More Lessons: 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
Kingston, Ontario, CANADA
{karim.lounis, mz}@queensu.ca

***Abstract*—Authentication constitutes the foundation and vertebrae of all security properties. It is the procedure in which communicating parties prove their identities to each other, and generally establish and derive secret keys to enforce other services, such as confidentiality, data integrity, non-repudiation, and availability. 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 [1], we reviewed the security of some PUF-based authentication protocols that were proposed between 2016 and October 2020, and drew important security lessons to consider by future authentication protocol designers. In this paper, we extend our previous work by reviewing the security of fifteen PUF-based authentication protocols that were recently published during the past two years (2020 and 2021). We first provide the necessary background on PUFs and how they are used for authentication. Then, we analyze the security of these authentication protocols to identify and report common security issues and design flaws. We draw lessons and recommendations for future authentication protocol designers.**

***Index Terms*—Physical Unclonable Functions (PUFs), PUF-based authentication, PUF security, and PUF attacks.**

I. Introduction

Authentication is the procedure in which communicating parties prove their identities to each other, and generally establish and derive secret keys to enforce other services, such as confidentiality, data integrity, non-repudiation, and availability. 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 [25]–[28]. In the meantime, researchers have turned their attention to PUF technology to develop lightweight and secure-by-design authentication protocols for IoT applications.

PUFs 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. 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 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. 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 that were published during the past two years (2020 and 2021). 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. This would constitute a complementary work for our previous contribution in [1] where we analyzed the security of 15 PUF-based authentication protocols that were published between 2016 and 2020, and drawn security lessons for future designs.

To summarize, the contributions of this paper are twofold: (i) We analyze the security of some PUF-based authentication protocols that were published between the year of 2020 and 2021, and identify possible attacks. (ii) 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 II, 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 III reviews the design and the security of PUF-based authentication protocols that were published during the past two years (2020 and 2021). We draw lessons and recommendations for future PUF-based authentication protocol development. Some of the drawn lessons are complementary to the ones provided in [1]. In Section IV, we summarize and discuss the lessons learned. We conclude the paper in Section V.

II. PUFs (Physical Unclonable Functions)

A. PUFs Overview

PUFs (Physical Unclonable Functions), also known as POWFs (Physical One-Way Functions) [2] or PRFs (Physical Random Functions) [3], 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, which stands for Challenge-Response Pairs.

In the electronics and nanotechnology disciplines, PUFs[1] are constructed from the idiosyncrasies in the manufacturing process of semiconductors [4]–[6] (e.g., inherent delay characteristics of wires and transistors [7]) to assign uniquely distinct fingerprints to semiconductor pieces [8]. Thus, physical cloning of a piece of semiconductor becomes very difficult. This for example, allows distinguishing a genuine integrated circuit from a counterfeit one (including exact copies produced during the same manufacturing process [8]). 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.

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[2] [9]. 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* [10]. 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 [11], [12].

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.

B. 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 [3], ring oscillator-PUFs [13], and glitch-PUFs [14]), memory-based PUFs (e.g., SRAM-PUFs [15], DRAM-PUFs [16], SR Latch PUF [17], and Rowhammer-PUFs [18])[3], acoustical-PUFs [19], coating-PUFs [20], optical-PUFs (e.g., paper-PUFs [21], Compact Disc-PUF [22], and liquid crystal-PUF [23])[4], and magnetic-PUFs [24]. 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., [10], 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 P60 microcontroller and LPC54S0xx family of microcontrollers [25]. Also, Microsemi company has adopted SRAM-PUFs on their SmartFusion2 System-on-Chip FPGA devices [26]. Other

[1]When a PUF is constructed from a semiconductor, which is generally made of Silicon, the PUF is categorized as a Silicon-PUF (SPUF).

[2]Strong PUFs, such as arbiter-PUF, are generally used for device authentication, whereas weak PUFs are mainly used in key generation.

[3]Delay-based and memory-based PUFs are often classified as electronic CMOS (Complementary Metal Oxide Semiconductor)-PUFs.

[4]In some literature, coating-based and optical-based PUFs are often classified as MEMS (Micro-Electro-Mechanical Systems)-based PUFs.

hardware from various companies have adopted SRAM-PUFs to implement physical security, such as Coherent Logix HyperX, Intel Altera Stratix 10 FPGA [27], Redpine Signals WyzBee, and the Xilinx Zynq Ultrascale+ [28]. 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. Also, 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[5] of PUFs, in general, is affected by environmental fluctuations, which include temperature variations, voltage variation, and ambient noise [30]–[34]. 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 instances of 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.

C. 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 [35], [36].

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 [3]. The protocol consists of two main phases, the registration phase (a.k.a., enrolment 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_{id} will have its proper set of challenge and response pairs in the database $\Gamma_{id} = \{(c_0, r_0), \ldots, (c_n, r_n)\}$.

The authentication of a registered device d_{id} consists of randomly selecting a pair of registered challenge and response $(c_i', r_i') \in \Gamma_{id}$ and interrogating the device d_{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., $r_i' \simeq r_i''$, the device d_{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[6] of Fig. 1, where $\Psi_p(\cdot)$ 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 [37].

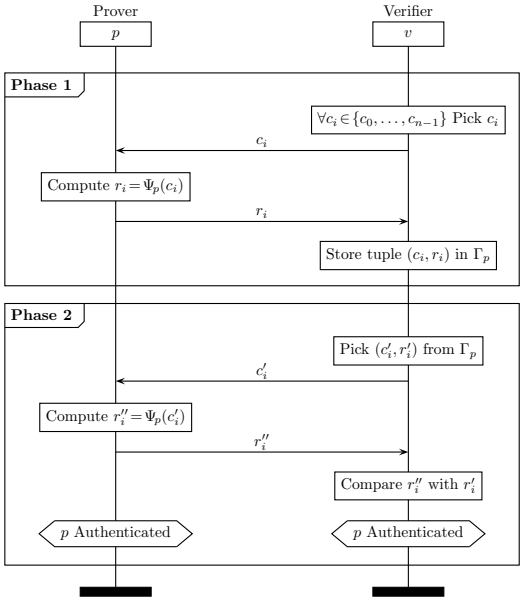

Fig. 1: A typical PUF-based authentication protocol using challenge-response mechanism, where **Phase 1** is the registration phase and **Phase 2** is the authentication phase.

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 [38] and FPGA circuits [39]–[42]), software components (e.g., software IP binding [43] and images [44]), and embedded devices' firmware integrity [45]. They are also employed for generating cryptographic keys and random numbers [46]–[50], and securing memories and processors.

D. 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

[5]The 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 [29].

[6]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) [59].

so that the PUF's output gets altered. Therefore, any future authentication attempt 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 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.

Side-Channel Attack. 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) 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 registered devices will be revealed. This would allow attackers to impersonate the PUFs of those devices.

III. Security Analysis of PUF-based Authentication Protocols

In this section, 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. Also, we distinguish two categories of protocols based on the communication scheme for which each protocol was designed for. The first category represents 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, whereas the second category groups protocols that were designed for the T2T scheme, where two resource-constrained devices authenticate each other. This classification is based on the corresponding authors' claim when implementing the protocols.

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 [59]. 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. 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 I summarizes the differences between the reviewed PUF-based authentication protocols for T2M and T2T schemes, respectively. 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

Scheme	Protocol	Used PUF	Protocol properties P1	P2	P3	P4	P5	Evaluation P6	P7	P8	Implementation platform	Possible attacks on the protocol
T2M	**Satamraju et al.,** [56] (2020)	Arbiter PUF RO-PUF	✗	✓	•	✗	✓	✗	✓	✗	Nexys A7 board Artix-7 FPGA & Raspberry Pi 3B+	Spoofing attacks DoS attack CRP disclosure
	Zerrouki et al., [57] (2021)	Arbiter PUF	✗	✓	•	✗	✗	✗	✗	✗	Simulation	Spoofing attack CRP disclosure MITM attack DoS attack Modeling attack
T2T	**Narwal et al.,** [51] (2020)	Unspecified	✗	✗	•	✗	✗	✓	✓	✗	—	Spoofing attack Desing bug Key cracking
	Ghafi et al., [52] (2020)	Unspecified	✓	✗	✓	✗	✗	✓	✓	✗	Simulation	Spoofing attack DoS attack
	Zheng et al., [58] (2021)	Unspecified	✗	✓	•	✗	✓	✓	✗	✗	—	Spoofing attack CRP disclosure DoS attack
	Yoon et al., [53] (2020)	Unspecified	✗	✓	•	✗	✓	✓	✗	✗	—	Spoofing attack DoS attack
	Quershi et al., [54] (2021)	3-1 DPUF	✗	✓	✓	✗	✓	✓	✓	✓	Xilinx Zynq-7000 zc706 board	Spoofing attack Key cracking CRP disclosure
	Tan et al., [55] (2021)	RO-PUF	✓	✓	•	✗	✗	✓	✓	✗	Xilinx Zynq-7000 zc706 board & OMNET++	Spoofing attack DoS attack CRP disclosure Modeling attack

TABLE I: 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 Pi, where P1: Mutual authentication using PUFs, P2: Lightweight (no asymmetric cryptography and no preshared keys), P3: Scalability, P4: Smart, P5: Key establishment (for message authenticity, confidentiality, and integrity), P6: Security evaluation, P7: Performance evaluation, P8: PUF-circuit size evaluation. The symbol ✓, ✗, and •, indicate **Yes**, **No**, and **Possible**, respectively.

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 [60], [61], [63], 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 below.

A. PUF Yoking-based Authentication Protocol

In [51], Narwal et al., proposed a PUF-based authentication protocol for wearable devices. The protocol relies on Yoking-proof and a cloud server to allow wearable devices get mutually authenticated to a user's smartphone. The authentication protocol was empirically evaluated and its security was discussed. Nevertheless, we have analyzed the protocol and found the following security issues:

Bug. The protocol has a serious design issue that causes a self-denial of service. In fact, after the server receive its first authentication message, it uses the secret S_{WD_i} and its proper PUF function to compute the secrets $S^{\sharp}_{WD_i}$ and $S^{\sharp\sharp}_{WD_i}$. Then, it uses these secretes to compute the parameters A_{WD_i}, B_{WD_i}, C_{WD_i}, and D_{WD_i}, to be sent to the wearable device. At this stage, the value of $S^{\sharp\sharp}_{WD_i}$ that the wearable device holds is different from the once that was computed by the server using its PUF. Therefore, any future authentication attempt would automatically fail and the protocol will be stuck forever.

Spoofing Attack. Regardless the design issue pointed out above, and from an insider viewpoint, the protocol is vulnerable to spoofing attack. In fact, the mutual authentication of a wearable device and the server relies on proving the knowledge of two keys: (i) the shared secret S_{WD_i} and (ii) the shared session key $sKey$. The first key allows a particular wearable device to prove its identity as this key is device-specific. It also allows the device to verify that it is communicating with the server. The second key seems to be a session key that is shared among the server and all connected wearable devices. It allows a device to prove that a message has not been tampered and its source is authentic. In this case, since all the connected wearable devices know the session key $sKey$, a malicious wearable device, WD_j would just need to discover the secret key S_{WD_i} of the wearable device WD_i to be able to impersonate it. For that end, the malicious device intercepts an authentication session that involves the wearable device WD_i (e.g., let us assume WD_k authenticating WD_i). From the first message that the server sends to device WD_i, the attacker computes the new value of $S^{\sharp\sharp}_{WD_i} = C_{WD_i} \oplus D_{WD_j} \oplus \alpha$ that will be used in the next authentication session. Also, the value of $S^{\sharp}_{WD_i}$ is

updated to $S^{\sharp\sharp\sharp}_{WD_i}$, which would make $S^{\sharp\sharp\sharp}_{WD_i} == S^{\sharp}_{WD_i}$. Then, during a second authentication session, the value of the secret S_{WD_i} is set to $S^{\sharp\sharp\sharp}_{WD_i}$, which is known by the attacker. Thus, at the end of the second authentication session, the attacker would know the values of S_{WD_i}, $S^{\sharp}_{WD_i}$, and $S^{\sharp\sharp\sharp}_{WD_i}$, that will be used during the next authentication session. Now the attacker needs to determine the pseudorandom identifier of the target wearable device (i.e., $PRID_i$). The authors did not mention whether this information is kept secret or available to the public. If the information is not secret, then the attacker has all the required parameters to impersonate another wearable device. Otherwise, if the pseudorandom identifier is a secret information, the attacker may try to brute-force it offline using the captured value of $\mathcal{H}(PRID_i \oplus \alpha)$. This would be quickly performed if the identifier is in 16 bits. Furthermore, the server has access to all the information about all the connected wearable devices. Therefore, if the server is compromised, or acting as a malicious insider, it will be able to impersonate every single connected wearable device.

Key Cracking. In this protocol, the authentication is actually relying on the secrecy of the session key $sKey$. This key is shared among all wearable devices and hence can be used to compromise the service of non-repudiation (due to its symmetric nature). From an outsider perspective, an attacker can intercept messages exchanged during authentication sessions and compute the keys S_{WD_i}, $S^{\sharp}_{WD_i}$, and $S^{\sharp\sharp\sharp}_{WD_i}$ that will be used during the next authentication session as we have pointed out above. What the attacker is missing is the secret key $sKey$, which is not transmitted during authentication sessions. The attacker will have to brute-force the key to crack it. The current size of the key is 64 bits, and there is no key updating procedure in the current design of the protocol. This would increase the chance for an attacker to brute-force the key.

Lessons

1. We have seen, in this authentication protocol, how a malicious insider took advantage of a legitimate authentication session and used the learned information to impersonate another device. This issue of not considering insider threats has become common design flaw that puts the security of authentication protocols into question. A manual analysis of insider threats is highly recommended since most automated protocol verifiers cannot detect such attack scenarios.

2. The authentication should not be realized through the use of one single long-term shared key. This constitute a single point of failure for the system. In this protocol, the authentication is strongly relying on the secrecy of the session key $sKey$. This key is known by all enrolled devices, which voids the service of non-repudiation. If this key is revealed to an outsider, the protocol is compromised as discussed above.

B. Distributed PUF-based Mutual Authentication Protocol

Ghafi et al., [52] proposed a distributed PUF-based mutual authentication protocol with self-correction. The protocol allows two IoT devices to perform mutual authentication using PUFs, asymmetric cryptography, and smart contracts in blockchain. The protocol was simulated and its response time was evaluated. The protocol claims resilience to many attacks, including spoofing, man-in-the-middle, replay, and DoS attacks.

In this authentication protocol, each device deploys a smart contract in the blockchain where one CRP for the device is stored. When two devices need to get authenticated, each device proves its identity by successfully performing an authentication with the related smart contract (on-chain phase) and by successfully encrypting a nonce using a key that is sent to the other party only in the case where the previous authentication with the smart contract succeeds (off-chain phase). For instance, if a device d_i wants prove its identity to another device d_j, device d_i needs first to authenticate itself to the related smart contract and then sends back an encrypted nonce that device d_j previously created. If the authentication with the smart contract succeeded, the smart contract sends to device d_j the key to decrypt the nonce that d_i has sent. If the result is the same as the initial nonce that d_j had created, then d_j authenticates d_i. Then, in order for device d_j to authenticate device d_i, device d_j would execute the same steps that device d_i executed to establish a mutual authentication.

We have analyzed the protocol and found the feasibility of the following attacks:

DoS Attack. The protocol uses a countermeasure that blocks the devices from performing any future authentication if the device fails three successive authentication attempts. An attacker may exploit this countermeasure to cause a denial of service on the devices by impersonating the latter and intentionally causing multiple failed authentications. This would block legitimate devices from using the system and performing authentication. Moreover, removing this countermeasure and allowing multiple authentication failures would create another security flaw. The protocol would be vulnerable to denial of service through desynchronization. The protocol is designed in such a way so that at each successful authentication, the device's challenge and response pair at the smart contract is updated so that a given CRP is used only once. An attacker may try to brute-force the response of a challenge by attempting multiple authentication trials until it hits the correct response. This would update the smart contract with a newer CRP and discard the current one. In this case, the legitimate device would fail all future authentication attempts as it will be using the old CRP. Additionally, if an attacker manages to intercept the new CRP (assuming no encryption is used during the on-chain phase), it will be able to cause a desynchronization by performing a fake authentication, which would update the CRP that is stored on the smart contract with an arbitrary "fake" one.

Spoofing Attack. Storing one single CRP per device at the server and updating the CRP after each successful authentication is a type of design in which parts of the enrolment phase are inserted within the authentication phase. Therefore, if an attacker manages to determine the current CRP, it can impersonate the concerned device, run a successful authentication, and update the CRP with arbitrary values (to be used in the next authentication session). In this case, the attacker guarantees that it will probably succeed the next authentication challenge as it knows the PUF response to be used. This would work unless the legitimate device starts an authentication and fails three times. After three failures, both the attacker and the legitimate device are locked out by the system.

Lessons

3. In this protocol, the authentication is locked after three failures. It is important to note that setting up a countermeasure that can be used against the system itself is a major vulnerability. Authentication protocol designers should evaluate the inverse consequences of a countermeasure.

4. Similar to this protocol, many other protocol update the stored CRP upon a successful authentication. However, this can be exploited by an attacker through probing to update the CRP with an arbitrary value, which would fail any future legitimate authentication.

C. PUF-based Device-to-Device Authentication Protocol

In [53], Yoon et al., proposed a PUF-based mutual authentication protocol for IoT. The protocol allows two IoT devices to authenticate each other through a central server. The servers stores one CRP per device. The CRP is updated to a new CRP after each successful authentication. Also, the protocol allows the establishment of a session key to provide secrecy after the authentication (viz., Fig. 2). The authors claimed that the protocol is resilient to information leakage, replay attacks, man-in-the-middle, cloning attacks, side channel attacks, and machine learning attacks. However, we have discovered that the authentication protocol is vulnerable to spoofing attack as well as to a denial of service attack.

Spoofing Attack. In this attack scenario, we assume that one of the devices is compromised (malicious insider). The malicious device, let us say device B, starts the attack by performing a legitimate authentication with a target device, let us say device A. During this authentication, the attacker (device B) establishes the shared session key $K_S = \mathcal{H}(N_A \oplus N_S)$. Then, it uses this key to decrypt the last message that device A sends, which contains the value of R_{A_2}. Also, because device B knows the value of N_S, it computes the value of $C_{A_2} = \mathcal{H}(C_{A_1} \oplus N_S)$. At this point, the attacker has learned the new CRP of device A, i.e., (C_{A_2}, R_{A_2}). This new CRP can be used to perform a successful authentication on behalf of device A (successful spoofing).

Denial of Service. Once the attacker learns the new CRP of a given target device and then performs a fake successful authentication, the CRP of the target device gets updated at the server. In this case, any future authentication attempt from device A will certainly fail as the CRP that device A uses are not the same as the one the server uses (desynchronization).

Other Attacks. Another issue that we have noticed with this protocol is that if an attacker starts a fake authentication, the latter will be dropped after the protocol runs 21 steps out of 26. This design issue may be exploited to deplete devices' resources (e.g., battery) by running fake and spoofed authentication attempts. Moreover, since the authentication of the second device (e.g., Device B in Fig. 2) by the first device (i.e., Device A) is relies on device B successfully relaying an encrypted message from the server to device A, an attacker can impersonate device B and compromise the authentication claim at device A. This attacker is illustrated in the MSC of Fig. 3. Last but not least, as the server knows the one single CRP of each device, a malicious insider controlling the server would be able to impersonate all the devices to perform successful authentications. This poses a serious security issue in the authentication protocol.

Lessons

5. In this protocol, a malicious insider can take advantage of a legitimate authentication session and use the learned information to impersonate another device. This security flaw exists due to the negligence of insider threats during the security evaluation of the authentication protocol.

6. Similar to many protocols, this protocol updates the stored CRP upon a successful authentication. This can be exploited by an attacker through probing to update the CRP with an arbitrary value and cause authentication failures when the legitimate device tries authenticating itself (Similar to **Lesson 4**).

D. PUF-RAKE-based Authentication protocol

Quershi et al., [54] proposed PUF-RAKE, a PUF-based lightweight authentication protocol with key establishment. The protocol allows devices to be authenticated to a server and exchange valid public keys between two devices to establish a shared secret. It uses random number generators and masking functions to obfuscate the exchanged parameters during the authentication. The protocol was implemented on a Xilinx Zynq-7000 zc706 evaluation board, and the security of the protocol was discussed. The protocol is resilient to impersonation attacks (from an outsider perspective), man-in-the-middle, and information leakage. Nevertheless, the analysis

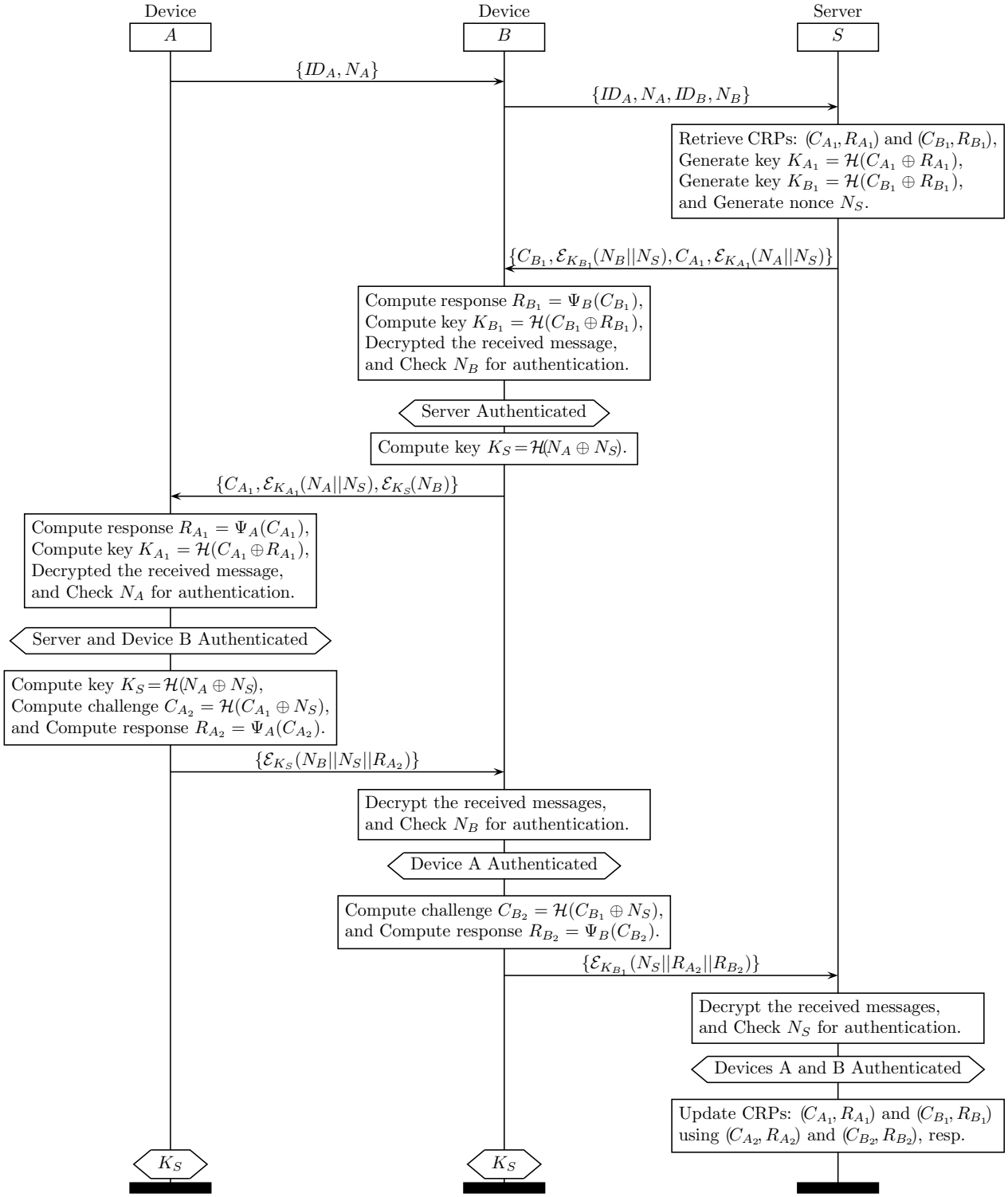

Fig. 2: Device-to-Device PUF-based Authentication Protocol by Yoon et al., [53] .

of this authentication protocol shows that the security of the system may be compromised using the following attacks:

Key Cracking. The security of the protocol relies on the secrecy of a challenge, denoted as C_s. This challenge is assumed to be stored in a secure NVM and is used to derive a secret key K_s. This key is employed to decrypt devices' entries stored in the database. Since an assumption is not a fact, it may happen that, within a given implementation, the secret challenge may be revealed to the attacker. In that case, the security of the

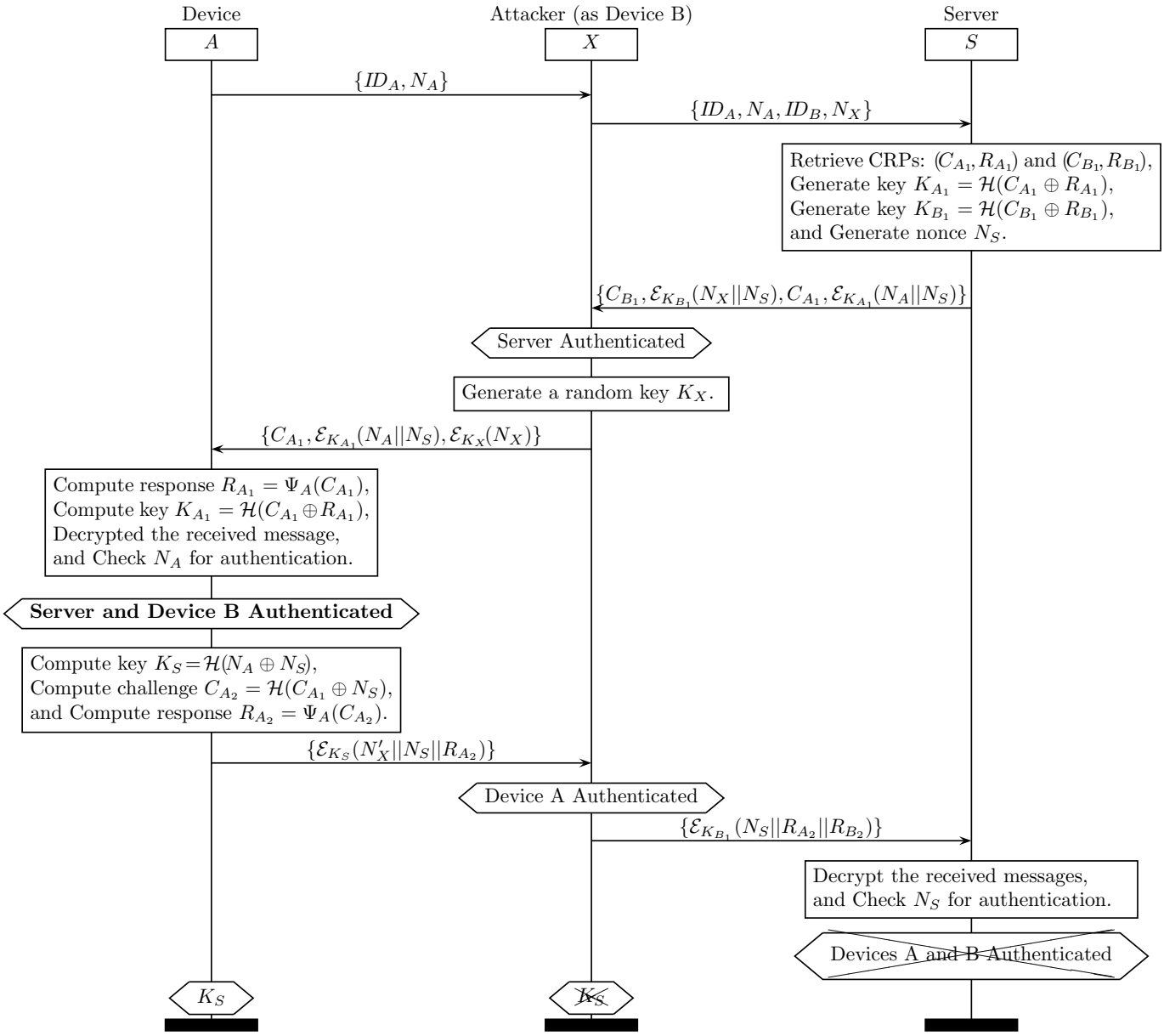

Fig. 3: Compromising the Authentication Claim of Yoon et al.,'s Authentication Protocol.

system may be compromised as the attacker can impersonate every enrolled device, including the server. Furthermore, the secret challenge C_s needs to be updated at some point as per cryptographic key security requirements; otherwise, an attacker can brute-force the key and manage to decrypt the entries of the database using the IDs of the concerned devices as a concrete indicator.

Spoofing Attack. In this protocol, the server is attributed a high-level of trust. However, if the server acts as a malicious insider (from an insider perspective), the latter can impersonate any device in the system. In fact, as it holds the secret key for decrypting devices' entries from the database, it can access all information that is required to impersonate a given device and it can disclose the CRPs of the enrolled devices as well.

Lessons

7. The long-term security of an authentication protocol should not be relying on the secrecy of one single parameter. That would constitute a single-point of failure. Indeed, if the secret is revealed to the attacker, the security of the entire system may get compromised. Security protocol designers should study the dependency of the used secrete and the scenario where the secrets are revealed (Similar to **Lesson 2**).

8. Neglecting insider threats would result in the protocol becoming vulnerable to insider attacks, as well as, to CRP disclosure through physical invasion. Whatever

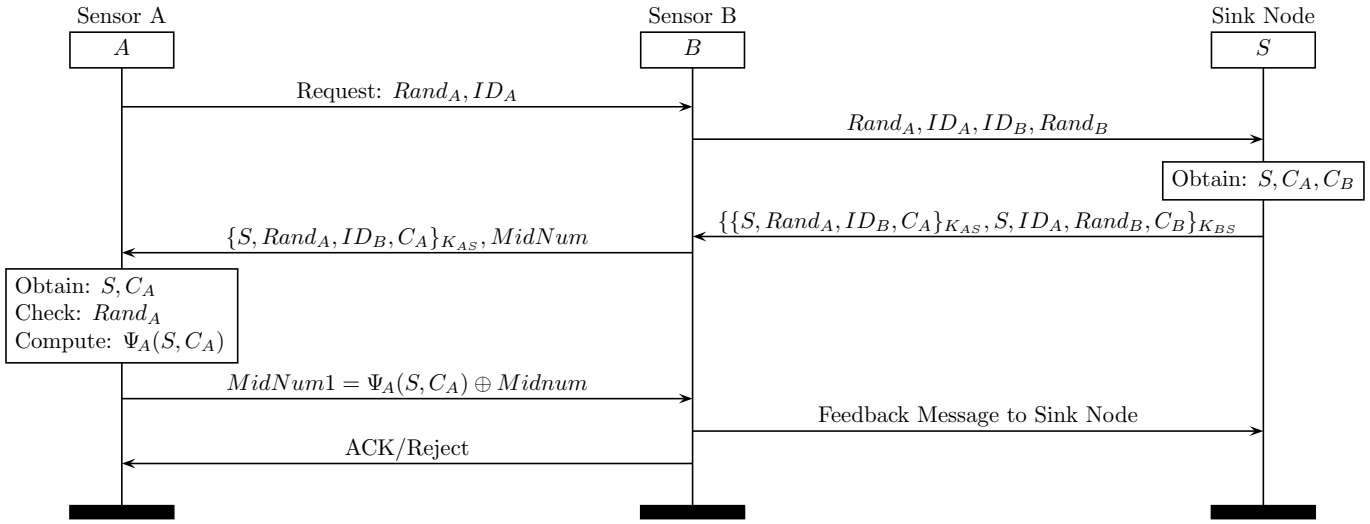

Fig. 4: Authentication Protocol for Outer Nodes by Tan al. [55].

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 1** and **5**).

E. BAN PUF-based Authentication protocol

In [55], Tan et al., proposed a PUF-based authentication protocol for wireless body area networks (WBAN) for single-hop hierarchical topologies. The protocol allows sensor nodes to be authenticated to a central sink node and with the help of a cloud-based trusted third party. Depending on the distance that separates the sensor nodes from the sink node, the protocol defines two authentication scenarios: the inlayer nodes authentication, where sensor nodes directly communicate with the sink node, and the outer nodes authentication, where some sensor nodes perform authentication with the sink node through intermediary neighbouring sensor nodes for energy-saving purposes. The protocol was implemented on a Xilinx Zynq-7000 zc706 board and its performance was evaluated. Also, the security of the protocol was briefly discussed. Notwithstanding, we have found the feasibility of the following attacks on the authentication protocol:

Spoofing Attacks. The inlayer authentication (i.e., sensor nodes authenticating to the sink node) consists of comparing the PUF response of a sensor node with the PUF response that is stored in the cloud and retrieved by the sink node. We observe that there is actually no real authentication. The protocol can be compromised by intercepting and tampering with messages during an authentication run. For example, during the authentication of a sensor node to a sink node, messages are exchanged in plaintext and the authentication information that is used to prove ones identity can easily be disclosed and reused (it is not protected). Technically, there is no authentication between the sink node and the cloud TTP server. The TTP just checks whether the ID of the sink is valid or not. In this case, and since the ID of the sink is not a secret information (as it is being broadcasted by the sink node), an attacker can impersonate the sink node and send fake authentication requests to the server. Furthermore, sensor nodes are subject to impersonation (spoofing attacks). Indeed, sensor nodes are required to send the xor of a challenge and its corresponding response to the sink node for the authentication. This xor result is similar to the value of C_A that the cloud server sends to the sink node in the beginning of the authentication. Hence, by intercepting the parameter C_A, an attacker can easily responds to the sink node by impersonating the concerned sensor node, i.e., sensor node A. Moreover, from an application perspective, sensors are just components of the body network that collect measures within the body environment and transfer the collected information to the outside word, i.e., cloud, through the sink node. Since the sink node has access to all CRPs of the enrolled sensor nodes, then if it acts as a malicious insider, it will be able to impersonate every single sensor node and upload fake and erroneous information onto the could. Last but not least, with respect to the outer sensor nodes authentication, which relies on proving the knowledge of a shared-key between sensor nodes, we observe that some of the messages (viz., Messages 5, 6, and 7 in Fig 4) are not authenticated, i.e., their source cannot be verified. Hence, spoofing attacks are possible in this authentication phase. For instance, Message 6 (feedback message to sink) is not authenticated, which allows an attacker to impersonate sensor node B and send a fake feedback message to the sink.

DoS Attack. As a consequence of each successful inlayer authentication, the server removes the used CRP from the database (as per the protocol). Hence, sending multiple fake authentication requests to the server would delete a large number of CRPs of a given sensor node (or all enrolled sensor nodes) and cause a denial of service to the system.

Another scenario to cause denial of service would be to send fake reject-messages (Message 7 in the outer sensor node authentication) to abort the authentication. This is possible as Message 7 cannot be authenticated.

CRP Disclosure. As a consequence of a valid request from the sink node to the cloud server, the server sends back a selection signal S, the xor of a challenge C and its corresponding response R, denoted as $C_A = C \oplus R$, and the response R. The authors claimed that by sending the xor of the challenge and response instead of sending the challenge in plaintext would prevent an attacker from directly learning the real challenge. However, because the corresponding response R is sent along with the xor result, i.e., C_A, an attacker will easily retrieve back the challenge C by xoring C_A with the response R. Also, during the outer node authentication phase, a sensor, let us say B, sends a parameter $MidNum$ (Message 4) to another sensor, let us say A, which uses it to obfuscate the PUF response $R_A = \Psi(C_A)$ using xor, i.e., sensor A computes $MidNum1 = \Psi_A(C_A)$. Sensor A sends the value of $MidNum1$ to sensor B. However, since the value of $MidNum$ was sent by sensor B encrypted, an attacker can use Message 4 and 5 to infer the value of the PUF's response. Additionally, the cloud server is considered a Trusted Third Party (TTP) and stores all CRPs in plaintext. If the server is compromised or acts as a malicious insider, the server can impersonate all enrolled sensor nodes, including the sink node.

Modeling Attack. The communication between the sensors and the sink node are not encrypted. This results in the challenges and their corresponding responses being transmitted in plaintext. An attacker may intercept a large number of CRPs to build a PFU model for the sensor node's PUF.

Lessons

9. When provers' CRPs are used only once (deleted after each authentication) and replaced by a new one following a successfully authentication, there is a risk of desynchronization that an attacker may cause if there is not limit in the number of failed authentication attempt. An attack may brute-force the authentication and succeed in replacing the next used CRP causing a DoS. In the other hand, the protocol designer should avoid setting up any harsh countermeasure that may be used against the system to cause a DoS.

10. Security protocol designer should carefully handle the use of the XOR operator. Although, this bitwise operator provides properties that allow the implementation of secure and lightweight cryptographic operation, it can however, destroy the security of the protocol if it is used incorrectly. In this protocol, the XOR operator was not used properly, which resulted in the possibility to generate the CRP disclosure attack.

F. IoT PUF-based Authentication Protocol

In [56], Satamraju et al., proposed a PUF-based mutual authentication protocol for IoT. The protocol allows IoT devices to securely connect to a trusted server to access some services over the cloud (e.g., upload device sensors' collected data). With respect to protocol implementation, Raspberry Pis model B+ were used as IoT devices and a Nexus A7 board (with Artix-7 FPGA) was used to implement a hybrid PUF circuit which consists of a combination of an arbiter PUF with a ring-oscillator PUF. The reliability and uniqueness of the PUF was evaluated within an ambient temperature. However, the security of the protocol was not discussed. We have found serious security issues with this protocol:

Spoofing Attack. The authors claimed that the protocol provides mutual authentication. We have analyzed the protocol and realized that the protocol does not provide any authentication at all. It merely aim to derive a shared key without any prior authentication (as per Fig. 6 in [56]). Indeed, the protocol starts by having the device sending its ID to the server, which uses the ID to randomly retrieve one CRP from a database and then sends back a challenge and the helper data for that CRP to the device. The server generates an ephemeral key pair and sends the public key to the device. The device computes the PUF response, using the challenge and the helper data, and then computes a shared secret by multiplying the PUF response with the received public key. The result is fed to a key derivation function to output a shared key. The server perform the same operations to compute the same shared key. At the end, neither the server nor the device proved to the other one its identity. Therefore, in this protocol, IoT devices are subject to impersonation.

DoS Attack. As there is no way for a device to prove its identity to the server, and attacker can impersonate a device and send fake nonsense data to the server, which would be decrypted to a random content and stored over the cloud.

CRP Disclosure. Last but not least, the protocol relies on a trusted authentication server that stores devices' CRPs in a secure database. Therefore, in the case where the servers acts as a malicious insider, it can disclose the CRPs of the enrolled devices. It can also impersonate the enrolled devices.

Lessons

11. Authentication cannot be established by having a device prove that it knows its registered ID that is being sent in plaintext and can be intercepted by attackers.

12. Since the trusted server stores all devices' CRPs, the protocol becomes 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.

G. Arbiter PUF-based Authentication Protocol

Zerrouki et al., [57] proposed a low-cost Arbiter PUF-based authentication protocol. The protocol allows a resource-constrained device to be authenticated to a server using its embedded PUF. It uses a fuzzy extractor to correct PUF responses from noises and provide higher PUF reliability. The protocol was simulated to demonstrate its good performance. However, the security of the protocol was not discussed. We have analyzed the security of the protocol and found some security design vulnerabilities. In the following paragraphs, we discuss some of feasible attacks:

Modeling Attacks. The protocol is vulnerable to machine learning attacks. Due to the fact that the protocol does not provide mutual authentication (only the devices proves their identity to the prover–"server"), an attacker would be able to impersonate the prover. In this case, the attacker would generate challenges and obtain plaintext responses (probing), as per the protocol, to constructs the device's CRPs. The current version of the protocol has a maximum of 65,536 possible CRPs since the challenges are expressed in 16 bits. After collecting a large number of CRPs, a PUF model of the device's PUF can be constructed.

Spoofing Attack. After collecting all 65,536 possible CRPs from the previous attack, the attacker impersonates the concerned device and realizes successful authentication attempts.

MITM Attack. As the protocol does not provide mutual authentication, an attacker can mount a middleman attack. The attacker impersonates the device and initiates an authentication attempt. When it receives the challenge, it forwards it to the device as the prover (server) to obtain the corresponding PUF response. The attacker then provides the prover with the received PUF response and passes the authentication.

DoS Attack. In this protocol, each CRP is used only once to prevent replay attacks. However, the current implementation of the protocol suffers from a design issue due to the fact of the absence of a CRP updating procedure. Indeed, for each enrolled device, the database stores a certain and finite number of CRPs (i.e., $\mathcal{H}(K_j)$ information), let us say $n \geq 1$ CRPs. Since after each successful authentication the used CRP is deleted from the database, there will be a maximum of n-possible authentications. After that, the device cannot be authenticated as there will be no CRP in the database for that device. This can be exploited by the attacker to cause a denial of service attack. Factually, if the attacker manage to build the corresponding PUF model, it can run multiple fake and successful authentication attempts and cause the server to delete the CRPs. After that, if the legitimate device requests an authentication to the server, it will be denied.

Lessons

13. The challenges and their corresponding responses should not be exchanged in plaintext. Otherwise, attackers may collect a large number of CRPs and build a precise model of the used PUF.

14. The size of the challenges and the responses should be large enough to widen the possibility interval. Otherwise, brute-force attacks would be possible. In this protocol, there are only 65,536 possible CRPs.

15. When an authentication protocol does not provide mutual authentication, protocol designers should verify whether middleman attacks are possible. The exchanged parameters should be used in a way where the involvement of a third party during an authentication can be detected by the legitimate parties.

16. Many PUF-based authentication protocols adopt the design of onetime CRP, where a given CRP at the verifier is deleted after successful authentication. A CRP updating procedure should be implemented to guarantee that there will always be some CRPs available to use. Otherwise, the CRPs will run out and the protocol stops (Similar to **Lesson 4**).

H. Secure PUF-based Authentication Protocol

Zheng et al., [58] proposed a secure mutual PUF-based authentication and key exchange protocol for IoT. The protocol allows resource-constrained devices to be authenticated using PUF technology and without having to locally store a large number of CRPs. It also allows the devices to derive a shared session key to establish a secure communication. The security of the protocol was evaluated using ProVerif and was proved to enforce secrecy and mutual authentication, and be resilient to replay and man-in-the-middle attacks. Unfortunately, we have found that the protocol is vulnerable to the following attacks:

CRP Disclosure. The server stores the CRPs of the devices in plaintext in a database. If the server is compromised, the CRPs can be disclosed. Furthermore, in the current design of the protocol, the shared session key that two authenticated devices establish at the end of an authentication session is known by the server (it is P_{2A} and P_{2B}). Hence, if the server is compromised (or acts maliciously), it can decrypt and read the encrypted and exchanged messages between two authenticated devices. The server may also inject fake and erroneous data to one of the devices and cause inconsistencies.

Message Corruption. The current design of the authentication protocol allows two resource-constrained devices to be authenticated to each other through a central server. Nevertheless, the communication between the devices and the server is not authenticated. The devices have no procedure to verify that they are communicating with the authentic server and not with a man-in-the-middle. Therefore, the devices cannot very whether the messages that they receive from the server are authentic and have not been tampered with during the

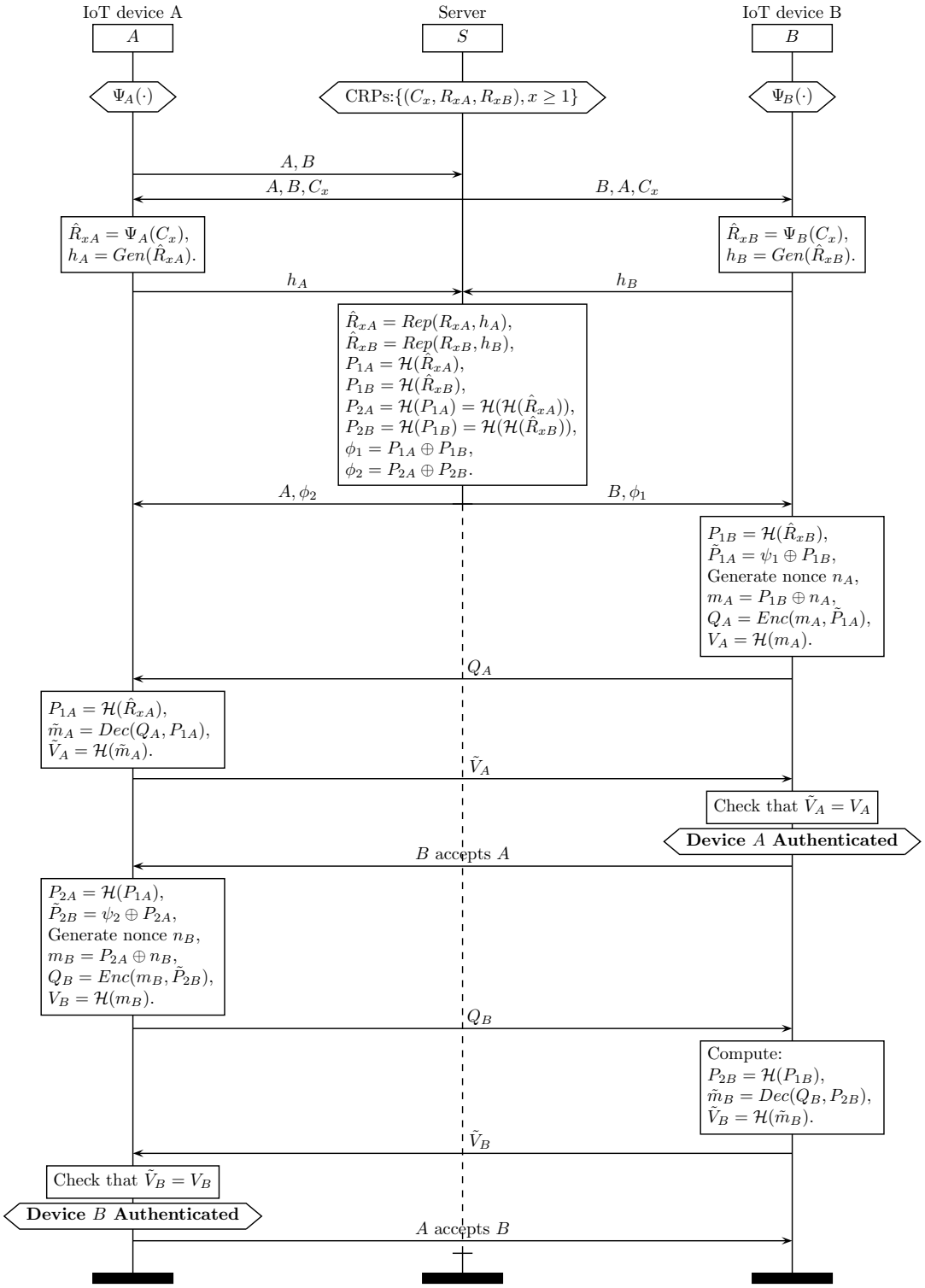

Fig. 5: Secure PUF-based Authentication Protocol by Zheng et al., [58].

transmission. This may cause a denial of service to the devices due to failure in the authentication.

Spoofing Attacks. As the server has access to the CRPs, then if the server acts as a malicious insider, the server would be able to impersonate every device that it enrolled in the system. Moreover, the challenge-response entry that was used during an authentication run is not removed from the server. This allows the entry to be reused again. A malicious device may successfully impersonate another device by taking

advantage of a previously conducted legitimate authentication and reusing the collected information to generate a spoofing attack when the same challenge is used again. For example, assuming that device X establishes a legitimate authentication with another device B (viz., MSC of Fig. 5, here $X = A$). At the end of the authentication, device X would have collected the parameters h_B, P_{1B}, P_{2B}, and n_X. The parameter h_B is intercepted when being sent to device B. The parameter P_{1B} is inferred from intercepting and xoring the parameter ϕ_1 with P_{1A}. The parameter P_{2B} is computed by xoring the received ϕ_2 with the locally computed parameter P_{2A}. Finally, the nonce n_X is inferred by xoring the computed m_X with the previously computed parameter P_{1B}. At this stage, the malicious device X can start a new authentication with another device, let us say C, on behalf of device B. This attack scenario is illustrated in the MSC of Fig. 6.

Lessons

17. 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.

18. It it important to consider insider threats when evaluating the security of an authentication protocol. We have seen in this protocol how a malicious insider took advantage of a legitimate authentication session and used the learned information to impersonate another device. Security protocol provers, e.g., AVISPA, Tamarin, and ProVerif, may not detect such attack scenario and hence the analysis has to be performed manually (Similar to **Lesson 1**).

IV. 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 [56]–[58]). 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 $\oplus$) 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 [55]. 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. 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.

Single-Point of Security. We have found out that the security of some PUF-based authentication protocols, e.g., [51] and [54], rely on the security of a long term secret. This constitutes a critical security flaw as the disclosure of the secret would reduce the security of the entire systems into void.

Countermeasure Inverse Consequences Flaw. Some authentication protocols (e.g., [52]) 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.

CRP-Updating Flaw. We have seen that certain PUF-based authentication protocols, e.g., [52], [53], [55] and [57], update the CRP at the verifier upon a successful authentication. This would constitute a security flaw if there is a possibility to guess the response and succeed in an authentication attempt. For example, if the number of failed authentication attempt is not limited and the size of the CRPs is not that large, then it is possible to brute-force the response and succeed in the authentication. This would allow an attacker to update the CRP at the verifier with an arbitrary value for future authentications. This however, will prevent the legitimate party from authenticating to the verifier as the CRP that is uses is not the same as the one stored at the verifier.

Ignoring Insider Threats. We have observed that a large number of the reviewed protocols do not consider insider

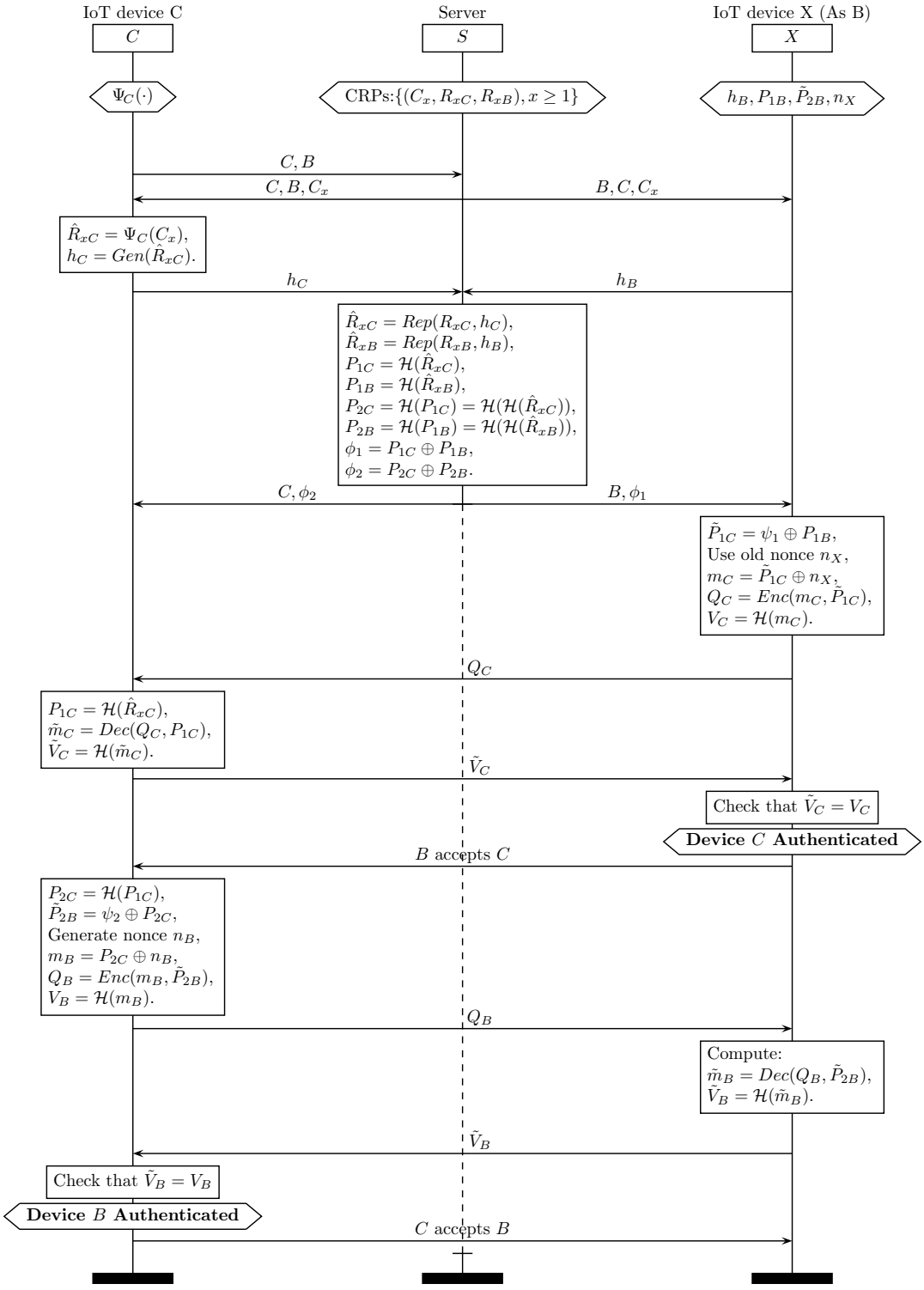

Fig. 6: Impersonation Attack on Authentication Protocol by Zheng et al., [58].

threats (e.g., [51], [53], [54] and [58]). 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 authentication protocol.

Last but not least, we point out that all reviewed protocols are vulnerable to connection deprivation through race-condition attacks [60]–[63]. 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 [61]–[63].

V. 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) single-point of security, (4) countermeasure inverse consequences flaw, (5) CRP-updating flaw (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 lessons that we learned in [1] to design and implement a PUF-based mutual authentication protocol (T2T-MAP) for Thing-to-Thing architectures [64]. The authentication protocol is secure against the reported attacks and is conform to IoT security and performance requirements.

References

[1] K. Lounis, and M. Zulkernine, "Lessons Learned: Analysis of PUF-based Authentication Protocols for IoT". Digital Threats: Research and Practice, ACM, DOI:https://doi.org/10.1145/3487060, 2021.

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

[3] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, "Silicon Physical Random Functions," Proceedings of the 9th ACM conference on Computer and communications security, ACM, pp. 148-160, 2002.

[4] 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.

[5] 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.

[6] S. R. Nassif. "Modeling and Forecasting of Manufacturing Variations," In Proceedings of ASP-DAC, 2001.

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

[8] H. Sun, M. 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. STh1N.4, Optical Society of America, 2017.

[9] U. Ruhrmair, F. Sehnke, 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.

[10] 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.

[11] Y. Dodis, R. 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.

[12] 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.

[13] G. E. Suh and S. Devadas. "Physical Unclonable Functions for Device Authentication and Secret Key Generation," Proceedings of the 44th annual Design Automation Conference. ACM, 2007.

[14] 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.

[15] 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.

[16] F. Tehranipoor, N. Karimian, K. Xiao, and J. A. Chandy, "DRAM-based Intrinsic Physical Unclonable Functions for System Level Security," in Proceedings of the 25th Edition on Great Lakes Symposium on VLSI, pp. 15-20, 2015.

[17] L. Lin, D. Holcomb, D. K. Krishnappa, P. Shabadi, W. Burleson. "Low-power Sub-Threshold Design of Secure Physical Unclonable Functions," In the 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design, pp. 43-48, 2010.

[18] 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.

[19] S. Vrijaldenhoven, "Acoustical physical uncloneable functions," M.S. thesis, Eindhoven University of Technology, 2004.

[20] 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.

[21] National-Research-Council. "Counterfeit Deterrent Features for the Next-Generation Currency Design,", The National Academies Press, Publication NMAB-472, pp. 1-144, https://doi.org/10.17226/2267, 1993.

[22] 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.

[23] G. Lenzini, S. Ouchani, P. Roenne, P. Y. A. Ryan, Y. Geng, J. Lagerwall, J. Noh. "Security in The Shell: An Optical Physical Unclonable Function Made of Shells of Cholesteric Liquid Crystals," in Proceedings of IEEE Workshop on Information Forensics and Security (WIFS), pp. 1-6, 2017.

[24] R. S. Indeck and M. W. Muller, "Method and Apparatus for Fingerprinting Magnetic Media," United States Patent US5365586A, 1994.

[25] NXP. "Secure Storage with SRAM PUF on NXP LPC54S0xx", AN12292, https : / / www.nxp.com / docs / en / application - note / AN12292.pdf, pp. 1-17, 2018.

[26] Microsemi. "Using SRAM-PUF System Service in SmartFusion2 - Libero SoC v11.7", Application Note AC434, pp. 1-19, 2016.

[27] 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-secure-high-end-fpga/#gs.0qyiip, 2015.

[28] 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.

[29] A. Maiti and P. Schaumont, "Improving the Quality of a Physical Unclonable Function using Configurable Ring Oscillators," in the 2009 International Conference on Field Programmable Logic and Applications, pp. 703-707, IEEE, 2009.

[30] S. Pandey, S. Deyati, A. Singh, and A. Chatterjee, "Noise-Resilient SRAM Physically Unclonable Function Design for Security," in IEEE 25th Asian Test Symposium (ATS). IEEE, pp. 55-60, 2016.

[31] D. Jeon, J. H. Baek, D. K. Kim, and B. D. Choi, "Towards Zero Bit Error-Rate Physical Unclonable Function: Mismatch-based vs. Physical-based Approaches in Standard CMOS Technology," in 2015 Euromicro Conference on Digital System Design. IEEE, pp. 407-414, 2015.

[32] K. H. Chuang, E. Bury, R. Degraeve, B. Kaczer, D. Linten, and I. Verbauwhede, "A Physically Unclonable Function using Soft Oxide Breakdown Featuring 0% 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] T. Esbach, W. Fumy, 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] M. A. Gora, A. Maiti, 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.

[44] 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.

[45] K. Müller, 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.

[46] J. Zhang and G. Qu. "Physical Unclonable Function-Based Key Sharing via Machine Learning for IoT security", in IEEE transactions on Industrial Electronics, vol. 67, no. 8, pp. 7025-7033, 2020.

[47] 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.

[48] O. Günlü, O. İşcan, 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.

[49] R. Maes, A. Van Herrewege, and I. Verbauwhede, "PUFKY: A Fully Functional PUF-based Cryptographic Key Generator," in Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems, pp. 302-319, Springer, 2012.

[50] 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.

[51] B. Narwal, A. Ojha, N. Goel, and S. Dhawan, "A Yoking-Proof based Remote Authentication Scheme for Cloud-aided Wearable Devices (YPACW)," in the 2020 IEEE International Conference for Innovation in Technology (INOCON), pp. 1-5, IEEE, 2020.

[52] B. K. Ghafi, and B. M-N. Maybodi, "A Distributed PUF-Based Mutual Authentication System with Self-Correction," in the 28th Iranian Conference on Electrical Engineering (ICEE), pp. 1-5, IEEE, 2020.

[53] S. Yoon, B. Kim, Y. Kang, and D. Choi, "PUF-based Authentication Scheme for IoT Devices," in the 2020 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1792-1794, IEEE, 2020.

[54] M. A. Qureshi, and A. Munir, "PUF-RAKE: A PUF-based Robust and Lightweight Authentication and Key Establishment Protocol," in Transactions on Dependable and Secure Computing, pp. 1-18, IEEE, 2021.

[55] X. Tan, J. Zhang, Y. Zhang, Z. Qin, Y. Ding, and X. Wang, "A PUF-Based and Cloud-Assisted Lightweight Authentication for Multi-Hop Body Area Network," in Tsinghua Science and Technology, vol. 26, no. 1, pp. 36-47, 2021.

[56] K. P. Satamarju, and B. Malarkodi, "A PUF-based Mutual Authentication Protocol for Internet of Things," in the 5th International Conference on Computing, Communication and Security (ICCCS), pp. 1-6, IEEE, 2020.

[57] F. Zerrouki, S. Ouchani, and H. Bouarfa, "A Low-Cost Authentication Protocol Using Arbiter-PUF" in the proceedings of the International Conference on Model and Data Engineering (MEDI 2021), LNCS 12732, pp. 101–116, Springer, 2021.

[58] Y. Zheng, and C-H. Chang, "Secure Mutual Authentication and Key-Exchange Protocol Between PUF-Embedded IoT Endpoints," in the

2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5, IEEE, 2021.

[59] S. Mauw and V. Bos. "Drawing Message Sequence Charts with LaTeX", in TUGBoat Journal, vol. 22, no. 1-2, pp. 87-92, 2001.

[60] K. Lounis and Mohammad Zulkernine. Bad-Token: Denial of Service Attacks on WPA3. In the Proceedings of the 12th International Conference on Security of Information and Networks, Article no. 15, pp. 1-8, ACM, 2019.

[61] K. Lounis and M. Zulkernine. "WPA3 Connection Deprivation Attacks," In the Proceedings of the 14th International Conference on Risks and Security of Internet and Systems, vol. 12026 of LNCS, pp. 164-176, Springer, 2019.

[62] K. Lounis and M. Zulkernine. "Exploiting Race-Condition for Wi-Fi Denial of Service Attacks," In the 13th International Conference on Security of Information and Networks, SIN'20, Istanbul, Turkey, November 4-7, 2020, pages 1-8, ACM. 2020.

[63] K. Lounis and M. Zulkernine. "Attacks and Defenses in Short-Range Wireless Technologies for IoT," in IEEE Access Journal, vol. 8, pp. 88892-88932, IEEE, 2020.

[64] K. Lounis. "Security of Wireless Short-Range Technologies and an Authentication Protocol for IoT," Ph.D. Thesis, School of Computing, Kingston, Ontario, Queen's University, 2021.