker can spoof the legitimate device and perform successful authentication with other devices that authenticate the spoofed device using the same CRPs. This could also occur in the T2T architecture where devices store CRPs of each other for authentication.

**Eavesdropping Attack.** An attacker intercepts authentication transactions that contain the challenge values and their response values that are computed by the PUF of a target device. The attacker then spoofs the device (prover) and responds correctly to the verifier. The success of this attack relies on the following two assumptions: (1) The number of possible challenge-response pairs is too small. (2) The same challenge is used twice. Such assumptions generally exist on weak PUFs and certain strong PUFs that are predictable.

**Machine-Learning Attack.** This attack is also known as PUF reverse-engineering or PUF modeling attack. In such an attack, an attacker collects a large number of challenge and response pairs (CRPs) used during an authentication or by interrogating a device’s PUF with many challenges to obtain their responses. The collected CRPs are then used to apply machine learning algorithms in order to produce a software model of the PUF that is capable of correctly predicting the responses of new challenges. The efficacy of this type of attack on a given PUF depends on the number of CRPs to be collected from the PUF and the time it requires to build the model given the collected CRPs. This type of attack has been demonstrated in a large number of subsequent publications [74–87].

**Side-Channel Attack.** Side-channel attacks on PUFs have been demonstrated in [82, 88–94]. The goal of these attacks is mostly to gain additional information about the PUF and use that information to improve machine-learning computation times (i.e., training times) from exponential to polynomial. For example, by applying power-based side-channel techniques, an attacker would aim to determine the amount of current drawn from the supply voltage during an output computation (e.g., transition from zero to one of a latch [89]) and hence the power consumption of the function. Additionally, by applying timing-based side-channel techniques, an attacker would aim to learn additional information about the delay of individual response bits. Nevertheless, this type of attack requires complete physical access to the PUF, which is not always possible.

**Replay Attack.** This attack is possible if the PUF-based authentication protocol allows the use of the same challenge twice and the challenges, as well as their responses, are sent in plaintext. An attacker can then intercept some challenge values as well as their corresponding response values to replay them later on. This would allow the attacker to impersonate the device whose challenge and response values were intercepted.

**CRP disclosure Attack.** As many PUF-based authentication protocols are designed by extending the typical authentication protocol illustrated in Fig. 1, these protocols are subject to CRP disclosure. In fact, as the CRPs are stored in plaintext at the verifier, if the latter is compromised, the stored CRPs of the concerned (registered) devices will be revealed. This would allow attackers to impersonate the PUFs of those devices.

In addition to the above mentioned attacks, other attacks on PUFs have been reported in the literature on specific physical unclonable functions. For example, Helfmeier et al., [103] demonstrated a PUF-cloning attack. They showed how they had managed to successfully reproduce the responses of a 2-kilobytes SRAM-based PUF by gaining access to the initial states of the SRAM cells after memory-startup. They applied the PEA (Photonic Emission Analysis) technique [104] to dynamically extract the full content of the SRAM and create a clone of the SRAM. Nevertheless, this attack requires physical access to the PUF circuit, extremely expensive equipments, as well as very long time, which makes it not affordable for most attackers. Nedospasov et al., [105] demonstrated a semi-invasive attack on SRAM-PUFs. They applied a laser probing to read the logic states of an SRAM cells.
Furthermore, some researchers have reviewed PUFs and examined their security issues, highlighted known attacks, and discussed potential countermeasures [113].

3 RECENT PUF-BASED AUTHENTICATION PROTOCOLS

In this subsection, we analyze the security of some recent PUF-based authentication protocols. We identify security flaws and report possible attack scenarios on the reviewed protocols. This would allow future authentication protocol designers to learn from the identified flaws; lessons to consider when designing new PUF-based authentication protocols. Furthermore, we have classified these protocols into two categories based on the communication scheme for which each protocol was designed for. The first category (discussed in Subsection 3.1) gathers protocols that were developed for the T2M communication scheme, where a resource-constrained device (e.g., a sensor) and a resourceful device (e.g., desktop) securely authenticate each other. The second category (discussed in Subsection 3.2), groups protocols that were designed for the T2T communication scheme, where two resource-constrained devices authenticate each other. This classification is based on the corresponding authors’ claim when implementing the protocols.

Throughout this subsection, when analyzing a particular protocol, we use the notation adopted by the original author of the protocol. This would help the reader refer to the original work. Also, for some of the reviewed protocols, we have used MSC (Message Sequence Chart) as a standard graphical tool to illustrate the protocol message flows and security protocol claims [114]. We note that every object or symbol in an MSC has a proper semantic. For example, a condition, denoted by a hexagon, is used to express that the system has entered a certain state (e.g., claimed a security property). Additionally, when using an MSC, we rely on the operational semantics of security protocols to express security properties as claims [115]. Therefore, when a claim is compromised, the related security property is compromised. In the operational semantics of security protocols, a claim is an event where an agent, e.g., an authenticating party, reaches a particular state in the protocol and assumes that a certain security property, e.g., key secrecy, is satisfied. If a particular claim is compromised by an attack, then the concerned agent would assume that the claim is true, where in reality it is not the case. In such a scenario, the protocol is considered compromised and vulnerable to that attack.

Table 1 summarizes the differences between the reviewed PUF-based authentication protocols. The protocols are compared using six characteristics (Column 1, and 3 to 8): (1) Authentication scheme indicates the considered architecture to be either a Thing-to-Thing, in the sense resource-constrained to resource-constrained device, or a Thing-to-Machine, in the sense resource-constrained device to a resourceful device architecture. (2) Used PUF shows the type of PUF being used for the implementation of the authentication protocol referred in Column 2. (3) Properties P1, P2, P3, P4, and P5 mean whether the referred protocol uses PUFs on both authenticating parties, is lightweight from a cryptographic perspective, scalable, smart, i.e., resilient to race-condition attacks [106–109], and allows the establishment of cryptographic keys, respectively. (4) Implementation platform specifies the type of hardware used to implement the PUF and the communicating parties. (5) Evaluation specifies whether the protocols were evaluated in terms of security P6, performance P7, and circuit size P8. (6) Possible attacks indicates the different attacks that can be generated on the referred protocol as we demonstrate in Section 3.

3.1 T2M PUF-based Authentication Protocols

3.1.1 Load Modification-Resistant Smart Meter Authentication. Boyapally et al., [9] proposed a PUF-based authentication protocol for smart meters in the context of smart grid applications. The protocol allows a smart meter to be authenticated to a server using PUFs. The protocol adopts the DAPUF (Double Arbiter-PUF) [111] augmented with a linear feedback shift register (LSFR) module to take a 64-bit challenge and outputs a 128-bit response (which makes it harder to guess). The authors showed that it is resilient to man-in-the-middle attacks as well as...
to PUF-modeling attacks. They implemented the protocol over the Digilent Nexys Artix-7 FPGA-board in a smart meter. The protocol operates symmetric encryption (using AES) along with some asymmetric computations, such as bilinear pairings, to encrypt the response of the smart meters and store it securely on the server. Also, the responses of the smart meter are never exchanged in plaintext, which would prevent modeling attacks. Although the protocol seems to perform an authentication of a smart meter to a server and establish a shared key using Diffie-Hellman key establishment protocol, we have found that the authentication (i.e., the authentication claim) can be easily breached and the server can be spoofed through a replay attack. We present the attack in the next paragraph.

**Attack Scenario.** During the authentication of the smart meter to the server, the smart meter initiates the authentication by sending an authentication request. Based on the source of the request, the server retrieves from the database a cryptographic information, denoted by \( \sigma = (C, \sigma_1, \sigma_2) \). This information contains the challenge \( C \) as well as two values, \( \sigma_1 \) and \( \sigma_2 \). The value of \( \sigma_1 \) was generated during the enrolment phase as \( \sigma_1 = E_{sk_A}(\alpha_2) \), where \( sk_A = H(C) \) is the secret key of the smart meter, \( \alpha_2 = H(sk_{id} | C | A) \). The identity of the smart meter, \( sk_{id} \) is the secret key of the server, \( \Psi(\cdot) \) is the smart meter PUF function, and \( H(\cdot) \) is a hash function. The authentication is illustrated in the MSC of Fig. 2. To generate the attack, upon receiving an authentication request from the meter, the attacker impersonates the server and sends the value \( \sigma \) and \( n = 1 \), where \( \sigma \) is the fixed cryptographic information intercepted from a previous authentication and \( n = 1 \) a nonce. The smart meter performs the computations shown in the MSC of Fig. 3 and sends the value \( \beta_2 = H(\sigma_1, \alpha_2 | n) \) to the attacker. The attacker receives the value of \( \beta_2 \), saves it in a variable \( w \) (i.e., \( w = \beta_2 \)), then drops the authentication. Upon a second authentication attempt from the smart meter, the attacker sends to the smart meter the value \( \sigma \) and \( n \). The smart meter performs the authentication computations and replies by sending the value \( \beta_2 = H(\sigma_1, \alpha_2 | n + 1) \). The attacker sends the value of \( w \) to the smart meter. The smart meter checks the value of \( w \) with the value of \( \beta_1 = H(\sigma_1, \alpha_2 | n) \) and finds that they are equal, which authenticates the attacker as the legitimate server. This attack is illustrated in Fig. 3. Additionally, an attacker can go even further and compromise the key-establishment claim of the protocol. A detailed description of the attack to smash the security of this authentication protocol can be found in [116].

**Lessons**

1. The nonces that are used by both parties are dependent (e.g., Server uses \( n \) and Meter uses \( n + 1 \)). Indeed, there is no randomness introduced by the Meter (i.e., no actual nonce). This consequently results in the non-uniqueness of authentication sessions, and hence the possibility of generating replay attacks.

2. The authentication proofs (i.e., \( \beta_2 \) and \( y_1 \)) used by both authenticating parties are strongly related. This would allow an attacker to easily find a way to construct forged authentication proofs with minimum
Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT

3.1.2 Identity Preserving Protocol for IoT Authentication. Qureshi et al., [11] proposed a PUF-based identity-preserving authentication protocol, called PUF-IPA, for T2M authentication scheme. Similar to the concept used in [12], the protocol uses shuffling techniques to store obfuscated and uncorrelated information about the registered devices’ PUFs in the server instead of storing the challenge and response pairs (CRPs) of registered devices in plaintext. The server can authenticate the registered devices without having the original CRPs stored on its database. The authors showed that the authentication protocol is resilient to machine-learning attacks.

Unfortunately, the protocol is vulnerable to replay attacks. In fact, no verification at the server side is performed to check whether the random number (denoted as \( n_x \)), sent from the device to the server, was previously used or not. Thus, if an attacker intercepts the exchanged values (denoted as \( n_x \) and \( s'_x \)), it will be able to impersonate a server and perform a successful authentication.
the device and respond using the intercepted values. Additionally, the devices are subject to locking attacks (denial of service attacks). In fact, the protocol implements a countermeasure to prevent attackers from conducting a brute force attack. The countermeasure is based on FSM (Finite State Machine) and was first proposed in [98]. It works by locking the device’s PUF circuit after a certain number of authentication failures. Hence, an attacker can paralyze the availability of the system by brute-forcing the registered devices and locking them all.

By analyzing the protocol, we observe that the security of the server relies on the security of a pre-installed cryptographic key stored on the server. If that key is compromised, an attacker can decrypt the stored information and impersonate the server to send dangerous commands to the devices. Finally, the work did not specify what happens when the stored random numbers on the server are all consumed (since there is a finite number of values stored on the server). In the case where the protocol restarts from the beginning, it will be eventually subject to replay attacks.

**Lessons**

3. Authentication protocol designers should make sure that all involved authenticating parties use their own proper nonces during authentication sessions to make the sessions unique. Otherwise, replay attacks become feasible.

4. When adopting a countermeasure, protocol designers should verify whether the adopted countermeasures can be used against the system to cause a denial of service, e.g., not responding after a certain number of authentication failures. In fact, an attacker may have the intention to only make the system unavailable by abusing the countermeasure.

5. A secure system should not rely on a single secret information to protect many other confidential and critical assets (i.e., no single point of failure).

3.1.3 Energy-Efficient Authentication of Medical Devices. An FPGA PUF-based authentication protocol was proposed for medical IoT devices by Yanambaka et al., in [12]. The protocol allows a medical device to be authenticated to the server (one-way authentication). A prototype of the protocol was implemented using two FPGA-boards, a microcontroller (medical IoT device), and a Raspberry Pi (server). The protocol applies the classical challenge-response scheme but in an obfuscated way. In fact, the server first generates a challenge $c$ and computes its response $r$ (i.e., $\Psi_s(c) = r$, where $\Psi_s(\cdot)$ is the server’s PUF) using its embedded PUF. The server sends the response to the device, which uses the response as a challenge to produce a response $r'$ using its PUF (i.e., $\Psi_d(r') = r'$, where $\Psi_d(\cdot)$ is the device’s PUF). The device sends back the response to the server, which uses $r'$ as a challenge for its PUF to produce a final response $r''$ (i.e., $\Psi_s(r') = r''$) and store the tuple or CRP $(c, H(r''))$, where $H(\cdot)$ is a hash function. This protects the CRPs in the case where the server is compromised. In fact, if an attacker gains access to the stored CRPs, it will not obtain the device’s CRPs as the stored response $H(r''')$ is not the response $r'$ of the device’s PUF when the latter (i.e., PUF) is interrogated using the challenge $c$.

However, the authentication protocol contains security vulnerabilities that can be exploited to generate different attacks. The fact that the same challenge and response values can be used during two different sessions without being mixed with any nonces (although they are obfuscated), an attacker will be able to learn the obfuscated CRPs of the device and generate a PUF-modeling attack on the device. The attack can collect the tuple $(r, r')$, which constitutes a valid CRP of the device (since $r$ and $r'$ are sent in plaintext). Also, if the attacker is an insider, it will be able to disclose the CRPs of the server. The attacker just has to keep track of the challenge values that it has sent to the server. Furthermore, the protocol is vulnerable to replay attacks. In fact, the exchanged messages do not contain any timestamp for data freshness and do not contain any information about the authenticity of the source, in particular, the server, as it is not authenticated to the device.
Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT • 13

6. When adopting a challenge-response scheme, the protocol should not exchange the PUF-responses in plaintext. In fact, this would allow an eavesdropper to intercept the challenge values and their corresponding responses and disclose the CRPs. Additionally, by intercepting a large number of CRPs, PUF-modeling attacks become feasible. A straightforward approach would be to hash the PUF-responses along with nonces so that a same CRP could be securely used more than once.

7. The CRPs should never be stored on any trusted location (e.g., trust center) in the form where they can be directly used for impersonation.

8. All authentication messages that are exchanged during an authentication session should be fresh through the use of nonces. Moreover, each party should use a nonce. Otherwise, previously exchanged messages (old messages) will be reused for impersonation.

3.1.4 Machine-Learning Resilient Authentication Protocol. Nozaki et al., [13] proposed a XOR-arbiter PUF-based one-way authentication protocol that is resistant to machine-learning attacks. In this protocol, instead of having the verifier $v$ send one challenge value $c_p$ to the prover $p$ (in plaintext) to receive the corresponding response value $r_p$ (in plaintext), the verifier sends two challenge values $c_p$ and $c_p'$ to the prover $p$, which replies back with one value $w_2$, called distributed value [112]. This distributed value $w_2$ is computed by combining portions of the two response values $r_p$ and $r_p'$ for the challenges $c_p$ and $c_p'$, respectively. That is $w_2 = S_2 \oplus R_0 \ || \ S_1 \oplus R_1$, where $r_p = S_1 \ || \ S_2$ and $r_p' = R_0 \ || \ R_1$. The verifier stores one response (e.g., $r_p$) and one distributed value $w_0$ or $w_1$ (where $w_0 = S_0 \oplus R_0 \ || \ S_2 \oplus R_1$, $w_1 = S_1 \oplus R_0 \ || \ S_0 \oplus R_1$, and $S_0 = 0 \ldots 0$). The stored distributed value $w_0$ (or $w_1$) is used by the verifier to verify the response of the prover. By adopting this obfuscation approach, the prediction rate for a successful machine-learning attack is considerably reduced. Thus, if an attacker intercepts the exchanged distributed value $w_2$, it will not be able to learn the responses $r_p$ and $r_p'$ for the challenges $c_p$ and $c_p'$, respectively. The protocol was implemented on a Xilinx Virtex-5 XC5VLX30 FPGA-board. The protocol is vulnerable to spoofing and CRPs disclosure. In fact, the verifier is capable of constructing the CRPs of the prover (as it stores the first response $r$ and can compute $r'$ using the stored distributed value $w_0$ or $w_1$). Hence, it will be able to impersonate the prover. Also, if the verifier is compromised, the CRPs are disclosed.

3.1.5 CRPs Preserving Authentication Protocol. Chatterjee et al., [14] proposed a PUF-based authentication protocol for IoT. The protocol allows an IoT device (i.e., Thing) to get authenticated to a verifier (e.g., gateway) by using its proper PUF and applying elliptic curve operations (2 multiplications and 3 additions), pairing operations, and a hash function. It adopts a hierarchical architecture, where each verifier authenticates a set of IoT devices and all the verifiers communicate with a central root security association provider, which stores correlated information about the CRPs of IoT devices and their associated verifiers. The protocol was implemented using a
5-4 DAPUF (Double Arbiter-PUF with 5 inputs and a 4-bit output) on the Digilent Nexys-4 FPGA-board (which contains the Xilinx Artix-7-FPGA module) for a video surveillance system.

Notwithstanding, we have found that the protocol has some security issues, where attacks can be generated. For example, to prevent DoS (Denial of Service) attacks, the authors configured the devices (Things) to shut down (or slow down) the protocol execution (e.g., by adopting an exponential back-off algorithm) when a large number of requests are sent to the devices. This countermeasure can be exploited by an attacker whose intention is to shut down the devices for some time. Also, the verifier is configured to reject any further request that regards the same pair for which it has previously received a request. This can be exploited by an attacker since the initial request is not authenticated. The attacker has to spoof a device \( d_i \) and generate requests to devices \( d_j \neq d_i \) before the legitimate device does (cf., race-condition attacks [106–109]). This would make the verifier reject any request coming from the legitimate device \( d_i \) to communicate with another device \( d_j \).

Moreover, the protocol is designed in such a way so that IoT devices (which are assumed to be resource-constrained) perform 2 elliptic multiplications, 3 elliptic additions, one pairing operation, and 7 hash operations, before being able to authenticate the verifier. An attacker can send regular challenge requests to force the device to perform unnecessary operations that drain their power-source (usually a battery).

Furthermore, although the authors have suggested the use of a strong PUFs and to delete authentication information about provers from the verifier database, it is possible for a malicious insider verifier to spoof any prover that is associated to it. In fact, when a prover (denoted as \( A \)) starts an authentication, the verifier (denoted as \( S \)) retrieves authentication information, denoted as \((C_A, C_S, HLP_A, a, B, d_i)\), about the prover. This information can be used to impersonate the prover \( A \).

Finally, the authentication protocol assumes that the CRPs as well as security keys are stored in a secure “offline” database in a trusted environment. Also, it assumes that the secret keys of each prover (denoted as \( K_S \)) are stored in a secure non-volatile memory, i.e., tamper-proof NVM (which is an expensive approach). If these keys are compromised, the security of the system may be compromised as well.

Lessons

11. Setting up a countermeasure that can be used against the system itself is a major vulnerability. Protocol designers should evaluate the inverse consequences of a countermeasure (Similar to Lesson 4).

12. Protocols should be designed in such a way so that devices can decide whether to consider/drop a particular message by just performing fast and costless operations. Otherwise, attackers would flood the system with encrypted bogus messages to be unnecessary decrypted and processed before being dropped.

13. Authentication information, such as CRPs, should not be transmitted or stored in a form where they can be easily generated and reused for impersonation. This is actually a major security vulnerability that most PUF-based authentication protocols suffer from.

14. When designing a protocol, the cost of assumptions has to be estimated as certain assumptions are expensive and are not always feasible.

3.1.6 Service-Centric RFID Authentication Protocol for IoT. A PUF-based authentication protocol was developed for RFID systems in the context of IoT by Liang et al., in [15]. The protocol allows an RFID-tag to be authenticated to a back-end server by proving the possession of the PUF and the authentication of the server to the tag by proving that it knows the PUF model of the RFID-tag. The protocol consists of two phases, the seed generation phase and the verification phase. During the seed generation phase, the server (verifier) and the tag (prover) synchronize on a seed (denoted \( M_i \)) to be used as in input for a PRNG (Pseudo-Random Number Generator) to generate the challenges for the verification phase (s.t., \( c_i = \psi(M_i) \), where \( c_i \) is the challenge and \( \psi \) is the PRNG Digit. Threat. Res. Pract.
function). At each verification session, the seed is used to generate a challenge. The challenge is fed to the PUF to compute the response (i.e., $r_i = \Psi(c_i)$, where $r_i$ is the response and $\Psi$ is the PUF function). The prover divides the response into two parts ($r_i = A \mid B$), xores the two parts, $A$ and $B$, to obtain a value $PW = A \oplus B = C \mid D$, and then randomly pads the results with bits before sending the concatenation of the results, i.e., $(b_0 \ldots b_n \mid C \mid D$, where $b_i$ are random bits). The verifier, which has computed the response $r_i'$ using the PUF model and followed the same steps as the prover (i.e., xored the two parts to obtain $PW' = C' \mid D'$), searches for the presence of the two parts in the received sequence of bits. If both parts are found, the server (verifier) authenticates the RFID-tag (prover). A similar process is conducted to perform the authentication of the server to the tag using another seed, which is the XOR of the two previous parts of the response. The authors proposed a two-stage multiple-choice arbiter PUF (TSMCA-PUF) to design the protocol. The proposed PUF is claimed to be resilient to modeling attacks. It used the BAN (Burrows-Abadi-Needham) logic to prove the correctness of the protocol and performed an informal security analysis of the protocol with respect to modeling attacks, man-in-the-middle attack, replay attack, CRPs exposure, and spoofing attacks. A performance analysis was also performed. The protocol was implemented using the Virtex II-PRO FPGA-board and Xilinx XUPV5-LX110T development boards. Nevertheless, our analysis shows that it is possible to generate a desynchronization attack through message modification.

**Attack Scenario.** During the seed generation phase, the server (verifier $v$) generates a nonce $T_v$, constructs a message $M_1 = \{\mathcal{H}(T_v) \oplus T_v\}$, and sends it to the RFID-tag (prover $p$). The tag generates a nonce $T_p$, constructs a message $M_2 = \{M_1 \oplus T_p\}$, and sends $\{M_2 \mid T_p\}$ to the server (where $|$ is the concatenation). The server computes $T_v' = M_2 \oplus T_p \oplus \mathcal{H}(T_v)$ and checks if $T_v = T_v'$. If $T_v = T_v'$, the server sets up the seed to be $s = M_2$. An attacker $e$ could have changed the tag message to $\{M_2 \oplus T_p \oplus T_e \mid T_p \oplus T_e\}$, which would have made the server set the seed to $M_2 \oplus T_p$ instead of $M_2$. In this situation, both the tag and the server will be synchronized over a different seed, which would result in an authentication failure. The system can only be recovered by resetting (re-synchronizing) the tag and the server to use the same seed. Furthermore, the server keeps a PUF model of the registered tags on its database. Compromising the server (e.g., through a malicious insider) would result in the disclosure of the PUF models of the tags and hence their CRPs. Additionally, the server can spoof any registered tag as it knows how to generate all the CRPs of the tag.

**Lessons**

15. When an authentication protocol is based on synchronization, protocol designers have to make sure that information related to the synchronization has to have its integrity protected. Otherwise, any unauthorized modification would create desynchronization and hence a malfunction of the protocol.

16. Establishing authentication parameters using unauthenticated messages is a serious security issue when there is no way for the involved parties to verify and confirm that the parameters have not been modified.

17. The concept of distributed value, used in [13], can be used as a good obfuscation approach to solve the issue of keeping the CRPs of registered devices secure (Similar to Lessons 7, 10, and 13).

**3.1.7 Single CRP-based Authentication Protocol for IoT.** Kim et al., [16] proposed a PUF-based authentication protocol to allow a smart device to be authenticated by a server. The protocol tries to reduce the load issue of storing all the CRPs of all registered devices on a particular server (i.e., trust center) by storing only one CRP, $(c_{d_i}, r_{d_i})$, per device $d_i$, at a time. Therefore, at the registration phase, only one CRP per device is saved into the database. The stored CRP for a given device is updated after each authentication. That is, when a device $d_i$ is authenticated, it generates a new CRP, $(c_{d_i}, r_{d_i})$, and sends it to the server. A prototype of the protocol was
implemented on an STM32F4 MCU-board. The solution does not actually eliminate the storage-issue of storing all CRPs on a server. It only reduces the amount of space occupied by the storage of the CRPs. In fact, after deleting a used CRP, the server keeps a list of challenge values, which grows proportionally with the number of authentication attempts. Furthermore, we have found that the protocol is vulnerable to certain attacks:

**Attack Scenario 1.** In this protocol, the device starts by requesting an authentication from the server by sending its identity. The server replies back by sending a challenge $c_i$. The device responds to the server by computing the corresponding response $r_i$ and generating a symmetric key $K_0 = c_i + r_i$ to send an encrypted response message to the server. The response message contains a new CRP, $(c_j, r_j)$. The server computes the key, decrypts the message, and then verifies the value of the response. If the response is correct, the device is authenticated, the new CRP, $(c_j, r_j)$, is saved, and an acknowledgement is sent to the device. Thus, as the authentication of the device $d_i$ is based on successfully responding to one challenge $c_i$, an attacker can brute force the response $r_i$ by generating all possible values for $r_i$ until getting an acknowledgement from the server. This would eventually depend on the size of the response, which is in general smaller than the challenge. That is, if the responses are on $k$-bits, the attacker just has to try at most $2^k$ authentication attempts. Once the value of $r_j$ is disclosed, the secret key $K_0$ is cracked. Interestingly, if the attacker forges its authentication response messages in such a way so that the new CRP value is nonsense, e.g., $(c_j, r_j) = (0, 0)$, the server would then save the incorrect CRP. This would result in all future authentication attempts from the legitimate smart device to fail until a CRP-updating procedure is initiated to update the stored CRP with a new pair.

**Attack Scenario 2.** The server and the smart device are both subject to DoS (Denial of Service) attacks. With respect to the server, as authentication messages, sent from the device to the server, are completely encrypted, the server has to first decrypt the messages before authenticating the source (i.e., device). Thus, if an attacker floods the server with bogus messages, the server will perform decryption operations before dropping the messages. With respect to the device, as the server is not authenticated to the device, an attacker can flood the device with spoofed new CRP requests and force it to consume its battery to unnecessarily generate new CRPs (since here the server is the initiator of the CRP-updating procedure).

**Lessons**

18. Protocols should use variables (e.g., PUF-responses) of larger sizes to harden the feasibility of brute force attacks. Hash functions could be used to complement and enlarge the size of the PUF-responses by keeping the size of the PUF-circuits smaller.

19. Many IoT applications are built using resource-constrained devices that operate on rechargeable batteries. If authentication protocols are designed in a way where devices can be abused by performing unnecessary heavyweight computations, then the power-source of these devices will be drained and their availability will be breached (Similar to Lesson 12).

**3.1.8 Smart home Authentication Protocol for IoT.** Mughal et al., [18] proposed PAS (PUF-based Authentication Scheme), an authentication protocol for smart devices in IoT smart home applications. It allows users, equipped with smartphones, to be authenticated to a gateway so that they can send command-messages to connected smart devices (e.g., a light bulb or thermostat). However, the gateway is not authenticated to the users (no mutual authentication). Additionally, the protocol assumes that smart devices and the gateway share a pre-established symmetric key to encrypt the communication (i.e., the command-messages sent by the user). If the key is compromised (e.g., through device physical invasion), an attacker will be able to forge new command-message (e.g., turn camera Off) as well as modify command-messages sent to smart devices by the legitimate user (e.g., increase temperature instead of decreasing it). In fact, the command-messages are not authenticated. The protocol
Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT

Lessons

20. Relying on a single point of security is a major flaw. We urge that protocol designers should perform a dedicated analysis where the dependence and relationship between systems’ assets is evaluated. For example, taking over an asset, e.g., a secret key, should not allow attackers to transitively take over the remaining assets (Similar to Lesson 5).

21. Message authentication code, in particular HMAC\(^a\), should be the last cryptographic operation to be performed after the encryption of a message and not the other way around. In fact, if devices have to decrypt a message to be able to verify the authenticity of the source of the message (i.e., verify the HMAC), then attackers may take advantage of this design flaw to overwhelm resource-constrained devices with nonsense messages forcing them to decrypt the messages and draining their batteries, which would make those devices become unavailable (Similar to Lessons 12 and 19).

\(^a\)HMAC (Keyed-Hash Message Authentication Code) provides a message integrity code that is computed over a message and using a shared authentication key. This allows the authentication of the source as well as the protection of the integrity of the message.

3.1.9 Neural Network-based Authentication Protocol for IoT. An arbiter PUF-based authentication protocol was proposed for the T2M authentication scheme by Yilmaz et al., in [22]. Based on the use of a neural network model of the PUF, the protocol allows a verifier \(v\) to authenticate a prover \(p\) without having to store the challenge and response pairs (CRPs) on the verifier’s memory (database). The protocol makes use of a device MAC address and a timestamp to obfuscate the PUF response by applying the RC5 cipher. They showed, using Zolertia RE-Motes, that the protocol performs better than DTLS (Datagram Transport-Layer Security) and UDP (User Datagram Protocol), in terms of memory and energy consumption. Nevertheless, the protocol relies on the assumption that creating a neural network model for a PUF is evident, which is not always the case. The authors have built a PUF-model for arbiter PUFs which are known to have the simplest structure. We have found that the protocol has some security vulnerabilities that can be exploited to generate various attacks.

The authentication proof that is generated by the prover, denoted as \(\alpha_d\), and the authentication proof that is generated by the verifier, denoted as \(\alpha_v\), are exactly the same, i.e., \(\alpha_d = \alpha_v = \text{RC5(MAC, } R_i \oplus T_d)\); where MAC is the 48-bit MAC-address of the device, \(R_i\) is the device PUF’s response to a challenge \(C_i\) and, \(T_d\) is the device’s timestamp. This technically means that there is no authentication. In fact, the verifier can just resend the authenticity proof that it has received from the prover to prove its authenticity to the prover. An attacker would eventually do the same to bypass the authentication. In addition, these authentication proofs are computed using RC5 symmetric cipher along with a pre-shared symmetric key. The size of the key has not been reported, which raises the question of RC5 cracking possibility, and hence PUF-responses and CRP disclosure attacks.

The protocol is vulnerable to insider attacks. A verifier knows the complete PUF model of the prover, it can generate all the CRPs of the prover. Hence, the verifier can easily impersonate the prover. Also, if the verifier is compromised, the PUF model of all registered provers will be disclosed and the security of the system is paralyzed.
22. The authentication proofs of the verifier and the prover to prove their identities to each other are identical $a_d = a_o$. This actually allows any device to impersonate the verifier and bypass the mutual authentication of the protocol. Authenticating parties should not be able to take advantage over one another to produce correct proofs. A straightforward approach would be to require each authenticating party to use a proper nonce, so that the proofs become unique per authentication session and per party (Similar to Lesson 2).

23. Since the verifier (server) stores neural network models of PUFs of all registered provers (devices), the protocol is vulnerable to insider attacks, as well as, to CRP disclosure through physical invasion. Whatever the level of trust that is assigned to the verifier, the latter should not be able to reconstruct the CRPs of any other device (Similar to Lesson 10).

3.1.10 Secret-Message Exchange Authentication Protocol. Idriss et al., [23] proposed a PUF-based authentication protocol with key exchange for resource-constrained systems, such as RFID systems. The protocol relies on a challenge-challenge scheme, where a verifier $v$ sends $m$ challenges (of size $n$), $\{c_1, \ldots, c_m\}$, to the prover which replies back with $2 \times m$ challenges, $\{(c_{1,1}, c_{1,2}), \ldots, (c_{m,1}, c_{m,2})\}$, such that for every challenge $c_i$ sent by the verifier the following equation should hold:

$$\Psi_p(c_i) = \Psi_p(c_{i,1}) \oplus \Psi_p(c_{i,2})$$

In the same way, the prover authenticates the verifier by sending $m$ challenges and obtaining $2 \times m$ challenges for verification. Basically, the authentication of the prover to the verifier relies on the prover proving that it has the authentic PUF, whereas the authentication of the verifier to the prover relies on the verifier proving that it has the model of the prover’s PUF to compute its CRPs. The authors analyzed the security of the authentication protocol against guessing and challenge collection attacks.

Nevertheless, based on the authors’ assumption of using the protocol for RFID systems (resource-constrained systems), this protocol has the following issues: (1) It requires from the device (which models the RFID-tag in [23]) to send $3 \times m \times n$ bits (transmission) and receive $2 \times m \times n$ bits (reception) to complete an authentication, which seems to be large amount of bits to be transferred on an RFID system. (2) Given a challenge value $c_i$, the prover as well as the verifier, have to go through a loop to generate the values $c_{i,1}$ and $c_{i,2}$ such that the condition $\Psi_p(c_{i,1}) = \Psi_p(c_{i,1}) \oplus \Psi_p(c_{i,2})$ holds. The probability to generate the values of $c_{i,1}$ and $c_{i,2}$ so that the previous equation is satisfied is $(1/2)$ for one challenge $c_i$ and $(1/2)^{-m}$ for all the challenges $c_i, i \in \{1, \ldots, m\}$. This means that there is 50% chance to fall into an infinite loop when looking for $c_{i,1}$ and $c_{i,2}$ for a given challenge $c_i$, which would make the system nonresponsive. In fact, when the server receives the challenge $(c_1, \ldots, c_m)$, it uses the PUF model ($\Psi_p(\cdot)$) of the device to compute the response $R' = (\Psi(c_1), \ldots, \Psi(c_m)) = (r_1, \ldots, r_m)$. Then, it takes each individual response $r_i$ and uses a True Random Generator (TRNG) to generate two independent values $c_{i,1}'$ and $c_{i,2}'$ such that condition $\Psi_p(c_{i,1}') \oplus \Psi_p(c_{i,2}') = r_i$ holds. For each response $r_i$, the server enters a loop and repeats the generation of the random numbers $c_{i,1}'$ and $c_{i,2}'$ till the condition holds. For each attempt, if at least one single bit in $c_{i,1}'$ or $c_{i,2}'$ is different from bit sequence of the real value of $c_{i,1}$ and $c_{i,2}$, the value of $r_i$ will be different and the condition will not hold. Thus, there is 50% of chance for the server to fail an attempt and hence there is 50% of chance that the server may go through an infinite loop. (3) The PUF model of the prover is stored at the verifier. This would allow the verifier to be able to spoof the prover. For example, if the verifier is a malicious insider, it will be able to impersonate the prover (tag). Additionally, since the PUF model is stored on the verifier as is, compromising the latter would disclose the PUF model and hence the CRPs of the prover, i.e., the tag.
24. A performance analysis has to be performed to demonstrate that the protocol conforms the performance requirements of the application for which it was developed for. In fact, many applications use low-cost and resource-constrained devices that cannot implement and operate certain heavyweight computations (e.g., using ECC on passive RFID-tags).

25. A formal verification of the protocol, using well-established tools should be performed to determine whether the protocol terminates, e.g., Issue 2 might end up in an infinite execution.

26. This authentication protocol allows a verifier to store PUF-models about other devices’ PUFs. This assumes that all verifiers are trustworthy, which is not always true. Authentication protocols should be designed in a way where insiders are considered part of the attacker’s model (Similar to Lesson 10).

3.2 T2T PUF-based Authentication Protocols

3.2.1 Vehicle-2-Grid Authentication Protocol. Bansal et al., [10] proposed a PUF-based authentication protocol with key establishment, called SUKA, for vehicles smartgrid infrastructures. The protocol uses PUFs to allow the authentication of electric vehicles (EVs) to aggregators (a.k.a., mediator) as well as to the grid-server (GS). This protocol can be perceived as a T2T authentication protocol as the electric vehicles as well as the aggregator are considered resource-constrained devices that embed a proper PUF circuit. The authors performed a formal security analysis using Mao-Boyd Logic to prove the security of their protocol. Nevertheless, we have found that the security of the protocol can be breached and attacks such as spooing and message forging can be generated. In fact, we discovered that it is possible to crack the derived session key between an aggregator and server (denoted $S_k$ as in [10]) and use the key to disclose all the CRPs (denoted as $(C'', K'')$ in [10]) of an electric vehicle. The attack can be repeated on each vehicle to disclose its CRPs.

**Attack Scenario.** To understand the attack, Fig. 4 illustrates the authentication part between the grid-server and the aggregator. The attacker starts by intercepting all the exchanged messages during the authentication. It uses the second message sent from the server to the aggregator to extract the values of $C$ and $V$. From the value $C$ the attacker can learn how many challenges are being used, i.e., the value of $m$. Then, using the value of $m$, the attacker computes $K_0$ (the aggregator’s PUF response for the first challenge $C_0$) using the value $N$ (i.e., $K_0 = m \oplus K_0 \oplus m$). Next, the attacker intercepts the third message and extracts the value $N'$ to compute the nonce $N_C$ using the previously learned value of $K_0$ (i.e., $N_C = N' \oplus K_0$). At this stage, to compute the secret key, $S_K = F(K_0, N_B) \oplus F(K_0, N_C)$, the attacker just needs to find the value of the nonce $N_B$. To that end, the attacker can run an offline brute force attack on the used message authentication code (MAC) function on the second message. The attacker holds the code value, i.e., MAC$(ID_m | M | m | N_B)$, as well as three out of four input parameters for the function. If the attacker cracks the nonce $N_B$, it will compute the secret key $S_K$ and decrypt all the vehicle’s CRPs that will be used next time (denoted as $(C^\#, K^\#)$ in [10]). In this case, the attacker will have to spoof the electric vehicle before the vehicle performs a new authentication with the system, otherwise, the CRPs will be updated at the server-side in the next authentication session.

Furthermore, the electric vehicle authenticates an aggregator by having the vehicle perform computations and infer that the aggregator knows the vehicle CRPs, which assumes that the aggregator already performed an authentication to the server and obtained the vehicle CRPs. In this case, there are two issues. The first issue is that if an attacker performs the attack described above, the attacker will be able to impersonate an aggregator and

---

9The reader might think that cracking a hash function could take a long time. However, these will depend on the size of the nonce $N_B$ and the type of the MAC-function. The nonce could be cracked in the same way as a Wi-Fi WPA-key is cracked after capturing the WPA 4-way-handshake packets.
Fig. 4. The authentication protocol by Bansal et al., [10] that allows an aggregator to be authenticated to a grid server and establish a shared secret session key $S_k$.

scam the vehicle. The second issue is that the grid server is considered a trusted party that stores “in plaintext” the CRPs of all registered electric vehicles as well as their identities. Thus, if the server is compromised (e.g., through a malicious insider), all the CRPs are disclosed and the security of the system is completely breached. Finally, it was claimed that the protocol preserves the identity of the vehicle owners to protect them from being tracked. However, the identity of the vehicle owners is sent in plaintext when initiating an authentication with an aggregator. This would allow an attacker to keep track of the owners, which invades its privacy.
Lessons

27. The XOR operator should not be applied to obfuscate secret information using easily derivable parameters. The XOR is a commutative and absorbent operator. In fact, \( \forall x, y, z \mid z = x \oplus y \), then \( z \oplus y = y \oplus z = x \). Thus, if we assume that \( x \) is a secret, there should be no way for the attacker to learn the value of \( y \) in the case where the attacker intercepts \( z \).

28. Use automatic security protocol provers, e.g., TAMARIN, to check whether there is any vulnerability related to how the protocol applies the XOR operator.

29. The CRPs should never be stored in plaintext in any location, whatever is the trust of that location. This flaw is usual as many security protocol designers do not consider the case of insider attacks.

3.2.2 Synchronous Authentication Protocol. Barbareschi et al., [19] proposed PHEMAP (PUF-based Mutual Authentication Protocol), an SRAM PUF-based protocol designed for terminal to terminal and terminal to gateway. It is based on the notion of chains, which are sequence of values generated by recursively invoking a PUF on a device. The chain is used to authenticate a device by other authenticating parties, which have to store parts of that chain, called links. The protocol strongly relies on synchronization. For example, at a given time \( t \), if a device \( v \) (verifier) wants to authenticate another device \( p \) (prover), then the verifier \( v \) has to be synchronized with the prover \( p \). This synchronization consists of having the verifier \( v \) store a specific link \( \ell^p_t \) about the prover \( p \) and having the prover \( p \) be aware that the verifier \( v \) is currently storing the link \( \ell^p_t \). In 2019, the protocol was extended and adopted for cloud-edges IoT systems [20]. The protocol was implemented using the STMicroelectronics STM32F7-board (for terminals) and the Cubieboard Cubietruck-board (for gateways). We have found that the protocol is vulnerable to desynchronization attacks. In the following, we discuss two attack scenarios:

**Attack Scenario 1.** During the verification phase of the authentication, an attacker may spoof a prover \( p \) (e.g., terminal) and send an incorrect response to the verifier \( v \) (e.g., gateway) before the legitimate prover does. Upon sending the correct response by the prover \( p \), the latter updates its link (e.g., \( \ell^p_t \)) to the next value on the chain (e.g., \( \ell^p_{t+1} \)). Then, when the verifier \( v \) receives the incorrect response, it drops the message and unauthenticates the prover \( p \) even if it receives the correct response later on. At that time, the prover \( p \) and the verifier \( v \) are synchronized over a different value of the link (\( p \) stores the link \( \ell^p_{t+1} \), whereas \( v \) stores the link \( \ell^p_t \)). Thus, any future authentication attempt would fail. This attack is illustrated in the MSC of Fig. 5. Also, we note that this attack scenario can be generated during the initialization phase of the protocol by spoofing the second message.
We can observe that the strength of the protocol resides in keeping the value of the nonce when hitting the correct value of $\ell^p_t$ (i.e., $\ell^p_{t+1}$). When the correct value is hit (viz., the first message in the MSC of Fig. 5), the prover $p$ authenticates the verifier $v$ and updates the current link to $\ell^p_{t+2}$, which desynchronizes the prover and the verifier. Although this second scenario seems to be time-consuming, its impact is worse than the one of the first scenario. In fact, after this attack scenario, the prover $p$ will be storing the link $\ell^p_{t+2}$, whereas the verifier $v$ will be storing the link $\ell^p_t$. The attacker can repeat this attack scenario multiple times in such a way so that the current chain gets invalidated.

**Lessons**

30. Synchronization information should not be updated before making sure that authentication is established. For example, in this protocol, a device updates the synchronization information “link” before confirming that it is communicating with the right counterpart, which caused the desynchronization of both devices, and hence a denial of service attack.

31. The PUF challenges and responses should be large enough to harden the feasibility of brute force attacks (Similar to Lesson 18). Also, as it is more common to adopt a countermeasure that increases the complexity of a brute force attack, e.g., by limiting the number of requests, protocol designers should be careful when adapting a countermeasure that can be used to abuse the system’s availability.

### 3.2.3 Secure Wi-Fi Authentication Protocol

A PUF-based authentication protocol was proposed for secure Wi-Fi authentication of IoT devices by Mahalat et al., in [17]. The protocol is adapted for a T2M authentication scheme, where a Wi-Fi IoT device gets authenticated to a Wi-Fi router. This protocol aims to add to the existing security mechanisms, e.g., WPA2 (Wireless Protected Access 2), a physical security to mitigate existing MAC-layer attacks.

For each registered Wi-Fi device, the protocol stores three CRPs on the Wi-Fi router. The router uses all three CRPs to authenticate the device (the device infers that it is communicating with the legitimate router as the router knows the device’s CRPs). Additionally, the CRPs are updated after each authentication to prevent replay attacks (in case the same nonce is used twice or the nonce is brute-forced). Similar to other protocols, this protocol stores the CRPs of registered Wi-Fi devices into the Wi-Fi router database in a plaintext format. Thus, if the access point is compromised, the CRPs of the registered Wi-Fi devices will be disclosed and the affected devices can be spoofed. Furthermore, the security of the protocol can be entirely compromised by observing two authentication sessions from the same Wi-Fi device as explained in the next paragraph.

**Attack Scenario 2.** If we assume that the links are encoded on $n$-bits, then at a given time $t$, the attacker may spoof the verifier $v$ and try all possible values for the current link $\ell^p_t$ (i.e., $2^n$ combinations). The attacker stops the attack when hitting the correct value of $\ell^p_t$ (i.e., $\ell^p_{t+1}$). When the correct value is hit (viz., the first message in the MSC of Fig. 5), the prover $p$ authenticates the verifier $v$ and updates the current link to $\ell^p_{t+2}$, which desynchronizes the prover and the verifier. Although this second scenario seems to be time-consuming, its impact is worse than the one of the first scenario. In fact, after this attack scenario, the prover $p$ will be storing the link $\ell^p_{t+2}$, whereas the verifier $v$ will be storing the link $\ell^p_t$. The attacker can repeat this attack scenario multiple times in such a way so that the current chain gets invalidated.
between the device and the router. Since the parameter $C_3$ is sent in plaintext during the authentication and the next new value for $C_3$ is $C_3 \oplus nonce$, the attacker will be able to compute the value $nonce = C_3 \oplus nonce \oplus C_3$. Using the value $nonce$ and the intercepted values $C_3'$, $C_2'$, $R_2'$, and $R_3'$, the attacker infers the value of $C_1$, $C_2$, $R_2$, and $R_1$. This would allow the attacker to infer the new value of the nonce and compute the next CRPs of the device to mount other attacks as illustrated in the last part of the MSC of Fig. 7.

32. The verifier (access point) stores the CRPs of the registered Wi-Fi stations in plaintext. This is fundamentally insecure as it allows attackers to disclose the CRPs using different types of attacks, as well as, to perform spoofing attacks in the case where the access point is hijacked. A secure PUF-based authentication protocol should store the CRPs in a way so that it cannot use them directly. For example, the concept of distributed value, as adopted in [13], can be used as a good obfuscation approach to solve this issue.

33. In this protocol, the authentication responses used in a former session are dependent on the authentication responses that are used in a later authentication session. This creates the possibility of generating...
Fig. 7. Attack on PUF-based mutual authentication protocol [17]. For clarification, the blue values are captured by the attacker during a previous authentication session (viz., Fig. 6), whereas the purple values represent the values that the prover computes in the current authentication session. The black values are the stored and transmitted ones. As can be seen at the end of the attack, the attacker has the CRPs of the prover that will be used in the next authentication session. The attacker can spoof both the router and the device, that can cause denial of service and generate more sophisticated attacks.

Digit. Threat. Res. Pract.
replay attacks. Therefore, authentication protocol designers should ensure that authentication responses from one session to another differ significantly and cannot be reused (Similar to Lessons 1 and 2).

34. Incorrect usage of the XOR operator can affect the security of the entire protocol. Protocol verifiers such as AVISPA, Tamarin, and ProVerif, can verify the correctness of the usage of the XOR operator (Similar to Lessons 27 and 28).

3.2.4 Lightweight Attestation and Authentication Protocol. Feng et al., [21] proposed a PUF-based lightweight attestation and authentication protocol, called AAoT, for IoT (Internet of Things) and CPS (Cyber-Physical Systems). The protocol uses PUFs along with fuzzy extractors to derive a secret key that is used to establish mutual authentication between devices. SRAM-PUFs were adopted to implement the protocol on MSP-EXP430G2 development boards. The protocol is vulnerable to spoofing attacks, CRP disclosure, and replay attacks. In fact, the protocol stores the CRPs of the prover on the verifier database. This would allow the verifier (in case the verifier is a malicious insider) to impersonate the provers whose CRPs are stored on the verifier's database. If the verifier is compromised (e.g., through invasion attack), the provers' CRPs are disclosed.

The prover relies on a 16-bit random number generator (RNG) along with a response of a challenge $c$ (sent by the verifier) to operate a fuzzy extractor module and produce a helper data $h$ as well as a secret key $K$. The helper value $h$ is sent to the verifier which uses it along with the response $r$ (as it has the prover's CRPs) to compute the key $K$. An attacker can intercept an authentication session $S_j$ and use the intercepted parameter (e.g., $c_j$, $h_j$, and $M_j$) to spoof the verifier in another session $S_k$. To that end, the attacker employs brute-forcing and challenges the prover with the same challenge $c_j$ (intercepted during session $j$) until the prover responds with $h_j$. In that case, the attacker will be able to correctly respond using $M_j$ (intercepted during session $j$) and get authenticated. The attacker will just have to challenge the prover at most $2^{16} = 65536$ times with the same challenge $c_j$. This attack is possible as long as the prover does not implement any mechanism to limit the number of challenges with the same challenge in short time duration.

35. CRPs stored in plaintext make the protocol vulnerable to CRP disclosure and PUF impersonation. Obfuscation techniques, such as distributed value (adopted in [13]), can be used to secure the CRPs of other devices when they are stored or transmitted (Similar to Lessons 7, 10, and 29).

36. PUF responses are expressed in 16 bits, which is relatively small to thwart brute force attacks. In fact, the attacker will have to try at most 65536 values before revealing the correct PUF response. This would take few seconds on a modern computer. Therefore, protocol designers should consider at least 64 bits, to increase the temporal complexity for performing brute force attacks (Similar to Lesson 18).

3.2.5 Third Trusty Party-based Authentication Protocol. Clupek et al., [24] proposed a PUF-based mutual lightweight authentication protocol for T2T schemes. It uses a third trusty party (TTP) as a trusted authority to establish a robust authentication between Things. It relies on the use of low-cost PUFs, simple hash functions, and the binary eXclusive-OR operator. We have found that this protocol is vulnerable to certain attacks as follows:

**Attack Scenario 1.** During the authentication, the authenticating parties exchange their messages (5 messages) over an insecure channel in plaintext. An attacker can disclose the secret information $s$ and breach the confidentiality of the system. In fact, if the attacker intercepts the second messages of the protocol, it captures the response $r_p$.
that was sent by the prover \( p \) to the verifier \( v \). Then, by intercepting the fourth message, which contains the value \( s \oplus H(r_p) \), the attacker would be able to compute the secret information \( s \) by evaluating \( s \oplus H(r_p) \oplus H(r_p) \), where \( H(\cdot) \) is a public hash function. In this case, the secret information \( s \) has no longer become secret to the attacker (i.e., no confidentiality).

**Attack Scenario 2.** During the authentication, the challenge values and their corresponding response values are exchanged in plaintext. This would allow an attacker to intercept these values and perform a machine-learning attack [76–78]. In this attack, the attacker learns different challenge and response values (i.e., CRPs) and build a perception model of a given PUF. Using the constructed perception model, the attacker can impersonate the concerned PUF of a target device. This would compromise the authenticity of the protocol.

**Attack Scenario 3.** As an attacker can read the content of all messages exchanged during an authentication, the attacker can eventually tamper-with the secret information \( s \). In fact, by intercepting the challenge \( c_p \) (from message 1) and its corresponding response \( r_p \) (from message 2), the attacker would be able to forge the fourth message that contains the secret information \( s \) and include another information \( s' \). The attacker just has to make sure that its message is received the first before the legitimate verifier sends it secret information. This attack is possible as there is no way to authenticate the source of a given message. In addition, the hash function used for message integrity does not use any secret. It can be computed by any party, including the attacker. This attack scenario is illustrated in the MSC of Fig. 8.
37. The XOR operator can be used to break the security of the protocol and reveal the secret parameter due to its incorrect usage. This bitwise operator provides properties that allow the implementation of secure and lightweight cryptographic operation. However, it can destroy the security of the protocol if it is used incorrectly (Similar to Lessons 27, 28, and 34).

38. Transmitting the CRPs in plaintext allows eavesdroppers to construct a model of the concerned PUF function by just intercepting a certain number of CRPs, depending on the type of PUF that is being adopted. The concept of distributed value, adopted in protocol Nozaki et al., [13], is an interesting concept for obfuscating the PUF responses when being transmitted. Also, sending the hash of the PUF response along with nonce would be a good approach as well (Similar to Lesson 13).

**4 DISCUSSION**

In this section, we further discuss the lessons that we have drawn from analyzing the security of recent PUF-based authentication protocols in the previous section. We summarize the major lessons learned from the most common security design flaws as follows:

**CRP Disclosure Flaw.** We have observed that many PUF-based authentication protocols suffer from CRPs disclosure attacks and PUF impersonation (e.g., attacks on protocols proposed in [10], [12–15], and [21–23]). This issue is due to the incorrect design of the part of the protocol that is responsible for keeping the CRPs of registered devices secure when these CRPs are transmitted or stored. This actually could be an interesting research direction to design PUF-based authentication protocols with a focus on securing the CRPs of registered devices from attackers. This would make the protocols resilient to CRP disclosure and PUF’s impersonation through malicious insider attacks. Also, we urge that authentication protocol designers should use security protocol verifiers, such as Isabelle/HOL, Tamarin, and ProVerif, to prove security properties in their protocols. For example, one can verify the secrecy of the CRPs to check whether it is possible for an attacker to reveal the CRPs of other devices.

**XOR Operator Misuse Flaw.** We have found that various protocols adopt the logical bitwise exclusive OR operator (i.e., XOR, denoted by ⊕) as a secure operator to perform lightweight computations. However, if this operator is used in an incorrect way, e.g., with easily deducible parameters to protect a credential, then this operator becomes the key for revealing other related credentials in a transitive way. In fact, due to the absorption property of XOR, publicly-known parameters can easily be eliminated from a logical expression that was computed using XOR, disclosing the values of other parameters, which could be secret parameters (e.g., attacks on protocols proposed by Bansal et al., [10], Mahalat et al., [17], and Clupek et al., [24]). Therefore, authentication protocol designers should be really careful when using this powerful logical operator. Automatic security protocol verifiers can be used to identify security flaws resulting from the incorrect usage of the operator.

**Authentication Session Uniqueness Flaw.** We have identified a common protocol-design flaw when it comes to the use of nonces. We have found that in some authentication protocols, the nonces are not used per authenticating party to make an authentication session unique w.r.t. the involved variables. This opens the possibility of generating replay attacks. Additionally, in some protocols, the nonces from different authenticating parties are related (dependent), which makes it possible for an attacker (spoofing a prover) to generate authentication proofs by exploiting an authenticating party, usually the verifier, and using the obtained proofs to bypass authentication (e.g., attack on the protocol by Boyapally et al., [9]).

**Countermeasure Inverse Consequences Flaw.** Some authentication protocols (e.g., Chatterjee et al., [14] and Qureshi et al., [11]) implement a security measure (countermeasure) to mitigate or slowdown certain unusual and
“suspicious” activities when the latter (activities) are detected. For example, when a large amount of authentication requests (or messages, in general) coming from the same source is detected, the system starts ignoring any new message coming from that suspicious source. Notwithstanding, an attacker could take advantage of this countermeasure to attack the availability of the system by spoofing legitimate devices and flooding the system with bogus messages. This would make the system blacklist those legitimate devices and ignore any message coming from them. Therefore, protocol designers should evaluate the impact of any countermeasure to check whether the countermeasure can be used against the system.

**Critical Resource Abusing Flaw.** Certain authentication protocols (e.g., Chatterjee et al., [14], Kim et al., [16], and Mughal et al., [18]) are designed in such a way so that devices operate heavyweight computations (e.g., message decryption) before being able to determine whether a given message is to be considered or ignored (dropped). In this case, resource-constrained devices that operate on batteries of limited lifetime can be abused for the purpose of draining their batteries (battery depletion attack [109]) by flooding those devices with bogus encrypted messages. Therefore, authentication protocols should be designed in such a way so that devices can easily and with the lowest cost determine whether to process or ignore a received message. For example, a message authentication code can be varied by computing an HMAC function.

**Ignoring Insider Threats.** We have observed that a large number of the reviewed protocols do not consider insider threats (e.g., [10, 12–15, 17, 21, 23, 24]). This makes these authentication protocols vulnerable to CRP disclosure attacks in the case where the “trusted” server is compromised (either by a remote attacker or by an insider). In the field of Information Technology Security, in general, and security protocols, in particular, considering insider threats is important and critical. In fact, a disgruntled employee (insider) commonly represents a higher risk than a black-hat hacker (outsider). Although outsider threats are more common than insider threats, insider threats still constitute a serious threat that cannot be ignored. Therefore, we recommend to PUF-based authentication designers to consider both insider and outsider threats when developing a secure protocol.

Last but not least, we point out that all reviewed protocols are vulnerable to connection deprivation through race-condition attacks [106–109]. In fact, existing authentication protocols, in general, and the reviewed authentication protocols, in particular, follow a state machine that transits from one state to another based on the first unauthenticated message that is received. That is, if the protocol is in a state of expecting the reception of a message, then it will transit to another state upon the reception of that message so that it processes the message and takes further decisions (e.g., reply or abort). This sounds totally consistent with respect to standard protocol behaviors. Notwithstanding, if we consider an attacker model where it is possible for an attacker to interfere during the execution of the protocol, then the message that is received could be a modified copy (e.g., containing incorrect values) of the expected message. Consequently, if the modified message is received before the genuine one, then the receiving device will be misled to fail the execution of the protocol (e.g., wrong password derivation). This generally ends up on the occurrence of a denial of service attack, where devices are deprived from being able to successfully establish authentication and get connected to a network as it was demonstrated in [106–108]. For example, in the protocol of Boyapally et al., [9], during the authentication of a legitimate meter with a server (viz. Fig. 2), if the attacker injects the value of $x = 0$ before the meter sends the value of $\beta_2$, the server will check whether $\gamma_2$ is equal to 0. The server will find that it is not and abort the authentication. The attacker will repeat this attack again and again to deprive the legitimate meter from performing a successful authentication. The attack can also be run on the server, for example, by sending the value of $x = 0$ before the server sends the value of $\gamma_1$. The meter will find that $\beta_1$ is not equal to 0 and abort the authentication. Thus, future authentication protocols should implement a countermeasure to thwart these attacks.
5 CONCLUSION

The service of authentication constitutes the spine of all security properties. It is the phase where entities prove their identities to each other and generally establish and derive cryptographic keys to provide confidentiality, data integrity, non-repudiation, and availability. Due to the heterogeneity and the particular security requirements of IoT (Internet of Things), developing secure, low-cost, and lightweight authentication protocols has become a serious challenge.

There has been a significant attention from the research community and the industry to develop lightweight and secure-by-design authentication protocols for IoT applications by adopting PUFs (Physical Unclonable Functions) technology. Many subsequent works have been made to propose secure, and lightweight PUF-based authentication protocols. This has noticeably turned our attention to investigate the security of the most recently published PUF-based authentication protocols.

In this paper, we have started by giving a brief overview of PUFs, their types, and related attacks. Then, we have reviewed the security of recent PUF-based authentication protocols. For each protocol, we have demonstrated the feasibility of some attacks. We have drawn lessons and proposed recommendations to be considered while developing future PUF-based authentication protocols so that the future protocols can be free from the identified security design flaws. Finally, we have summarized the major lessons by presenting six common security design flaws: (1) CRP disclosure flaw, (2) Xor operator misuse flaw, (3) authentication session uniqueness flaw, (4) countermeasure inverse consequences flaw, (5) critical resource abusing flaw, and (6) ignoring insiders threats. Furthermore, we have pointed out the vulnerability of existing authentication protocols to race condition-based attacks and future authentication protocols should implement a countermeasure to make the protocols smart and resilient against those attacks.

Finally, we have taken advantage of the learned security design lessons to design and implement a PUF-based mutual authentication protocol (T2T-MAP) for Thing-to-Thing architectures [117]. The protocol is secure against the reported attacks and is conform to IoT security and performance requirements.

ACKNOWLEDGEMENT

This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada Research Chairs (CRC) Program.

REFERENCES

[1] The-Guardian. "DDoS Attack that Disrupted Internet Was Largest of Its Kind in History, Experts Say," https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet, 2016.
[2] NewYork-Times. "Stuxnet Worm Attack on Iranian Nuclear Facilities," https://www.nytimes.com/2011/01/16/world/middleeast/16stuxnet.html, 2011.
[3] K. Zetter. "Inside The Cunning, Unprecedented Hack of Ukraine’s Power Grid," https://www.wired.com/2016/03/inside-cunning-unprecedented-hackukraines-power-grid, 2016.
[4] TheRegister. "Finns Chilling as DDoS Knocks out Building Control System," https://www.theregister.com/2016/11/09/finns_hilling_ddos_nocks_at_building_control_system/, 2016.
[5] The-Guardian. "Fiat Chrysler Recalls 1.4m Vehicles in Wake of Jeep Hacking Revelation," https://www.theguardian.com/business/2015/jul/24/jeep-chrysler-recall-jeep-hacking, 2015.
[6] BleepingComputer. "BrickerBot Dev Claims Cyber-Attack That Affected Over 60,000 Indian Modems," https://www.bleepingcomputer.com/news/security/brickerbotdev-claims-cyber-attack-that-affected-over-60-000-indian-modems, 2017.
[7] E. Ronen, A. Shamir, A.-O. Weingarten, and C. O’Flynn, "IoT Goes Nuclear: Creating a Zigbee Chain Reaction," IEEE Security and Privacy, vol. 16, no. 1, pp. 54-62, 2018.
[8] B. Schneier, "Click Here to Kill Everybody: Security and Survival in a Hyper-connected World" 2W. Norton & Company, Chpt. 5, pp. 2, 2018.
[9] H. Boyapally, P. Mathew, S. Patranabis, U. Chatterjee, U. Agrawal, M. Maheshwari, S. Dey, and D. Mukhopadhyay. “Safe Is The New Smart: PUF-based Authentication for Load Modification-Resistant Smart Meters”, IEEE Transactions on Dependable and Secure Computing, pp. 1-18, IEEE, 2020.

[10] G. Bansal, Naren, V. Chamola, B. Sikdar, N. Kumar, M. Guizani. "Lightweight Mutual Authentication Protocol for V2G Using Physical Unclonable Functions", In IEEE Transactions on Vehicular Technology, pp. 1-12, 2020.

[11] M. A. Qureshi and A. Munir. "PUF-IPA: A PUF-based Identity Preserving Protocol for Internet of Things Authentication," in the Proceedings of the IEEE 17th Annual Consumer Communications and Networking Conference (CCNC), pp. 1-7, 2020.

[12] V. P. Yanambaka, S. P. Mohanty, E. Kougianos, D. Puthal, L. Rachakonda. "PMsec: PUF-based Energy-Efficient Authentication of Devices in the Internet of Medical Things (IoMT)," in the IEEE International Symposium on Smart Electronic Systems (INES), pp. 329-321, 2019.

[13] Y. Nozaki and M. Yoshikawa. "Secret Sharing Scheme Based Secure Authentication for Physical Unclonable Function," in the Proceedings of the IEEE 4th International Conference on Computer and Communication Systems, pp. 445-449, 2019.

[14] U. Chatterjee, V. Govindan, R. Sadhukhan, D. Mukhopadhyay, R. S. Chakraborty, D. Mahata, M. M. Prabhu. "Building PUF Based Authentication and Key Exchange Protocol for IoT Without Explicit CRPs in Verifier Database", in IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 3, pp. 424-437, 2019.

[15] W. Liang, S. Xie, J. Long, K-C. Li, D. Zhang, and K. Li. "A Double PUF-based RFID Identity Authentication Protocol in Service-Centric Internet of Things Environments," in the Information Sciences Journal, vol. 503, pp. 129-147, Elsevier, 2019.

[16] B. Kim, S. Yoon, Y. Kang and D. Choi. "PUF-based IoT Device Authentication Scheme," in the International Conference on Information and Communication Technology Convergence, pp. 1460-1462, 2019.

[17] M. H. Mahalat, S. Saha, A. Mondal, B. Sen. "A PUF based Light Weight Protocol for Secure WiFi Authentication of IoT devices," in the Proceedings of the 8th International Symposium on Embedded Computing and System Design (ISED), pp. 183-187, IEEE, 2018.

[18] M. A. Mughal, X. Luo, Z. Mahmood, and A. Ullah. "Physical Unclonable Function Based Authentication Scheme for Smart Devices in Internet of Things,", in the Proceedings of the 2018 IEEE International Conference on Smart Internet of Things, pp. 160-165, 2018.

[19] M. Barbarechi, A. De Benedictis, N. Mazzeo. “A PUF-based Hardware Mutual Authentication Protocol;” in Journal of Parallel Distrib. Comput. vol. 119, pp. 107-120, 2018.

[20] M. Barbarechi, A. De Benedictis, E. La Montagna, A. Mazzeo. “A PUF-based Mutual Authentication Scheme for Cloud-Edges IoT Systems,” in FGCSs journal, vol. 10, pp. 246-261, 2019.

[21] W. Feng, Y. Qin, S. Zhao, D. Feng. "AAoT: Lightweight Attestation and Authentication of Low-Cost Things in IoT and CPS", Computer Networks Journal, vol. 134, pp. 167-182, Elsevier, 2018.

[22] Y. Yilmaz and S. R. Gunn and B. Halak. “Lightweight PUF-based Authentication Protocol for IoT Devices,” in the IEEE 3rd International Verification and Security Workshop (IVSW), pp. 38-43, 2018.

[23] T. Idriess and M. Bayoumi. "Lightweight Highly Secure PUF Protocol for Mutual Authentication and Secret Message Exchange", 2017 IEEE International Conference on RFID Technology and Application (RFID-TA), pp. 214-219, IEEE, 2017.

[24] V. Czepak and V. Zeman. “Robust Mutual Authentication and Secure Transmission of Information on Low-cost Devices Using Physical Unclonable Functions and Hash Functions” in the Proceedings of the 39th International Conference of Telecommunication and Signal Processing, pp. 100-103, 2016.

[25] P. S. Ravikanth, “Physical One-way Functions,” Ph.D. thesis, Cambridge, MA, USA, AAI0803255, 2001.

[26] D. S. Boning and J. E. Chung. "Statistical Metrology: Understanding Spatial Variation in Semiconductor Manufacturing," in Proceedings of SPIE 1996, Symposium on Microelectronic Manufacturing, 1996.

[27] K. A. Bowman, S. G. Duvall, and J. D. Meindl. "Impact of Die-to-Die and Within Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Gigascale Integration," Journal of Solid-State Circuits, vol. 37, no. 2, pp. 183-190, 2002.

[28] S. R. Nasiri. "Modeling and Forecasting of Manufacturing Variances," In Proceedings of ASP-DAC, 2001.

[29] K. C. Mugali and M. M. Patil, "Device Authentication by Physical Unclonable Functions," in the Proceedings of the 2018 IEEE International Conference on Computing Communication Control and Automation, pp. 327-329, Feb 2015.

[30] H. Sun, M. A. Alemohammad, B. T. Bosworth, B. C. Grubel, A. B. Cooper, M. A. Foster, and A. C. Foster. "Photonic Physical Unclonable Functions using Silicon Nitride Spiral Cavities", in Conference on Lasers and Electro-Optics, p. SThN-4, Optical Society of America, 2017.

[31] U. Ruhmair, F. Sennke, J. Solter, G. Dror, S. Devadas, and J. Schmidhuber, “Modeling Attacks on Physical Unclonable Functions,” in Proceedings of the 17th ACM conference on Computer and Communications Security, pp. 237-249, ACM, 2010.

[32] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, R. J. Young. "A PUF taxonomy," in Applied Physics Reviews, Rev. 6, no. 011303, pp. 1-25, AIP Publishing, 2019.

[33] Y. Dodis, B. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys From Biometrics and Other Noisy Data," In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT (Advances in Cryptography), vol. 3027, LNCS, pp. 523-540, Springer, 2004.
[35] A. Juels and M. Wattenberg, “A Fuzzy Commitment Scheme,” in Proceedings of the 6th ACM conference on Computer and Communications Security, pp. 28-36, ACM, 1999.
[36] G. E. Suh and S. Devadas, “Physical Uncloneable Functions for Device Authentication and Secret Key Generation,” Proceedings of the 44th annual Design Automation Conference. ACM, 2007.
[37] J. H. Anderson, “A PUF Design for Secure FPGA-based Embedded Systems,” in Proceedings of 15th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 1-6, 2010.
[38] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, “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, 2007.
[39] F. Tehranipoor, N. Karimian, K. Xiao, and J. A. Chandy, “DRAM-based Intrinsic Physical Uncloneable Functions for System Level Security,” in Proceedings of the 25th Edition on Great Lakes Symposium on VLSI, pp. 15-20, 2015.
[40] L. Lin, D. Holcomb, D. K. Krishnappa, P. Shahbadi, W. Burleson, “Low-power Sub-Threshold Design of Secure Physical Uncloneable Functions,” in the 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design, pp. 43-48, 2010.
[41] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gammeyer, S. Katzenbeisser, J. Szefer. “Intrinsic Rowhammer PUFs: Leveraging the Rowhammer Effect for Improved Security,” in Proceedings of IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 1-7, 2017.
[42] S. Vrijaldenhoven, “Acoustical physical uncloneable functions,” M.S. thesis, Eindhoven University of Technology, 2004.
[43] P. Tuyls, G. J. Schrijen, B. Skoric, J. v. Geloven, N. Verhaegh, and R. Wolters, “Read-Proof Hardware from Protective Coatings,” in Proceedings of Cryptographic Hardware and Embedded Systems, 2006.
[44] National-Research-Council. “Counterfeit Deterrent Features for the Next-Generation Currency Design,” The National Academies Press, Publication NMA472, pp. 1-144, https://doi.org/10.17226/2267, 1993.
[45] G. Hammouri, A. Dana, and B. Sunar, “CDs have fingerprints too,” in Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), pp. 348-362, 2009.
[46] G. Lenzini, S. Ouchani, P. Roemme, P. Y. A. Ryan, Y. Geng, J. Lagerwall, J. Noh. “Security in The Shell: An Optical Physical Uncloneable Function Made of Shells of Cholesteric Liquid Crystals,” in Proceedings of IEEE Workshop on Information Forensics and Security (WIFS), pp. 1-6, 2017.
[47] R. S. Indeck and M. W. Muller, “Method and Apparatus for Fingerprinting Magnetic Media,” United States Patent US5365586A, 1994.
[48] NXP. “Secure Storage with SRAM PUF on NXP LPC540xxx,” AN12292, https://www.nxp.com/docs/en/application-note/AN12292.pdf, pp. 1-17, 2018.
[49] Microsemi. “Using SRAM-PUF System Service in SmartFusion2 - Libero SoC v11.7,” Application Note AC434, pp. 1-19, 2016.
[50] IntelNewsroom. “Altera Partners with Intrinsic-ID to Develop World’s Most Secure High-End FPGA,” https://newsroom.intel.com/news-releases/altera-partners-intrinsic-id-develop-worlds-most-secure-high-end-fpga/#gs.0qyiip, 2015.
[51] CisionPRnewswire. “Verayo PUF IP on Xilinx Zynq UltraScale+ MPSoC Devices Addresses Security Demands” https://www.prnewswire.com/news-releases/verayo-puf-ip-on-xilinx-zynq-ultrascale-mpsoc-devices-addresses-security-demands-300357805.html, 2016.
[52] A. Maiti and P. Schaumont, “Improving the Quality of a Physical Uncloneable Function using Configurable Ring Oscillators,” in the 2009 International Conference on Field Programmable Logic and Applications, pp. 703-707, IEEE, 2009.
[53] S. Pandey, S. Deyati, A. Singh, and A. Chatterjee, “Noise-Resilient SRAM Physically Uncloneable Function Design for Security,” in IEEE 25th Asian Test Symposium (ATS). IEEE, pp. 55-60, 2016.
[54] D. Jeon, J. H. Baek, D. K. Kim, and B. D. Choi, “Towards Zero Bit Error-Rate Physical Uncloneable Function: Mismatch-based vs. Physical-based Approaches in Standard CMOS Technology,” in 2015 Euromicro Conference on Digital System Design. IEEE, pp. 407-414, 2015.
[55] K. H. Chuang, E. Bury, R. Degraeve, B. Kaczer, D. Lenten, and I. Verbauwhede, “A Physically Uncloneable Function using Soft Oxide Breakdown Featuring 9% Native BER and 51.8 fJ/bit in 40-nm CMOS,” IEEE Journal of Solid-State Circuits, vol. 54, no. 10, pp. 2765-2776, 2019.
[56] X. Lu, L. Hong, and K. Sengupta, “CMOS Optical PUFs using Noiseimmune Process-Sensitive Photonic Crystals Incorporating Passive Variations for Robustness,” IEEE Journal of Solid-State Circuits, vol. 53, no. 9, pp. 2709-2721, 2018.
[57] W. C. Wang, Y. Yona, S. N. Diggavi, and P. Gupta, “Design and Analysis of Stability-Guaranteed PUFs,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 4, pp. 978-992, 2017.
[58] R. Anderson and M. Kuhn, “Tamper Resistance: A Cautionary Note,” in Proceedings of the 2Nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce, vol. 2, pp. 1-1, USENIX Association, 1996.
[59] R. J. Anderson and M. G. Kuhn, “Low Cost Attacks on Tamper Resistant Devices,” in Proceedings of the 5th International Workshop on Security Protocols, pp. 125-136, Springer, 1998.
[60] A. C. D. Resende, K. Mochetti, and D. F. Aranha, “PUF-based Mutual Multifactor Entity and Transaction Authentication for Secure Banking,” Light Cryptogr. Secur. Priv., pp. 77-96, 2015.
[61] T. Esbach, W. Famy, O. Kulikovska, D. Merli, D. Schuster, F. Stumpf, “A New Security Architecture for Smartcards Utilizing PUFs,” in Proceedings of 2012 Securing Electronic Business Processes, pp. 180-194, Springer, 2012.

[62] D. Li, W. Liu, X. Zou, and Z. Liu, “Hardware IP Protection Through Gatelevel Obfuscation,” in Proceedings of the 14th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics), pp. 186-193, IEEE, 2015.

[63] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls, “Extended Abstract: The Butterfly PUF Protecting IP on Every FPGA,” in Proceeding of the IEEE International Workshop Hardware-Oriented Security and Trust, pp. 67-70, 2008.

[64] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, “Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection,” in Proceedings of the International Conference on Field Programming Logic Application, pp. 189-195, 2007.

[65] E. Simpson and P. Schaumont, “Offline Hardware/Software Authentication for Reconfigurable Platforms,” in Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, vol. 4249 of LNCS, pp. 311-323, 2006.

[66] M. A. Gora, A. Maui, and P. Schaumont, “A Flexible Design Flow for Software IP Binding in Commodity FPGA,” in Proceedings of the IEEE International Symposium on Industrial Embedded Systems, pp. 211-218, 2009.

[67] Y. Zheng, Y. Cao, C.-H. Chen. “A PUF-Based Data-Device Hash for Tampered Image Detection and Source Camera Identification,” in IEEE Transactions on information forensics and security, vol. 15, pp. 620-634, 2020.

[68] K. Mäljär, R. Ulrich, A. Stanitzki, and R. Kokozinski, “Enabling Secure Boot Functionality by Using Physical Unclonable Functions,” in 14th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), pp. 81-84, 2018.

[69] J. Zhang and G. Qu. “Physical Unclonable Function-Based Key Sharing via Machine Learning for IoT security,” in IEEE transactions on Cybernatics, vol. 43, no. 11, pp. 2864-2874, 2013.

[70] M. Spain, B. Fuller, K. Ingols, and R. Cunningham. “Robust Keys from Physical Unclonable Functions,” in 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 88-92, 2014.

[71] O. Gönül, O. Akşcan, and G. Kramer, “Reliable Secret Key Generation from Physical Unclonable Functions under Varying Environmental Conditions,” in 2015 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1-6, 2015.

[72] Y. Kang, M. G. Khan, and Y. Nozaki, “Towards Smart and Secure IoT Systems: A Quad-PUF-Based Approach,” in IEEE Transactions on Information Forensics and Security, vol. 15, no. 1, pp. 70-83, 2020.

[73] Z. Paral and S. Devadas, “Reliable and Efficient PUF-based Key Generation using Pattern Matching,“ in 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, pp. 128-133, June 2011.

[74] J. Zhang, L. Wang, Q. Wu, and G. Qu. “DMOS-PUF: Dynamic Multi-Key-Selection Obfuscation for Strong PUFs against Machine Learning Attacks. arXiv preprint arXiv:1806.02011, 2018.

[75] J. Delvaux. “Machine-Learning Attacks on PolyPUFs, OB-PUFs, RP-PUFs, LHS-PUFs, and PUFâšблагSMs,” IEEE Transactions on Information Forensics and Security, vol. 14, no. 8, pp. 2222-2236, 2019.

[76] U. Rührmair, J. Söller, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas. “PUF Modeling Attacks on Simulated and Silicon Data,” IEEE Trans. Information Forensics and Security, vol. 8, no. 11, pp. 1876-1891, 2013.

[77] J. Tobisch and T. B. Georg, “On the Scaling of Machine Learning Attacks on PUFs with Application to Noise Bifurcation,” in Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems, pp. 302-319, Springer, 2012.

[78] S. S. Avvaru, Z. Zeng, and K. K. Parhi. “Homogeneous and Heterogeneous Feed-Forward XOR Physical Unclonable Functions.” In IEEE Transactions on Information Forensics and Security, vol. 15, pp. 2485-2498, 2020.

[79] J. Zhang, C. Shen. “Set-Based Obfuscation for Strong PUFs Against Machine Learning Attacks”. IEEE Transactions on Circuits and Systems, vol. 67, no. 8, pp. 7025-7033, 2020.
[88] S. Wei, J. B. Wendt, A. Nahapetian, and M. Potkonjak, "Reverse Engineering and Prevention Techniques for Physical Unclonable Functions using Side Channels," in 51st ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1-6, 2014.

[90] U. Räihä, X. Xu, A. Majumdar, X. Wang, and W. Burleson, "Efficient power and timing side channels for physical uncloneable functions," in Cryptographic Hardware and Embedded Systems - CHES 2014, (Berlin, Heidelberg), pp. 476-492, Springer Berlin Heidelberg, 2014.

[92] J. Delvaux and I. Verbauwhede, "Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise," in 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 137-142, 2013.

[94] Y. Nozaki and M. Yoshikawa, "Power Consumption Aware Machine Learning Attack for Feed-Forward Arbiter PUF," pp. 49-62, 2019.

[96] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Semi-invasive EM Attack on FPGA RO-PUFs and Countermeasures," in Proceedings of the Workshop on Embedded Systems Security, WESS'11, (New York, NY, USA), vol. 2, pp. 1-9, ACM, 2011.

[98] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-Channel Analysis of PUFs and Fuzzy Extractors," Conference on Trust and Trustworthy Computing (TRUST 2011), vol. 6740 of LNCS, pp. 33-47, 2011.

[100] J. Long, W. Li, K. H. Li, D. Zhang, M. Tang, H. Luo, "PUF-based Anonymous Authentication Scheme for Hardware Devices and IPs in Edge Computing Environment," in IEEE Access journal, vol. 7, pp. 124785-124796, 2019.

[102] D. Liu, Z. Liu, Z. Yong, X. Zou, J. Cheng, "Design and Implementation of An ECC-Based Digital Baseband Controller for RFID Tag Chip," IEEE Transactions on Industrial Electronics, vol. 62, no. 7, pp. 4365-4373, 2015.

[104] J. Delvaux and I. Verbauwhede, "Side Channel Modeling Attacks on 65nm Arbiter PUFs Exploiting CMOS Device Noise," in 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 137-142, 2013.
[117] K. Lounis. "Security of Wireless Short-Range Technologies and an Authentication Protocol for IoT," Ph.D. Thesis, School of Computing, Queen’s University, pp. 1-323, 2020.