Journal of Parallel and Distributed Computing 169 (2022) 199–210

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Challenging the security of "A PUF-based hardware mutual authentication protocol"

Morteza Adeli [a], Nasour Bagheri [b,c,*], Honorio Martín [d], Pedro Peris-Lopez [e]

[a] Department of Science, Shahid Rajaee Teacher Training University, 16788-15811, Tehran, Iran
[b] CPS^2 Laboratory, Electrical Engineering Department, Shahid Rajaee Teacher Training University, 16788-15811, Tehran, Iran
[c] School of Computer Science (SCS), Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
[d] Department of Electronic Technology, University Carlos III of Madrid, 28911, Leganés, Madrid, Spain
[e] Computer Science Department, University Carlos III of Madrid, 28911, Leganés, Madrid, Spain

ARTICLE INFO

Article history:
Received 27 May 2021
Received in revised form 19 April 2022
Accepted 30 June 2022
Available online 14 July 2022

Keywords:
IoT
PHEMAP
Authentication
PUF
Security analysis

ABSTRACT

Recently, using Physical Unclonable Functions (PUF) to design lightweight authentication protocols for constrained environments such as the Internet of Things (IoT) has received much attention. In this direction, Barbareschi et al. recently proposed PHEMAP in *Journal of Parallel and Distributed Computing*, a PUF based mutual authentication protocol. Also, they extended it to the later designed Salted PHEMAP, for low-cost cloud-edge (CE) IoT devices.

This paper presents the first third-party security analysis of PHEMAP and Salted PHEMAP to the best of our knowledge. Despite the designer's claim, we show that these protocols are vulnerable to impersonation, de-synchronization, and traceability attacks. The success probability of the proposed attacks is '1', while the complexity is negligible. In addition, we introduce two enhanced lightweight authentication protocols based on PUF chains (called PBAP and Salted PBAP), using the same design principles as PHEMAP and Salted PHEMAP. With the performance evaluation and the security analysis, it is justified that the two proposed schemes are practically well suited for use in resource-constrained IoT environments.

1. Introduction

Internet of Things (IoT) is proliferating nowadays, and researchers are studying and developing different aspects of IoT applications. For example, IoT could be used in smart homes, where IoT devices such as sensors or actuators control temperature, light, and house security to improve the quality of life. IoT architecture has three layers: the device layer, the gateway layer, and the server or cloud data center layer. The device layer includes various IoT devices ranging from very constrained devices (e.g. RFID passive tags) to smartphones with some computing capabilities to devices with high processing capacities (e.g. general-purpose computers). Among them, RFIDs have fundamental importance in IoT, thanks to their cost-efficiency. An RFID can work in various environments without significant artificial interference, with low energy consumption, to detect, store and send information through wireless channels.

Generally, in RFID systems, a unique identity (ID) is assigned to each tag to find and recognize a specific device. The reader must authenticate an RFID tag before starting communication to ensure that the data being exchanged is protected. So far, many authentication protocols for different applications and environments have been proposed in the literature. Since the RFID tags usually have restricted computation power and storage size, they support only simple operations such as exclusive OR (XOR), pseudorandom number generator (PRNG), shift operation, etc.

Today, using physical unclonable functions (PUFs) in authentication protocols has been studied by many researchers and has been successfully adopted to achieve authentication and identification in the resource-constrained embedded devices [2,13,20–22,41]. A PUF works as a digital fingerprint and serves as a unique identity for a device [37]. When a device (like FPGA) is fabricated in the manufactory, a PUF entity is embodied in the physical structure. This primitive is unique and infeasible to duplicate or predict. In detail, an ideal PUF is expected to operate as a one-way function to be used in the authentication protocols based on challenge-response pairs.

* Corresponding author.
E-mail addresses: M.adeli@sru.ac.ir (M. Adeli), Nbagheri@sru.ac.ir, Nbagheri@srttu.edu (N. Bagheri), hmartin@ing.uc3m.es (H. Martín), pperis@inf.uc3m.es (P. Peris-Lopez).

https://doi.org/10.1016/j.jpdc.2022.06.018
0743-7315/

PUF architectures for silicon devices are mainly classified in two classes [30]: (1) delay-based approaches such as the arbiter PUF, the ring oscillator (RO) PUF, and the Anderson PUF, that use differences in paths delays within the specific circuit; and (2) memory-based solutions such as the SRAM PUF, butterfly PUF, sense amplifier PUF, flip-flop PUF, that exploit the mismatches of the memory cells to generate a response to a challenge.

In this paper, we do not focus on how to design an efficient PUF. Therefore we suppose that the PUFs used in the authentication schemes have good behavior, for example, enough stability and unpredictability. It is worth noting, to study the security of cryptographic constructions, it is common to suppose that the crypto primitives, e.g. pseudorandom functions (PRFs) or pseudorandom generators (PRGs), are all secure, efficient, and scalable. Concerning the security evaluation of a PUF-based authentication protocol, we suppose that the PUF function is secure and reliable. Although no one has ever built an ideal PUF, intensive research is being done to build PUFs with better properties, such as good entropy and small or even zero-bit error rates. The scientific community has proposed several PUF based authentication protocols for IoT systems in recent years, e.g., [3,11,24,25,39], but few of them are suited to use in IoT systems due to their security weaknesses [28,31,39].

1.1. Related works

Majzoobi et al. [31] introduced a Slender PUF protocol and claimed to be efficient and secure. However, later analysis demonstrated its vulnerabilities, such as the lack of privacy [5,18]. Aysu et al. [5] presented an efficiently PUF based mutual authentication scheme between a server and a resource-constrained device. Their report showed how the proposed scheme could be implemented efficiently on a resource-constrained platform such as SASEBO-GII board.

Kulseng et al. [28] proposed a mutual authentication and ownership transfer protocol based on PUF and Linear Feedback Shift Registers (LFSR). They claimed that their scheme can be implemented efficiently on hardware and is resistant to various attacks. However, as discussed in [39], Xu et al. showed that their claim is wrong and presented a de-synchronization attack on it. To address this vulnerability, they proposed a lightweight authentication protocol based on PUF functions and claimed that their proposed scheme could withstand various attacks. But Bendavid et al. [8] analyzed the Xu et al. scheme [39] and showed that it's vulnerable to de-synchronization and secret disclosure attacks.

Braeken [11] showed that the PUF based key agreement scheme, presented by Chatterjee et al. [12] is vulnerable to impersonation, replay, and man-in-the-middle attacks. She fixed these vulnerabilities and proposed a new efficient key agreement scheme based on PUF. Recently, Ameri et al. [3] proposed two PUF based authentication schemes for high-resource and low-resource devices and proved that their schemes could resist various known attacks. Gope et al. [20] proposed a PUF-based mutual authentication protocol for real-time data access in Industrial Wireless Sensor Networks (IWSN). Zhang et al. [42] connected PUF and blockchain to develop a privacy-aware PUFs-based multi-server authentication protocol in cloud-edge IoT systems. The main target of this protocol is to overcome the information leakage due to the explicit storing of the challenge-response pairs (CRPs) of PUFs generated by devices by each edge-server. In another recent work, and almost targeting the same problem, Chen et al. [14] proposed Shamir's secret sharing to solve the problem of storing the CRPs in the server-side in the proposed PUF-based authentication protocol.

One of the challenges to using most of the above PUF based authentication protocols is the vast number of challenge-response pairs of PUF needed to be stored by the authenticator and the devices embedding the PUF. On the other hand, many IoT devices are typically resource-constrained in real applications and cannot employ traditional PUF based authentication schemes. Therefore, most of the existing PUF based mutual authentication protocols are impractical and only able to verify the identity of the devices. One solution to reduce the number of challenge-response pairs is to construct sequences (chains) of challenge-response pairs by a recursive invocation of the PUF embedded on the devices. In this direction, recently, Barbareschi et al. [6,7] proposed two mutual authentication protocols (called PHEMAP and Salted PHEMAP schemes) for low-cost hardware devices which use PUF chains in their authentication procedures. The Salted PHEMAP has been specially designed for cloud-edge (CE) IoT systems. They analyzed their schemes through formal and informal security proof and claimed that their schemes are secure against various known attacks. In this paper, we analyze, in more detail, the security of these protocols and provide the first third-party security analysis of them to the best of our knowledge.

1.2. Our contribution

The main contribution of this paper contains two folds:

- First, we analyze the PUF based authentication protocol called PHEMAP, proposed by Barbareschi et al. [6] and show that this scheme is vulnerable to impersonate attack. Furthermore, we demonstrate that the PHEMAP scheme is traceable. In the following, we analyze the Salted PHEMAP protocol proposed by Barbareschi et al. [7] for Cloud-Edge (CE) IoT systems. This scheme is also vulnerable to impersonation, de-synchronization, and traceability attacks.
- Second, to address this weakness, first we propose a basic PUF based mutual authentication protocol (PBAP), and then we extend this idea and propose a Salted mutual authentication scheme (Salted PBAP) suited for CE systems. We analyze the security of the two proposed schemes through formal and informal methods and prove that these schemes do not have vulnerabilities of the two last proposed schemes (noted in this paper) and resist known attacks.

1.3. Organization

The remainder of this paper is organized as follows: in section 2 the required preliminaries are presented, including a brief description of the PHEMAP scheme [6] and Salted PHEMAP scheme [7], that has been designed for cloud-edge (CE) IoT systems. In section 3 we explain how to perform impersonation and traceability attacks on the PHEMAP scheme, and also, we will explain the weakness of the Salted PHEMAP scheme. To address these vulnerabilities, we proposed two improved lightweight authentication schemes based on PUF chains called PBAP in section 4 and Salted PBAP in section 5 that are resistant against various known attacks. Next, we analyze our two proposed schemes through formal and informal proof in section 6. The performance efficiency of our two proposed schemes has been discussed in section 7. In the end, the conclusion of the paper is described in section 8.

2. Preliminaries

Through the paper, we are using the notation represented in Table 1.

2.1. PHEMAP scheme

In this section, we give a brief description of the PHEMAP scheme [6]. This scheme uses only PUF and the bitwise exclusive

Table 1
Notation used in this paper.

Notation	Description
$\gamma_{D,c_0,M}$	a PUF chain of device D with root chain c_0 and length M
$\theta_D(.)$	physical unclonable function
l_i	i-*th* link in chain γ
n	nonce generated in the verifier
r	random number generated in the tag
S	sentinel period
AS	an authentication service
G	a gateway
D	a RFID device

OR operator to encrypt and decrypt messages transferred between a verifier and a tag. The proposed scheme contains three phases as follows: (1) enrollment, (2) initialization, and (3) verification. Before we describe the PHEMAP scheme, we need some definitions that are taken from [6].

Definition 2.1. Let $\theta_D(.)$ be PUF embedded in device D. The PUF chain $\gamma_{D,c_0,M}$ with root chain c_0 and length M is defined as:

$$\{c_0, \theta_D(c_0), \theta_D^2(c_0), ..., \theta_D^{M-1}(c_0)\} \tag{1}$$

where $\theta_D^i(.) = \overbrace{\theta_D(\theta_D(...))}^{i}$ and all of $\theta_D^i(.)$ are distinct. We referred each $\theta_D^i(c_0)$ to as *links* and noted by l_i hereafter.

Definition 2.2. Let $\gamma_{D,c_0,M}$ be a chain, σ_0 be a link on it and S be a positive integer. We refer to as chain **sentinels** all multiple of S, starting from link σ_0.

2.1.1. Enrollment

In this phase, the verifier generates T distinct chains $\gamma_{D,c_0,M}$ where each root chain c_0 is selected randomly, so each link appears only once over the extracted chains. The length of each chain (M) is different and depends on the number of the new distinct links that can be generated by iterating the PUF, starting from the random root chain c_0. All of the T generated chains are stored only in the verifier, and the devices store the last synchronized link, which is used to compute the last exchanged message.

The number of generated chains depends on the storage capacity of the verifier and the number of devices that can be managed by the verifier. In the end, the *sentinel* period S, is defined and embedded in both the devices and the verifier.

2.1.2. Initialization

The initialization contains four phases as following:

1. The verifier generates a random nonce n and sends

$$m_1 = \{l_i, (\oplus_{j=0}^{S-3} l_{i+j+1}) \oplus n, l_{i+S-1} \oplus n\} = \{l_i, v_1, v_2\} \tag{2}$$

to the device D.
2. Upon receiving the message m_1, the device D checks

$$\oplus_{j=0}^{S-2} \theta_D^{j+1}(l_i) \stackrel{?}{=} v_1 \oplus v_2 \tag{3}$$

If it holds true, the device D generates a random nonce r and computes

$$m_2 = \{\theta_D^S(l_i) \oplus r, \theta_D^{S+1}(l_i) \oplus r\} = \{d_1, d_2\} \tag{4}$$

and sends it to the verifier. Moreover, the device D saves d_2 in its secure register.

3. The verifier computes and checks

$$l_{i+S} \oplus l_{i+S+1} \stackrel{?}{=} d_1 \oplus d_2 \tag{5}$$

If it is true, the verifier authenticates the device D and sends

$$m_3 = \{l_i, l_{i+S+2} \oplus r\} = \{l_i, v_3\} \tag{6}$$

to D.
4. The device D computes and checks

$$\theta_D^{S+1}(l_i) \oplus \theta_D^{S+2}(l_i) \stackrel{?}{=} v_3 \oplus d_2 \tag{7}$$

If it holds true, the device D authenticates the verifier and saves $\theta_D^{S+2}(l_i)$ in its register.

2.1.3. Verification

Both the verifier and the device D can initiate this phase. Suppose that the verifier is initiator of the protocol. The verifier knows both l_i the last synchronized link and σ_0 the first sentinel link after initialization. If $l_{i+1} \neq \sigma_0$, the verifier sends l_{i+1} to the device D, else it sends l_{i+2} to it. The device D has the last synchronized link l_i in its register and knows the sentinel link σ_0 based on the current value of its counter. So it computes $l'_{i+1} = \theta_D(l_i)$ and checks $l_{i+1} \stackrel{?}{=} l'_{i+1}$ (or if $l_{i+1} = \sigma_0$, it checks $l_{i+2} \stackrel{?}{=} l'_{i+2}$). If it holds true, the device D authenticates the verifier and saved l_{i+1} (or l_{i+2}) in its register. On the verifier side, by using the same method as for the device D, the verifier authenticates the device D.

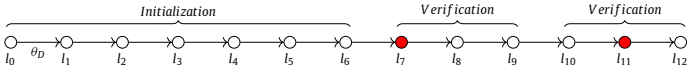
In Fig. 1, the PHEMAP scheme has been illustrated by an example. In this example, the first link is l_0 and the sentinel links are l_7, l_{11} and $S = 4$.

2.2. Salted PHEMAP scheme

In this section, we give a brief description of the Salted PHEMAP scheme, proposed by Barbareschi et al. [7] to be implemented in Cloud-Edge (CE) based IoT systems. CE-based IoT systems typically consist of three architectural layers, see Fig. 2: i) cloud service as top layer ii) gateway nodes as middle layer iii) terminal nodes or edge IoT devices as the lower layer.

The Salted PHEMAP scheme provides mutual authentication between a terminal node and the respective gateway in a CE-based IoT system. In Salted PHEMAP, a gateway acts as a local verifier for the underlying terminal nodes by using part of the enrolled PUF chains that are transferred from the authentication service (AS). Note that the Salted PHEMAP scheme is always performed after the basic PHEMAP scheme and also suppose that the terminal device D and AS are synchronized on link l_{i-1} of chain $\gamma_{D,l_0,M}$. The setup phase of the Salted PHEMAP scheme includes the following steps:

1. Device D sends a request message $m_0 = \{\theta_D(l_{i-1})\}$ to AS and also saves l_i in its local memory.
2. AS verifies that the received message m_0 is equal to immediately following link l_i in the current chain $\gamma_{D,l_0,M}$. If the two values match, AS authenticates D and extracts a carnet $\tau_{D,t_0,T} = \{t_0, t_1, ..., t_{T-1}\}$ from chain $\gamma_{D,l_0,M}$. Next AS generates random salt r_s for device D and computes message $m_1 = \{v_1, v_2\} = \{\theta_D(l_1), \theta_D^2(l_1) \oplus r_s\}$ and sends it to the device D.
3. Device D computes $l_2 = \theta_D(l_1)$ and compares it with v_1. If the two values match, device D authenticates the AS and extracts the salt r_s from v_2. Afterward, it computes $l_3 = \theta_D(l_2)$ and sends message $m_2 = \{l_3\}$ to the AS. It also saves $\theta_D(l_3) \oplus r_s$ in its local memory to communicate with the respective gateway G.



Phase	Verifier		Device
Init.	$v_1 = l_1 \oplus l_2 \oplus n$ $v_2 = l_3 \oplus n$	$\xrightarrow{m_1=\{l_0, v_1, v_2\}}$	
			$\theta_D(l_0) \oplus \theta_D^2(l_0) \oplus \theta_D^3(l_0) \stackrel{?}{=} v_1 \oplus v_2$
		$\xleftarrow{m_2=\{d_1, d_2\}}$	$d_1 = \theta_D^4(l_0) \oplus r$ $d_2 = \theta_D^5(l_0) \oplus r$ $store\ Q = d_2$
	$l_4 \oplus l_5 \stackrel{?}{=} d_1 \oplus d_2$ $v_3 = l_6 \oplus r$	$\xrightarrow{m_3=\{l_0, v_3\}}$	
			$\theta_D^5(l_0) \oplus \theta_D^6(l_0) \stackrel{?}{=} v_3 \oplus d_2$ $store\ Q = \theta_D^6(l_0)$
Verif.	$l'_0 = l_8$ $l'_1 = l_9$	$\xrightarrow{l_8}$	
		$\xleftarrow{\theta_D(l_8)}$	$\theta_D^2(Q) \stackrel{?}{=} l_8$ $store\ Q = \theta_D(l_8)$
	$l'_1 \stackrel{?}{=} \theta_D(l_8)$		

Fig. 1. PHEMAP protocol, where Init. and Verif. denote initialization and verification retrospectively.

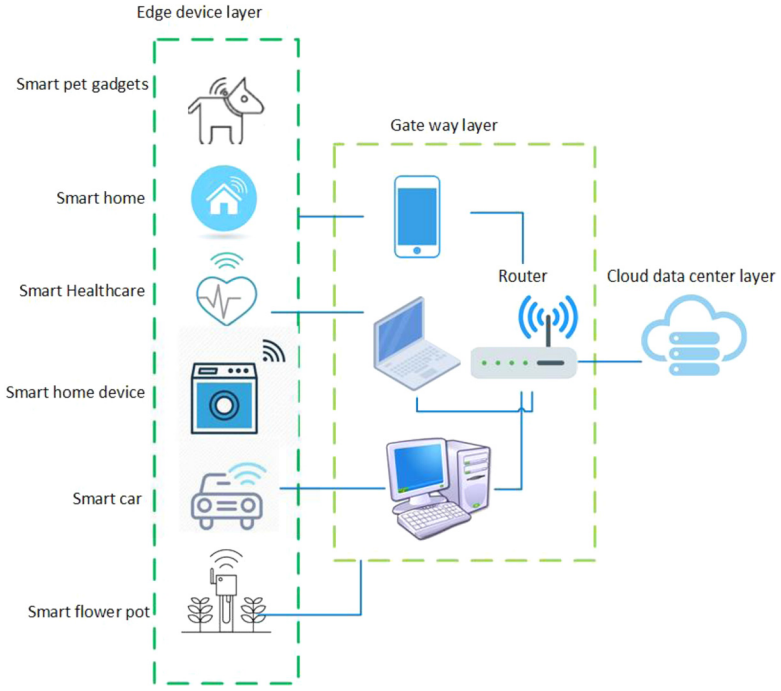


Fig. 2. CE-based architecture.

4. *AS* checks $\theta_D(l_2) \stackrel{?}{=} m_2$, if the two values equal, then *AS* computes a salted carnet $\chi_{D,x_0,T} = \{x_0, x_1, ..., x_{T-1}\} = \{t_0 \oplus r_s, t_1 \oplus r_s, ..., t_{T-1} \oplus r_s\}$ and sends message $m_3 = \{\chi_{D,x_0,T}\}$ to gateway through a secure channel.

Now, the device D and the respective gateway G use the salted carnet $\{\chi_{D,x_0,T}\}$ for subsequent authentication operation between themselves. Note that, after setup, the device D and the gateway G carry out the same interactions as the basic PHEMAP verification phase discussed in section 2.1. We illustrate the Salted PHEMAP scheme by an example in Fig. 3. In this example, suppose that l_0 is the synchronized link between *AS* and *D*.

3. Security analysis of PHEMAP and Salted-PHEMAP protocols

3.1. Security challenges of the PHEMAP protocol

3.1.1. *Impersonate attack*

In the initialization phase of the PHEMAP protocol, the attacker eavesdrops and records the two valid messages $\{m_1, m_2\}$ and intercepts the message $\{m_3\}$ between the verifier and the device:

- The query message of the verifier $m_1 = \{l_0, v_1, v_2\}$
- The response message of the device $m_2 = \{d_1, d_2\}$
- The response message of the verifier $m_3 = \{l_0, v_3\}$

Phase	Authentication Service	Gateway	Device
Init.		$\xleftarrow{m_0=\{l_1\}}$	*compute* $l_1 = \theta_D(l_0)$
	$l_1 \stackrel{?}{=} m_0$ $\tau_{D,t_0,T} = \{t_0, t_1, ..., t_{T-1}\}$ r_s: *random salt* $v_1 = l_2$ $v_2 = l_3 \oplus r_s$		
		$\xrightarrow{m_1=\{v_1,v_2\}}$	
			$\theta_D(l_1) \stackrel{?}{=} v_1$ $r_s = \theta_D^2(l_1) \oplus v_2$ *store* $\theta_D^4(l_1) \oplus r_s = l_5 \oplus r_s$
		$\xleftarrow{m_2=\{\theta_D^3(l_1)\}}$	
	$l_4 \stackrel{?}{=} \theta_D^3(l_1)$ $\chi_{D,t_0,T} = \{x_i = t_i \oplus r_s, t_i \in \tau_{D,t_0,T}\}$		
	$\xrightarrow[\text{a secure channel}]{m_3=\{\chi_{D,t_0,T}\}\text{ to gateway through}}$	m_3	
Verif.		$L_1 = x_1$ $L_2 = x_2$	
		$\xrightarrow{\{L_1\}}$	
			$\theta_D(x_0) \stackrel{?}{=} L_1$ *store* $Q = \theta_D^2(x_0)$
			$\xleftarrow{\{\theta_D^2(x_0)\}}$
		$L_2 \stackrel{?}{=} \theta_D^2(x_0)$	

Fig. 3. Salted PHEMAP protocol, where Init. and Verif. denote initialization and verification retrospectively.

Afterward, the attacker repeats the message $\{m_1\}$ and intercepts the response message $m_2' = \{d_1', d_2'\}$ of the device D. The attacker computes

- $\triangle r = d_1 \oplus d_1' = r \oplus r'$.
- $v_3' = v_3 \oplus \triangle r = l_6 \oplus r'$

and sends $m_3' = \{l_0, v_3'\}$ to the device. Upon receiving the message m_3', the device authenticates the attacker as a legitimate verifier.

3.1.2. *Traceability*

In the initialization phase, the attacker eavesdrops on the query message m_1, which is sent by the verifier and the response message $m_2 = \{d_1, d_2\}$ from the device. We know that the XOR of d_1 and d_2 is a constant value, because

$$d_1 \oplus d_2 = (\theta_D^4(l_0) \oplus r) \oplus (\theta_D^5(l_0) \oplus r) = (\theta_D^4(l_0) \oplus \theta_D^5(l_0)) \tag{8}$$

Therefore an attacker can trace a particular device D by sending a fixed query message m_1 to the victim and XORing two values d_1 and d_2 of its response message m_2.

3.2. Security challenge of the Salted PHEMAP protocol

3.2.1. *De-synchronization attack*

In de-synchronization attack on Salted PHEMAP, the attacker breaks the synchronization between a legitimate device and its respective gateway. In the setup phase, the attacker intercepts message $m_1 = \{v_1, v_2\} = \{\theta_D(l_i), \theta_D^2(l_i) \oplus r_s\}$ and modifies it to $m_1' = \{\theta_D(l_i), \theta_D^2(l_i) \oplus r_s \oplus \triangle\}$ and sends it to the device D. The device D can not verify the integrity of the message m_1', so it computes $r = \theta_D^2(l_i) \oplus v_2 \oplus \triangle$. Therefore the device D, unlike the gateway G which is synchronized on the link $\theta_D^4(l_i) \oplus r_s$, is synchronized on the link $\theta_D^4(l_i) \oplus r_s \oplus \triangle$. Hence, the verification process between the device D and the gateway G is failed.

3.2.2. *Impersonation attack*

Suppose that 1) the basic PHEMAP and subsequently, the Salted PHEMAP have been performed and 2) the transferred messages between a device D and the authentication service AS have been eavesdropped and recorded by an attacker. Let the initial link of the PUF chain be l_0. If the attacker sends the query message $\{l_0, v_1, v_2\}$ to the device D, it moves to the initialization phase of the basic PHEMAP authentication scheme. Therefore according to subsection 3.1, the attacker impersonates as a legitimate authentication service AS.

3.2.3. *Traceability attack*

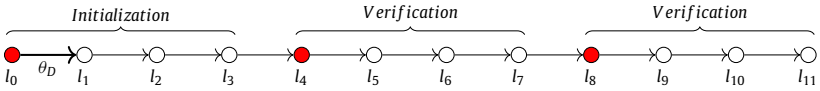
In the setup phase, three links l_1, l_2, l_4 are fixed values. So if an attacker eavesdrops the message m_0, he/she can trace a particular device D by replying the message m_0 and receiving the response messages m_2, m_3.

4. PBAP: a PUF-based authentication protocol

This section proposes a PUF based authentication protocol called PBAP that uses the recursive sequences (chains) of the challenge-response pairs (CRPs) of the PUF embedded on the devices. In the PBAP scheme, chains are stored in the verifier, which has no storage limit, and the devices need to store only the last PUF response used in the authentication procedure. The PBAP has no vulnerabilities of the past proposed scheme and can resist various known attacks. The PBAP scheme, see also Fig. 4 has three phases as follows:

4.1. Enrollment

In this phase, the verifier generates and stores a set of PUF chains for each device in its secure data register. These PUF chains are generated by iterating a PUF function $\theta_D(.)$, which is embedded in a hardware device (see Definition 2.1). This phase is performed in secure environment.



Phase	Verifier	Device
Init.	$m_1 = \theta_D(l_0) \oplus n$; $\xrightarrow{\{l_0, m_1\}}$	
		$n = m_1 \oplus \theta_D(l_0)$; r: *random number*; k_1: *Hamming weight of* r; k_2: *Hamming weight of* $r \wedge n$; $m_2 = \theta_D(l_0) \oplus r$; $m_3 = (((\theta_D^2(l_0) \lll k_1) \wedge (r \ggg k_2))$ $\vee(\theta_D(l_0) \lll k_2)) \oplus (n \lll k_1)$; store $l_2 = \theta_D^2(l_0)$; $\xleftarrow{\{m_2, m_3\}}$
	$r = m_2 \oplus \theta_D(l_0)$; $n \stackrel{?}{=} (m_3 \oplus ((((\theta_D^2(l_0) \lll k_1) \wedge (r \ggg k_2)))$ $\vee(\theta_D(l_0) \lll k_2)) \ggg k_1)$; $m_4 = ((\theta_D^3(l_0) \ggg k_1) \wedge (r \lll k_1))$ $\vee(\theta_D^2(l_0) \ggg k_2)$; *update* $l'_0 = l_3 = \theta_D^3(l_0)$; $\xrightarrow{\{m_4\}}$	
		$m'_4 = ((\theta_D(l_2) \ggg k_1) \wedge (r \lll k_1))$ $\vee(l_2 \ggg k_2)$; $m'_4 \stackrel{?}{=} m_4$; *store* $l'_0 = \theta_D(l_2)$;
Verif.	$m_5 = \theta_D(l'_0) \oplus \theta_D^2(l'_0)$; $0 \leq counter \leq k$; *if no response is received, counter++* *if counter=k, move to Init. phase* $\xrightarrow{\{m_5\}}$	
		$\theta_D(l'_0) \oplus \theta_D^2(l'_0) \stackrel{?}{=} m_5$; $m_6 = \theta_D^3(l'_0) \oplus \theta_D^4(l'_0)$; *store* $l''_0 = l_7 = \theta_D^4(l'_0)$; $\xleftarrow{\{m_6\}}$
	$\theta_D^3(l'_0) \oplus \theta_D^4(l'_0) \stackrel{?}{=} m_6$; *update* $l''_0 = l_7 = \theta_D^4(l'_0)$;	

Fig. 4. The PBAP protocol, where Init. and Verif. denote initialization and verification retrospectively.

4.2. Initialization

In initialization phase

1. The verifier selects a PUF chain with the initial link l_i and generates a random nonce n. Afterward, it sends the message $\{l_i, m_1 = \theta_D(l_i) \oplus n\}$ to the device D.
2. Upon receiving the message m_1, the device D generates random number r and computes
   - $n = m_1 \oplus \theta_D(l_i)$
   - k_1=The Hamming weight of r
   - k_2=The Hamming weight of $r \wedge n$
   - $m_2 = \theta_D(l_i) \oplus r$
   - $m_3 = (((\theta_D^2(l_i) \lll k_1) \wedge (r \ggg k_2)) \vee (\theta_D(l_i) \lll k_2)) \oplus (n \lll k_1)$
   - $Store\ l_{i+2} = \theta_D^2(l_i)$

   and sends $\{m_2, m_3\}$ to the verifier.
3. The verifier computes r and n' and compares the two values n and n'
   - $r = m_2 \oplus \theta_D(l_i)$
   - $n' = (m_3 \oplus ((((\theta_D^2(l_i) \lll k_1) \wedge (r \ggg k_2))) \vee (\theta_D(l_i) \lll k_2)) \ggg k_1)$
   - $n' \stackrel{?}{=} n$

   If two values match, the verifier authenticates the device D and computes
   - $m_4 = ((\theta_D^3(l_i) \ggg k_1) \wedge (r \lll k_1)) \vee (\theta_D^2(l_i) \ggg k_2)$
   - $Update\ l'_i = l_{i+3} = \theta_D^3(l_i)$

   Then, it sends the message m_4 to the device D.
4. The device D computes m'_4 and compares it with m_4
   - $m'_4 = ((\theta_D(l_{i+2}) \ggg k_1) \wedge (r \lll k_1)) \vee (l_{i+2} \ggg k_2)$
   - $m'_4 \stackrel{?}{=} m_4$
   - $Store\ l_{i+3} = \theta_D^3(l_i)$

   If the comparison succeeds, the device D authenticates the verifier and the initialization phase is finished successfully.

4.3. Verification

Both the verifier and the device can initiate the verification phase. Just after the initialization phase, the verifier and the device store in their secure register, the same last link that has been used in previous exchanges (let us assume it is l_{i+3}). Suppose that the verifier initiates the verification phase. This phase contains three steps as follows:

Phase	Authentication Service	Gateway	Device
Init.		$\xleftarrow{m_0=\{l_1\}}$	compute $l_1 = \theta_D(l_0)$;
	$l_1 \stackrel{?}{=} m_0$;		
	$\tau_{D,t_0,T} = \{t_0 = l_5, t_1, ..., t_{T-1}\}$;		
	r_s: *random salt*;		
	k: *Hamming weight of* r_s;		
	$v_1 = l_2 \oplus r_s$;		
	$v_2 = (l_3 \lll k) \wedge (r_s \ggg k)$;		
		$\xrightarrow{m_1=\{v_1,v_2\}}$	
			$r_s = \theta_D(l_1) \oplus v_1$;
			$v_2 \stackrel{?}{=} (\theta_D^2(l_1) \lll k) \wedge (r_s \ggg k)$;
			$v_3 = (\theta_D^3(l_1) \ggg k) \wedge (r_s \lll k)$;
			store $\theta_D^4(l_1) = l_5$ *and* r_s;
		$\xleftarrow{m_2=\{v_3\}}$	
	$(l_4 \ggg k) \wedge (r_s \lll k) \stackrel{?}{=} v_3$;		
	$\tau_{D,t_0,T} = \begin{cases} \{x_i = t_i \oplus r_s, \\ t_i \in \tau_{D,t_0,T}\}; \end{cases}$		
	$\xrightarrow[\text{a secure channel}]{m_3=\{\chi_{D,x_0,T}\} \text{ to gateway through}}$	m_3	
Verif.		$L_1 = x_1$;	
		$L_2 = x_2$;	
		$\xrightarrow{\{L_1\}}$	
			$\theta_D(l_5) \oplus r_s \stackrel{?}{=} L_1$;
			store $Q = \theta_D^2(l_5)$ *and* r_s;
			$L_2 = \theta_D^2(l_5) \oplus r_s$;
			$\xleftarrow{\{\theta_D^2(l_5)\oplus r_s\}}$
		$L_2 \stackrel{?}{=} \theta_D^2(l_5) \oplus r_s$;	

Fig. 5. The Salted PBAP protocol, where Init. and Verif. denote initialization and verification retrospectively.

1. The verifier computes m_5 and runs a *counter* where $0 \leq counter \leq k$
   - $m_5 = \theta_D(l_{i+3}) \oplus \theta_D^2(l_{i+3})$

   and sends message $\{m_5\}$ to the device D. If no response is received within a reasonable interval, the verifier increases the *counter* and computes a new message m_5 and sends it again to the device D. This process continues until the *counter* reaches k. In this situation, the verifier performs the initialization phase.
2. The device D verifies
   - $\theta_D(l_{i+3}) \oplus \theta_D^2(l_{i+3}) \stackrel{?}{=} m_5$

   If it holds true, the device D verifies the verifier and computes
   - $m_6 = \theta_D^3(l_{i+3}) \oplus \theta_D^4(l_{i+3})$
   - *store* $l_{i+7} = \theta_D^4(l_{i+3})$

   and sends the message m_6 to the verifier.
3. The verifier computes and compares
   - $\theta_D^3(l_{i+3}) \oplus \theta_D^4(l_{i+3}) \stackrel{?}{=} m_6$

   If the comparison succeeds, the verifier verifies the device D and stores $l_{i+7} = \theta_D^4(l_{i+3})$ in its secure register.

We describe PBAP by an example which is shown in Fig. 4. Let in this chain, $l_i = l_0$ is the initial link used for the initialization procedure, and l_4, l_8, are the initial links used for the verification procedure.

5. The Salted PBAP scheme

In the PBAP scheme, a central authentication service *AS* responds to all authentication requests and verifies the identity of all the underlying nodes. In cloud-edge (CE) systems, for scalability purposes and to prevent the negative effect of the centralized verifier *AS* as a bottleneck on the total latency, partial of the authentication capabilities of the *AS* is delegated to the gateway *D*. Therefore, it's able to authenticate mutually the underlying nodes without relying upon a central authentication service *AS*. In this section, based on the basic PBAP scheme, we propose a mutual authentication scheme (called Salted PBAP scheme) that is well suited for the CE systems and able to apply in a three-ring IoT structure. The Salted PBAP scheme allows a device D and its respective gateway G to prove their identities and authenticate each other mutually. For this purpose, we transfer the portion of the enrolled PUF chain $\chi_{D,x_0,T} \in \gamma_{D,l_0,M}$ from the central authentication service *AS* to a gateway G. A gateway G uses the sequence of consecutive links $\chi_{D,x_0,T}$ to verify the identities of the underlying devices. Note that the Salted PBAP scheme is always performed when the basic PBAP scheme has been performed successfully and *AS* and the device D have been synchronized on the link l_{i-1} of chain $\gamma_{D,l_0,M}$. The Salted PBAP scheme, see also Fig. 5, has two phases as follows:

5.1. Initialization

1. Device D sends a query message $m_0 = \{l_i = \theta_D(l_{i-1})\}$ to *AS*.
2. *AS* verifies that the value included in the query message corresponds to the link immediately following l_{i-1} in the current chain $\gamma_{D,l_0,M}$. If the two values match, *AS* authenticates the device D and then it extracts a carnet $\tau_{D,t_0,T} = [t_0, ..., t_{T-1}]$ from chain $\gamma_{D,l_0,M}$ starting from link $t_0 = l_{i+4}$. Next, it generates a random salt r_s and computes v_1 and v_2 as following:
   - $v_1 = l_{i+1} \oplus r_s$
   - $v_2 = (l_{i+2} \lll k) \wedge (r_s \ggg k)$

   and sends the message $m_1 = \{v_1, v_2\}$ to the device D.
3. D computes v_2' and compares it with v_2
   - $r_s = \theta_D(l_i) \oplus v_1$
   - $v_2' = (\theta_D^2(l_i) \lll k) \wedge (r_s \ggg k)$
   - $v_2 \stackrel{?}{=} v_2'$

   If the two values match, the device D authenticates *AS* and computes $v_3 = (\theta_D^3(l_i) \ggg k) \wedge (r_s \lll k)$. Then it sends the mes-

sage $m_2 = \{v_3\}$ to *AS* and stores $\theta_D^4(l_i) = l_{i+4}$ and r_s in its local memory.

4. *AS* computes v_3' and compares it with the message m_2
   - $v_3' = (l_{i+4} \ggg k) \wedge (r_s \lll k)$
   - $v_3 \stackrel{?}{=} v_3'$

   If the two values are equal, the *AS* authenticates the device *D*. Then it generates the Salted carnet $\chi_{D,x_0,T} = \{x_i = t_i \oplus r_s, t_i \in \tau_{D,t_0,T}\}$ and sends the message $m_3 = \{\chi_{D,x_0,T}\}$ to the gateway *G* through a secure channel.

5.2. Verification

After the initialization phase, both of the device *D* and the gateway *G* are synchronized on the salted link $x_0 = l_{i+5} \oplus r_s$. Using the salted carnet $\chi_{D,x_0,T}$, the gateway *G* and the device *D* can authenticate each other with exchanging two consecutive salted links $\{x_i, x_{i+1}\}$ of the salted chain $\chi_{D,x_0,T}$. Suppose that the gateway *G* wants to verify the identity of the device *D*.

1. The gateway *G* sends the message $L_1 = \{x_1\}$ to the device *D*.
2. The device *D* compares $\theta_D(l_{i+5}) \oplus r \stackrel{?}{=} L_1$. If two values match, the device *D* verifies the gateway *G* and stores $Q = \theta_D^2(l_{i+5})$ and r_s in its secure memory and sends the message $L_2 = \{\theta_D^2(l_{i+5}) \oplus r_s\}$ to the gateway *G*.
3. *G* compares $L_2 \stackrel{?}{=} \theta_D^2(l_{i+5}) \oplus r_s$. If it holds true, then the gateway *G* verifies the device *D* and the verification procedure is finished successfully.

6. Security analysis of the PBAP and the Salted PBAP protocols

In this section, we analyze and evaluate the security of the two proposed schemes. First, we provide informal proof for the PBAP scheme, and in the following, we informally discuss the security of the Salted PBAP protocol. We show that the two proposed schemes can resist common known attacks. Next, by scyther tool, a formal security analysis of the two proposed schemes is presented.

It is worth noting that through our analysis we consider the use of a reliable and robust PUF function in each device. More precisely, in this model, given challenges $\mathcal{C} \neq \mathcal{C}'$ then $PUF(\mathcal{C})$ and $PUF(\mathcal{C}')$ will be completely different but a PUF returns the same $PUF(\mathcal{C})$ for the same $\mathcal{C}$; even if it is tested for the same $\mathcal{C}$ again and again. In addition, different PUFs also return completely different responses for the same challenge. It is worth noting that designing such a PUF function is an active research area itself, and out of the scope of this paper, an interested reader can see [1,15,27,29,34] for the state of the art of the designing a reliable PUF and its challenges.

6.1. Informal security proof

6.1.1. Informal security proof of the PBAP scheme

1. ***Resistance to de-synchronization attack:*** In the initialization phase, in the first step, the verifier sends the initial link l_i to the device *D*, so it is synchronized on the link l_i with the device *D*. In the verification phase, we increase *counter* by one each time the message m_5 is sent to the device *D*. Two different de-synchronization cases may occur. First, the message m_5 does not properly reach the device, and second, the message m_6 does not reach the verifier. In both cases, the verifier increases *counter* by 1 and resends the message m_5 to the device *D*. If no response is received and *counter* reaches the value *k*, then the verifier detects the de-synchronization has occurred and triggers the re-initialization.
2. ***Resistance to replay attack:*** In the PBAP scheme, all exchanged messages m_1, m_2, m_3, m_4 are updated in each session, by random numbers *n* and *r*, which are generated by the verifier and the device *D* respectively. Therefore the attacker cannot carry out a replay attack by replaying messages from the previous sessions.
3. ***Resistance to traceability attack:*** In an authentication protocol, if in each session, a constant value is transferred between two parties of the protocol (i.e., the tag and the reader) or the response of a tag to a fixed query message includes a constant value, then the attacker can trace and find a particular tag. In the PBAP scheme, all exchanged messages include random numbers *n* and *r*, and change in each session. So the attacker can not trace the two parties of the communication.
4. ***Resistance to impersonation attack:*** In the PBAP scheme, the message m_1 includes the random nonce *n* which is changed in each session. We use the variables k_1, k_2 as parameters that are input to a function to construct messages m_3, m_4. These variables depend on random numbers n, r and an attacker cannot extract these values from any messages transferred between device and verifier without knowing the values *r* and *n*. Moreover, on the device side, we use the variables k_1, k_2 to generate the messages m_2, m_3. So an adversary cannot generate these messages or use the previous transferred messages to perform an impersonation attack.

6.1.2. Informal security proof of the Salted PBAP scheme

1. ***Resistance to de-synchronization attack:*** The Salted PBAP scheme is performed when the basic PBAP scheme has been performed successfully, and both of the device *D* and the verifier *AS* have been synchronized on link l_i. The verifier *AS* and the device *D* store in their secure memories the last link, which is exchanged in each step of the initialization or the verification phase. Therefore, they can detect the de-synchronization attack.
2. ***Resistance to replay attack:*** The values v_1, v_2 include the random salt r_s, therefore they change in each session. The device *D* also uses the random salt r_s to compute the value v_3. Therefore the attacker cannot perform a replay attack.
3. ***Resistance to traceability attack:*** The attacker cannot trace a particular device *D* because all values v_1, v_2, v_3 are changed in each session.
4. ***Resistance to impersonation attack:*** The attacker has no knowledge about the links l_i, l_{i+1} which are used to computing the values v_1, v_2. Also, he cannot compute the message m_1' by using the last exchanged message m_1. So he cannot impersonate as the legitimate verifier *AS*. The message m_2 does not reveal any information about the link l_{i+2}; therefore, the attacker cannot compute and send a response message m_2' to the verifier. Hence, he cannot impersonate as the legitimate device *D*.
5. ***Resistance to man-in-the-middle attack:*** We know that an attacker has no knowledge about $\theta_D(l_{i+1})$, therefore he/she can not compute r_s. Suppose that an attacker modifies the message m_1. In the device side, it computes v_2' and compares it with received message v_2. If an attacker changes v_1 or v_2, the device can detect an error occurred. The validity and integrity of the message v_3 is checked by the verifier. Therefor, our proposed scheme is resistance against a man-in-the-middle attack.
6. ***Resistance to eavesdrop attack:*** The verifier encrypts random salt r_s with $\theta_D(l_i)$, therefore an attacker cannot decrypt the message v_1. The messages v_3, v_4 are encrypted with $\theta_D^2(l_i)$ and $\theta_D^3(l_i)$ respectively. We suppose that an attacker has no knowledge about the links l_i's, therefore he/she can't eavesdrop on the communication between the verifier and a device.

Table 2
Security comparison of the two proposed protocols to other protocols where Imper, Trace, Repl, Desynch, Discl and M-i-m denote impersonation, traceability, reply, desynchronization, disclosure and man-in-the-middle attacks retrospectively.

Protocol	Imper	Trace	Repl	Desynch	Discl	M-i-m
Gope et al. [22]	✓	✓	✓	✓	✓	-
Ebrahimabadi [19]	✓	-	✓	-	✓	-
Kumar [33]	✓	✓	✓	✓	✓	✓
Xu et al. [39]	×	×	×	×	×	-
Barbareschi et al. [6]	×	×	✓	✓	✓	✓
Barbareschi et al. [7]	×	×	✓	×	✓	✓
Two proposed schemes	✓	✓	✓	✓	✓	✓

Table 3
Security analysis result of the PBAP scheme with Scyther.

Claim	Status	Comments
Secret l_0	Ok	No attacks within bounds
Secret n	Ok	No attacks within bounds
Secret r	Ok	No attacks within bounds
Secret m_1	Ok	No attacks within bounds
Secret m_2	Ok	No attacks within bounds
Secret m_3	Ok	No attacks within bounds
Secret m_4	Ok	No attacks within bounds
Secret m_5	Ok	No attacks within bounds
Secret m_6	Ok	No attacks within bounds
Niagree	Ok	No attacks within bounds
Nisynch	Ok	No attacks within bounds
Alive	Ok	No attacks within bounds
Weakagree	Ok	No attacks within bounds

Table 4
FFR for initialization and verification phases of PBAP and Salted PBAP protocols.

	Initialization	Verification
PBAP	$4e^{-10}$	$4e^{-10}$
Salted PBAP	$5e^{-10}$	$2e^{-10}$

Table 5
Device/verifier or gateway FAR for initialization and verification phases of PBAP and Salted PBAP protocols ($N = 128bits$).

	Protocol	Initialization	Verification
Device	PBAP	$7,45e^{-155}$	$7,45e^{-155}$
	Salted PBAP	$2,19e^{-193}$	$8,63e^{-78}$
Verifier	PBAP	$2,53e^{-116}$	$7,45e^{-155}$
	Salted PBAP	$2,53e^{-116}$	$8,63e^{-78}$

As shown in Table 2, we compare the security of the PBAP and Salted PBAP schemes with their predecessors and some other related protocols. The Comparison results show that the improved protocols have an acceptable level of security and could satisfy the security requirements of an authentication protocol for the Internet of things applications. Besides our protocols, the protocol proposed in [22] also provides desired security, but with more computation cost on the tag's side.

6.2. Formal security proof

Using software tools is an approach to evaluate the security of a cryptographic protocol. Several software tools like Avispa, Scyther, ProVerif, etc., usually support cryptographic primitives such as symmetric and asymmetric cryptography, hash functions, digital signatures, and bit-commitment. Authentication protocols involve at least two parties (e.g. the reader, the tag), and each party plays a role in authentication protocols. All events that occur in each party, like computing, comparing, sending, or receiving messages, are defined in a set of roles (e.g. role of the tag, reader's role). We use the Scyther tool to evaluate the security of our two proposed protocols. The roles of our two protocols are implemented by Security Protocol Description Language (SPDL) as represented in Appendix A. The report of the scyther tool shows that our two protocols are safe against all threats. Security analysis result of the two proposed schemes is presented in Table 3.

6.3. PUF security and reliability issues

The security and robustness of the proposed mutual authentication protocols are partly substantiated on the PUF security and reliability. One of the significant drawbacks of PUF technology is the instability of their outputs caused by the operational conditions, aging of the device, etc. The most extended mitigation techniques are the fuzzy-extractor blocks and helper data [17,23]. These blocks are dedicated to turning the noisy PUF responses into reliable responses using sophisticated error-correcting codes (ECC). Other solutions could be applied depending on the kind of PUF used. For instance, some advanced enrollment techniques try to enhance the responses of PUFs in hostile scenarios (e.g., improving reliability by removing unreliable bits [36] or re-enrollment during the lifetime of the PUF to mitigate the aging effects [38]).

In this regard, a measure to determine the quality of an authentication process that involves unreliable responses of a PUF instance is the false rejection rate (FRR). The FRR for the proposed authentication protocols is directly proportional to the number of PUF generations in each authentication phase. It is noteworthy that a simple bit-flip in one of the generations would change completely the chain sequence generating a mismatch between the expected PUF responses. Table 4 summarizes the FRR for each phase of the proposed protocols. To obtain these results, we have considered the scenario presented in [7], where an Anderson PUF and a BCH(63,51) error correction code are used to reach a final instability of $p = 10^{-10}$. The $FRR = 1 - (1 - p)^x$, where x is the number of new PUF responses that must be generated in the device.

The influence of the PUF on the security of the authentication mechanisms presented is related to the probability that an attacker can generate the correct response for a given challenge. In a PUF, assuming that there is no information entropy loss [7], this probability depends only on the number of bits (N) of the response $(1/2^N)$. At the protocol level, the False Acceptance Rate (FAR) measures the probability of a forge device/verifier being authenticated. We can define the FAR as follows: $FAR = (1/2^N)^x$, where N is the number of bits of each PUF response and x is the number of PUF responses that the device or the verifier must generate. Table 5

All in all, the proposed protocols have obtained good FFR and FAR metrics that are in line with the results presented in [7].

Table 6
Implementation results for Xilinx FPGA Zynq-7.

Algorithm	Encryption Function (AES)	Hash Function (SHA-3)	PRNG	PUF (Anderson PUF)	Elliptic Curve Cryptography
Approximate area (Slices LUTs)	≈ 431 [16]	≈ 735 [26]	≈ 1371 [32]	≈ 3936 [6]	≈ 2783 [35]

Table 7
Cost comparison, where S. denotes "Salted".

Protocol	Ebrahimabadi et al. [19]	Kumar et al. [33]	Gope et al. [22]	Xu et al. [39]	PHEMAP [6]	S. PHEMAP [7]	PBAP	S.PBAP
Computation cost of the tag	1×PUF	1×Hash+ 1×PUF+ 1×PRNG+ 1×Enc	5×Hash+ 2×PUF+ 2×PRNG	6×PUF+ 1×PRNG	7×PUF+ 1×PRNG	5×PUF	3×PUF+ 1×PRNG	5×PUF
Approximate area (Slices LUTs)	≈ 3936	≈ 6473	≈ 6042	≈ 5307	≈ 5307	≈ 3936	≈ 5307	≈ 3936

Machine learning (ML) attacks against PUFs are also an emerging source of concern for PUF-based technologies. Typically, ML attacks involve an attacker collecting a large subset of CRPs to create a mathematical model to predict unknown CRPs. Designing ML resistant PUFs such as XOR-APUF, FF-PUF or MPUF has emerged as an excellent solution to resist ML attacks [14]. At the protocol level, lockdown mechanisms (e.g.([40]) and the inclusion of randomization methods to preserve the mappings between challenges and responses is also an exciting solution [14]. In that vein, the PBAP protocol implements a randomization mapping of the challenge-response space at the generation of message *m1* that prevents several kinds of ML attacks. In addition to this, a lockdown mechanism like the one proposed in [40] could be easily integrated into the algorithm to avoid an excessive number of CRPs requests by an adversary.

7. Implementation

IoT systems typically consist of several sensors, RFID tags, and computing nodes with restrictions on power, processing capacity, and memory. It is worth noting, even mobile readers have enough resources to support conventional cryptographic primitives. Therefore, to design an authentication protocol, our main concern is edge devices, e.g., passive RFID tags. Hence, we focus on the computational cost of the edge devices and try to design a low-cost scheme. This section presents two implementations of the PBAP protocol in two different edge devices: a low-power microcontroller and an FPGA.

We have selected the implementation strategy presented in [10] where a weak PUF (e.g. SRAM) and a symmetric cypher (e.g. AES) are combined. The selected microcontroller for the PBAP implementation was the STMicroelectronics Nucleo F401RE board, equipped with a 32-bit microcontroller ARM® Cortex®-M4, 512 kB flash and 96 kB SRAM. We have used the official cryptographic library provided by STMicroelectronics to implement a cypher (AES-128) quickly. We have selected a memory region with a good distribution of 0's and 1's (uniformity) to obtain the PUF keys (starting address: 0x200003f4). We have collected and saved three different PUF chains as proof of concept implementation. The authentication service (verifier) was implemented using python on a desktop. A serial port was used to simulate the communication between devices. The average mutual authentication time measured in 100 consecutive operations using PBAP protocol is 59,3 ms.

We have selected the Xilinx Zybo Z7-20 board for the FPGA implementation, which embeds an xc7z020clg400-1 FPGA. The hardware resources used to implement the PBAP protocol in this FPGA are 6878 Slice LUTs and 707 flip-flops. These results do not include the hardware resources necessary to implement a true random number generator (TRNG).

The presented results depend on the selected PUF, cypher, architecture and technology. In addition, no optimizations have been carried out in the implementations. In order to have a fair comparison with other protocols, in terms of computational cost on the device side, we have counted the number of PUF, PRG, and hash functions in each protocol for evaluating the computational performance of our two schemes. The impact of logic operations such as exclusive OR (XOR) and SHIFT operations is meagre. In Table 6, we show the implementation cost based on the reports for cryptographic primitives in related literature, i.e. hash function (in this case SHA-3 [9]), PUF (in this case Anderson PUF [4]), PRNG and Elliptic Curve Cryptography (ECC) that are used by many authentication protocols. The performance comparison results of the PBAP and Salted PBAP with previously proposed schemes are presented in Table 7. It should be noted that some other related protocols are omitted in our comparison because we believe many passive tags can not comply with their requirements, such as the support of hash functions, symmetric encryption algorithms, public key algorithms, and we just mentioned [22] as an example.

8. Conclusion

This article analyzes the security of two recently proposed PUF-based protocols, i.e. PHEMAP and Salted PHEMAP. The detailed analysis shows that these protocols are vulnerable to various attacks such as device impersonation, de-synchronization, and traceability attacks. Based on the same design as Barbareschi et al.'s protocols [6] [7], we have proposed two lightweight authentication protocols based on PUF chains called PBAP and Salted PBAP. The proposed schemes use low-cost operations such as exclusive OR, SHIFT, and PUF, so they are compatible with low computing devices such as RFIDs and microsensors. Moreover, through formal and informal methods, we have shown how the proposed schemes have not inherited the security weaknesses of the PHEMAP and Salted PHEMAP schemes and are resistant against common known attacks.

CRediT authorship contribution statement

Morteza Adeli: Designing, Experimentation, Software, Validation, and Writing;

Nasour Bagheri: Conceptualization, Analyzing, Validation, Writing - Review and Editing.

Honorio Martin: Experimentation, Software, Validation, Review, and Editing.

Pedro Peris-Lopez: Conceptualization, Methodology, Experimentation, Validation, Writing – Review, Editing, Funding, and Supervision.

Declaration of competing interest

The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

Acknowledgment

This work was supported by Shahid Rajaee Teacher Training University under grant number 3564; and Spanish Ministry of Science, Innovation and Universities grant PID2019-111429RBC21 (ODIO); and by the Comunidad de Madrid (Spain) under the projects PUCFA (PUCFA-CM-UC3M) and CYNAMON (P2018/TCS-4566)–cofinanced by European Structural Funds (ESF and FEDER).

Appendix A. Security Protocol Description Language model of the proposed protocols

```
const PUF: Function;
const XOR: Function;
const SHIFT: Function;
const AND: Function;
const OR: Function;
macro m1 = XOR (PUF (l0),n);
macro m2 = XOR (PUF (l0),r);
macro m3 = XOR (OR (AND (SHIFT (PUF (l1),k1),SHIFT (r,k2)),SHIFT
(PUF (l0),k2)),SHIFT (n,k1));
macro m4 = OR (AND (SHIFT (PUF (l2),k1),SHIFT (r,k1)),SHIFT (PUF
(l2),k2));
macro m5=XOR (PUF (l3),PUF (PUF (l3)));
macro m6=XOR (PUF (PUF (PUF (l3))),PUF (PUF (PUF (PUF (l3)))));
protocol ThePBAPprotocol (Verifier,Device)
role Verifier
secret l0,l1,l2,l3;
fresh r,k1,k2,n: Nonce;
send1 (Verifier,Device,m1,m4);
recv2 (Device,Verifier,m2,m3);
match (n,XOR (m3,SHIFT (OR (AND (SHIFT (PUF (l1),k1),SHIFT
(r,k2)),SHIFT (PUF (l0),k2)),k1)));
send3 (Verifier,Device,m5);
recv4 (Device,Verifier,m6);
claim (Verifier, Secret, r);
claim (Verifier, Secret, n);
claim (Verifier,Niagree); claim (Verifier,Nisynch);
claim (Verifier,Alive); claim (Verifier,Weakagree);
role Device
secret l0,l1,l2,l3;
fresh r,k1,k2,n: Nonce;
recv1 (Verifier,Device,m1,m4);
match (m4,OR (AND (SHIFT (PUF (l2),k1),SHIFT (r,k1)),SHIFT (PUF
(l2),k2)));
send2 (Device,Verifier,m2,m3);
recv3 (Verifier,Device,m5);
match (m5,XOR (PUF (l3),PUF (PUF (l3))));
send4 (Device,Verifier,m6);
claim (Device, Secret, r);
claim (Device, Secret, n);
claim (Device,Niagree); claim (Device,Nisynch);
claim (Device,Alive); claim (Device,Weakagree);
```

References

[1] M.S. Alkatheiri, A.R. Sangi, S. Anamalamudi, Physical unclonable function (puf)-based security in Internet of things (iot): key challenges and solutions, in: B.B. Gupta, G.M. Pérez, D.P. Agrawal, D. Gupta (Eds.), Handbook of Computer Networks and Cyber Security, Principles and Paradigms, Springer, 2020, pp. 461–473.

[2] M.N. Aman, K.C. Chua, B. Sikdar, Mutual authentication in IoT systems using physical unclonable functions, IEEE Int. Things J. 4 (5) (Oct 2017) 1327–1340.

[3] M.H. Ameri, M. Delavar, J. Mohajeri, Provably secure and efficient PUF-based broadcast authentication schemes for smart grid applications, Int. J. Commun. Syst. 32 (8) (2019) e3935.

[4] J.H. Anderson, A puf design for secure fpga-based embedded systems, in: 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC), 2010, pp. 1–6.

[5] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, M. Yung, End-to-end design of a puf-based privacy preserving authentication protocol, in: T. Güneysu, H. Handschuh (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2015 – 17th International Workshop, Proceedings, Saint-Malo, France, September 13–16, 2015, in: Lecture Notes in Computer Science, vol. 9293, Springer, 2015, pp. 556–576.

[6] M. Barbareschi, A.D. Benedictis, N. Mazzocca, A PUF-based hardware mutual authentication protocol, J. Parallel Distrib. Comput. 119 (2018) 107–120.

[7] M. Barbareschi, A.D. Benedictis, E.L. Montagna, A. Mazzeo, N. Mazzocca, A PUF-based mutual authentication scheme for cloud-edges IoT systems, Future Gener. Comput. Syst. 101 (2019) 246–261.

[8] Y. Bendavid, N. Bagheri, M. Safkhani, S. Rostampour, Iot device security: challenging "a lightweight RFID mutual authentication protocol based on physical unclonable function", Sensors 18 (12) (2018).

[9] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche Keccak, in: T. Johansson, P.Q. Nguyen (Eds.), Advances in Cryptology – EUROCRYPT 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 313–314.

[10] M. Bhargava, K. Mai, An efficient reliable puf-based cryptographic key generator in 65 nm cmos, in: 2014 Design, Automation Test in Europe Conference Exhibition (DATE), 2014, pp. 1–6.

[11] A. Braeken, Puf based authentication protocol for IoT, Symmetry 10 (8) (2018).

[12] U. Chatterjee, R.S. Chakraborty, D. Mukhopadhyay, A PUF-based secure communication protocol for IoT, ACM Trans. Embed. Comput. Syst. 16 (3) (Apr. 2017) 67.

[13] W. Che, M. Martin, G. Pocklassery, V.K. Kajuluri, F. Saqib, J. Plusquellic, A privacy-preserving, mutual PUF-based authentication protocol, Cryptography 1 (1) (2016).

[14] S. Chen, B. Li, Z. Chen, Y. Zhang, C. Wang, C. Tao, Novel strong-puf-based authentication protocols leveraging Shamir's secret sharing, IEEE Int. Things J. (2021).

[15] K. Choi, S. Baek, J. Heo, J. Hong, A 100% stable sense-amplifier-based physically unclonable function with individually embedded non-volatile memory, IEEE Access 8 (2020) 21857–21865.

[16] L. Daoud, F. Hussein, N. Rafla, Optimization of advanced encryption standard (AES) using vivado high level synthesis (HLS), in: G. Lee, Y. Jin (Eds.), Proceedings of 34th International Conference on Computers and Their Applications, CATA 2019, Honolulu, Hawaii, USA, March 18–20, 2019, in: EPiC Series in Computing, vol. 58, EasyChair, 2019, pp. 36–44.

[17] J. Delvaux, D. Gu, D. Schellekens, I. Verbauwhede, Helper data algorithms for puf-based key generation: overview and analysis, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 34 (6) (2015) 889–902.

[18] J. Delvaux, R. Peeters, D. Gu, I. Verbauwhede, A survey on lightweight entity authentication with strong PUFs, ACM Comput. Surv. 48 (2) (2015) 26.

[19] M. Ebrahimabadi, M. Younis, N. Karimi, A puf-based modeling-attack resilient authentication protocol for iot devices, IEEE Int. Things J. 9 (5) (2022) 3684–3703.

[20] P. Gope, A.K. Das, N. Kumar, Y. Cheng, Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks, IEEE Trans. Ind. Inform. 15 (9) (2019) 4957–4968.

[21] P. Gope, J. Lee, T.Q.S. Quek, Lightweight and practical anonymous authentication protocol for RFID systems using physically unclonable functions, IEEE Trans. Inf. Forensics Secur. 13 (11) (2018) 2831–2843.

[22] P. Gope, B. Sikdar, Lightweight and privacy-preserving two-factor authentication scheme for IoT devices, IEEE Int. Things J. 6 (1) (Feb 2019) 580–589.

[23] C. Herder, L. Ren, M. van Dijk, M.-D. Yu, S. Devadas, Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions, IEEE Trans. Dependable Secure Comput. 14 (1) (2017) 65–82.

[24] C. Hristea, F.L. Tiplea, A PUF-based destructive private mutual authentication RFID protocol, in: J. Lanet, C. Toma (Eds.), Innovative Security Solutions for Information Technology and Communications – 11th International Conference, SecITC 2018, Revised Selected Papers, Bucharest, Romania, November 8–9, 2018, in: Lecture Notes in Computer Science, vol. 11359, Springer, 2018, pp. 331–343.
[25] C. Hristea, F.L. Tiplea, Destructive Privacy and Mutual Authentication in Vaudenay's RFID Model, IACR Cryptology ePrint Archive, vol. 2019, 2019, p. 73.
[26] H.S. Jacinto, L. Daoud, N. Rafla, High level synthesis using vivado hls for optimizations of sha-3, in: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), 2017, pp. 563–566.
[27] D. Jeon, J. Baek, Y. Kim, J. Lee, D.K. Kim, B. Choi, A physical unclonable function with bit error rate ×10−8 based on contact formation probability without error correction code, IEEE J. Solid-State Circuits 55 (3) (2020) 805–816.
[28] L. Kulseng, Z. Yu, Y. Wei, Y. Guan, Lightweight mutual authentication and ownership transfer for RFID systems, in: 2010 Proceedings IEEE INFOCOM, 2010, pp. 1–5.
[29] S. Lee, M. Oh, Y. Kang, D. Choi, Design of resistor-capacitor physically unclonable function for resource-constrained iot devices, Sensors 20 (2) (2020) 404.
[30] R. Maes, Physically Unclonable Functions: Constructions, Properties and Applications. SpringerLink: Bücher, Springer, Berlin Heidelberg, 2013.
[31] M. Majzoobi, M. Rostami, F. Koushanfar, D.S. Wallach, S. Devadas, Slender PUF protocol: a lightweight, robust, and secure authentication by substring matching, in: 2012 IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, May 24–25, 2012, 2012, pp. 33–44.
[32] B. Paul, G. Trivedi, P. Jan, Z. Němec, Efficient prng design and implementation for various high throughput cryptographic and low power security applications, in: 2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA), 2019, pp. 1–6.
[33] M. Prasanna Kumar, N. Nalini, P.N. Hamsavath, An effective puf based lightweight authentication and key sharing scheme for iot devices, in: N.R. Shetty, L.M. Patnaik, H.C. Nagaraj, P.N. Hamsavath, N. Nalini (Eds.), Emerging Research in Computing, Information, Communication and Applications, Springer Singapore, Singapore, 2022, pp. 257–264.
[34] D.P. Sahoo, D. Mukhopadhyay, R.S. Chakraborty, P.H. Nguyen, A multiplexer-based arbiter PUF composition with enhanced reliability and security, IEEE Trans. Comput. 67 (3) (2018) 403–417.
[35] P. Sasdrich, T. Güneysu, Efficient elliptic-curve cryptography using curve25519 on reconfigurable devices, in: D. Goehringer, M.D. Santambrogio, J.M.P. Cardoso, K. Bertels (Eds.), Reconfigurable Computing: Architectures, Tools, and Applications, Springer International Publishing, Cham, 2014, pp. 25–36.
[36] A. Schaub, J.-L. Danger, S. Guilley, O. Rioul, An improved analysis of reliability and entropy for delay pufs, in: 2018 21st Euromicro Conference on Digital System Design (DSD), 2018, pp. 553–560.
[37] K. Sha, W. Wei, T.A. Yang, Z. Wang, W. Shi, On security challenges and open issues in internet of things, Future Gener. Comput. Syst. 83 (2018) 326–337.
[38] J. Wallrabensteing, Remote re-enrollment of physical unclonable functions, in: US. Patent 2019/0138753, 2019.
[39] H. Xu, J. Ding, P. Li, F. Zhu, R. Wang, A lightweight RFID mutual authentication protocol based on physical unclonable function, Sensors 18 (3) (2018).
[40] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, I. Verbauwhede, A lockdown technique to prevent machine learning on pufs for lightweight authentication, IEEE Trans. Multi-Scale Comput. Syst. 2 (3) (2016) 146–159.
[41] J. Zhang, X. Tan, X. Wang, A. Yan, Z. Qin, T2FA: transparent two-factor authentication, IEEE Access 6 (2018) 32677–32686.
[42] Y. Zhang, B. Li, B. Liu, Y. Hu, H. Zheng, A privacy-aware pufs-based multi-server authentication protocol in cloud-edge iot systems using blockchain, IEEE Int. Things J. (2021).

Morteza Adeli is PhD candidate at Department of Science, Shahid Rajaee Teacher Training University, Tehran, Iran. He received his M.S. and B.S. degrees in mathematics from Iran University of Science and Technology (IUST) and Shahid Rajaee Teacher Training University respectively. His research interests include designing and analysis of cryptographic protocols for constrained environment, such as RFID tags and IoT edge devices.

Nasour Bagheri received the M.S. and Ph.D. degrees in Electrical Engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2002 and 2010 respectively. He is currently an associate professor at the electrical engineering department, Shahid Rajaee Teacher Training University, Tehran, Iran. He is the author of over 100 articles in information security. His research interests include cryptology, more precisely, designing and analysis of symmetric schemes such as lightweight ciphers, e.g., block ciphers, hash functions and authenticated encryption schemes, cryptographic protocols for constrained environment, such as RFID tags and IoT edge devices and hardware security, e.g., the security of symmetric schemes against side-channel attacks such as fault injection and power analysis.

Honorio Martin received the Ph.D. degree in electronics engineering from the Universidad Carlos III de Madrid, Spain, in 2015. He is currently a Postdoctoral Researcher with the Department of Electronic Technology, Universidad Carlos III de Madrid. His current research interests include the study of lightweight cryptography, hardware implementations, radiofrequency identification systems, and low-power design.

Pedro Peris-Lopez received the M.Sc. degree in telecommunications engineering and the Ph.D. degree in computer science from the Carlos III University of Madrid, Spain, in 2004 and 2008, respectively. He is currently an Associate Professor with the Department of Computer Science, Carlos III University of Madrid. His research interests are in the field of cybersecurity and e-health, digital forensics and hardware security. In these fields, he has published a large number of articles in specialized journals (57) and conference proceedings (44). His works have more than 4960 citations, and his H-index is 32. For additional information see: https://www.lightweightcryptography.com/.