Systematic Cryptanalysis of PUF-Based Authentication Protocols for IoT, A Case Study

Amir Masoud Aminian Modarres and Ghazaleh Sarbishaei

***Abstract*—Almost all IoT applications require secure user authentication protocols. Due to the limitations of most IoT devices and the possibility of physical attacks, lightweight authentication protocols based on Physical Unclonable Functions (PUF) are widely used. To design an efficient scheme, it is helpful to learn weaknesses of existing protocols and provide guidelines for future protocol designers. In this letter, using a formal framework, we discuss the vulnerabilities of a recent PUF-based protocol proposed by Gope et al. We prove our claims concerning protocol vulnerabilities in this framework according to propositional logic. Moreover, we suggest countermeasures for improving the protocol's security.**

***Index Terms*—Network security, authentication protocols, formal analysis of protocols, Internet of Things, PUF.**

I. Introduction

With the increasing use of IoT technology in our daily lives, it is more critical than ever to use efficient security mechanisms to protect ourselves against cyberattacks. As most security mechanisms generate secret keys during authentication procedure, any vulnerability in authentication protocols increases cyberattack risks. It is therefore essential to use well-designed authentication protocols. Recently, several authentication protocols have been reported for IoT applications. The limited computing resources, memory, and power in most IoT devices, as well as their vulnerability to physical attacks, encourage protocol designers to propose lightweight PUF-based authentication protocols [1], [2], [3], [4]. However, most of these schemes suffer from some shortcomings. Understanding the weaknesses and vulnerabilities of previous authentication techniques is necessary for developing better and more secure protocols. So, performing a systematic and precise cryptanalysis of previous protocols is of great importance.

Recently, Mall et al. [1] reviewed several PUF-based authentication protocols for IoT applications and compared their performance and security features. Although they compared the computational complexity of the protocols in detail, they only mentioned their security weaknesses without performing any analysis. In another study, Lounis and Zulkernine [5] analyzed the security of some PUF-based authentication protocols and reported their security issues. They also provided recommendations for future protocol designers. Studies presented by Cho et al. [6] and Adeli et al. [7] provide security analyses of particular protocols. Similar security analysis has also been presented in our recently published paper for a new PUF-based authentication protocol [3].

However, machine learning-based modeling attacks may be performed on PUF chips, if an attacker collects a set of challenge-response pairs (CRPs) and uses them to create a PUF model [8]. Preventing ML attacks can be performed at different levels. Some researchers use more complex PUF architectures to provide resistance at the circuit level. Recently, Kraleva et al. [9] presented a cryptanalysis of PUF-based authentication protocols and discussed the vulnerability of different PUF architectures to ML attacks. Some other methods aim to prevent these attacks at the protocol level. The protocols presented by Yu et al. [10] and Gao et al. [11] fall into this category. However, analysis presented in [8] showed that both [10] and [11] are vulnerable to some protocol attacks.

Recently, Gope et al. [2] proposed a lightweight authentication protocol that is resistant to ML attacks. This scheme utilizes a weak PUF chip for key generation. It also uses a strong PUF chip which is a one-time PUF and is reset every session, so even if the attacker has physical access to it, he cannot predict its behavior. Moreover, unlike most previous protocols, [2] considers noisy behavior of strong PUF chips. Although this scheme uses some good ideas, it has serious security weaknesses.

In this letter, we present a thorough cryptanalysis of Gope et al.'s protocol [2]. In contrast to most previous studies, we presented a formal framework for analyzing this protocol. Using this framework, we prove our claims concerning protocol vulnerabilities through a procedure based on propositional logic. Additionally, based on cryptanalysis results, we suggest countermeasures and methods for improving the protocol's security. It is worth noting that our presented framework can be used as a systematic method for precisely analyzing similar schemes as well.

The rest of this letter is organized as follows: Section II reviews Gope et al.'s scheme [2], and its cryptanalysis is presented in Section III. In Section IV, some suggestions are made to improve the protocol's security. Finally, a conclusion is presented in Section V.

II. Review of Gope et al.'s Scheme

This section reviews Gope et al.'s authentication protocol [2] briefly. As shown in Fig. 1, the scheme uses a three-layered architecture for IoT systems, including servers, gateways, and IoT devices. IoT devices can be medical sensors or other smart devices. The devices connect to a server using

Manuscript received 23 May 2023; accepted 21 July 2023. Date of publication 25 July 2023; date of current version 2 January 2024. The associate editor coordinating the review of this article and approving it for publication was K. Almeroth. *(Corresponding author: Amir Masoud Aminian Modarres.)*

The authors are with the Department of Electrical and Bio Electric Engineering, Sadjad University, Mashhad 9188148848, Iran (e-mail: am_aminian@sadjad.ac.ir; gh_sarbisheie@sadjad.ac.ir).

Digital Object Identifier 10.1109/LNET.2023.3298775

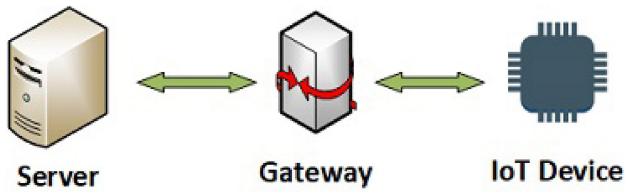

Fig. 1. System architecture.

TABLE I
NOTATIONS USED IN THE PROTOCOL

Notation	Description
$ID_t, \widetilde{ID_t}$	Temporary identities of IoT device T for the current session and next session (respectively)
$WPUF$	System-on-Chip weak PUF
$OPUF, \widetilde{OPUF}$	One-Time strong PUF configuration for the current session and next session (respectively)
(C_w, R_w)	CRP of the weak PUF
$(C_o, R_o), (\widetilde{C_o}, \widetilde{R_o})$	CRPs of the one-time strong PUF for the current session and next session (respectively)
$\widehat{R_o}$	Noisy response computed by the OPUF
MK	Master key of the Server
N_j	Random nonces
$H(.)$	A cryptographic hash function
$\oplus$, $\|\|$	XOR operation , Concatenation (respectively)
$\boldsymbol{A} \longrightarrow \boldsymbol{B} : \boldsymbol{M}$	A sends message M to B via a non-secure channel.
$\boldsymbol{A} \xrightarrow{\text{secure}} \boldsymbol{B} : \boldsymbol{M}$	A sends message M to B via a secure channel.

a wireless gateway, which transfers data between them and the server.

Assumptions:

1- The server is assumed to be a trusted party which is protected with well-known security methods.

2- Each IoT device contains two PUF chips: a weak PUF (WPUF) and a strong PUF that is reconfigurable or dynamic, known as an OPUF (one-time PUF). OPUF is reset after each session, resulting in a change in its behavior in response to challenges. As a result, an ML attack cannot be performed using the previous CRPs after each session.

3- WPUF is robust, while OPUF has noisy behavior and produces different responses in different measurements. Fractional Hamming Distance (FHD) is used as a metric to compare these different measurements [2].

The protocol consists of two phases: a) registration and b) authentication. Table I lists the notations used in this letter, and Table II shows different steps of this protocol.

Registration phase: In this phase, the communication channel between the server and the device is considered secure. The server S generates two random challenges, C_o and C_w, and sends them to the device T. Upon receiving the challenges, the device computes the responses of its WPUF and OPUF, R_w and R_o, and generates the secret K_i for the ith session, using a random key and R_w. In this protocol, R_o is divided into two parts. The device sends its real identity, RID, along with K_i, R_w, and R_o to the server (Steps 1-4 in Table II). The server computes a temporary ID, ID_t, using its secure master key and sends it to the device. The device stores ID_t in its memory, and the random key and C_w in the memory attached to WPUF (Steps 5-7).

TABLE II
GOPE ET AL.'S PROTOCOL

Registration Phase:

(1) S Generate (C_o, C_w),
(2) $\boldsymbol{S} \xrightarrow{\text{secure}} \boldsymbol{T}$: $\{\boldsymbol{C_o}, \boldsymbol{C_w}\}$.
(3) T Generate Random key K_{i-1},
Compute: $R_w = WPUF(C_w)$,
$K_i = H(R_w||K_{i-1})$; $R_o = OPUF(C_o) = R_o^1||R_o^2$,
(4) $\boldsymbol{T} \xrightarrow{\text{secure}} \boldsymbol{S}$: $\{\boldsymbol{RID}, \boldsymbol{R_o}, \boldsymbol{K_i}, \boldsymbol{R_w}\}$.
(RID is the real identity of the device)
(5) S Read master key: MK,
Generate a temporary ID: $ID_t = H(R_o||RID||MK)$,
Store ID_t, C_o, R_o, K_i, R_w,
(6) $\boldsymbol{S} \xrightarrow{\text{secure}} \boldsymbol{T}$: $\{\boldsymbol{ID_t}\}$.
(7) T Store ID_t in device memory,
Store K_{i-1}, C_w in memory attached to WPUF.

Authentication Phase:

(1) T Generate nonce N_t,
Compute: $R_w = WPUF(C_w)$; $K_i = H(R_w||K_{i-1})$
$N_t^* = N_t \oplus K_i$; $V_0 = H(N_t^*||K_i)$,
(2) $\boldsymbol{T} \longrightarrow \boldsymbol{S} : \boldsymbol{M_1} = \{\boldsymbol{ID_t}, \boldsymbol{N_t^*}, \boldsymbol{V_0}\}$.
(3) S Check ID_t,
Retrieve (C_o, R_o, K_i, R_w) from database where $R_o = (R_o^1||R_o^2)$,
Verify V_0,
(4) Generate nonce N_s,
Compute: $N_t = N_t^* \oplus K_i$; $N_s^* = N_s \oplus K_i$,
$R_o^{1*} = R_o^1 \oplus K_i$; $V_1 = H(N_s||K_i||R_o^{1*}||N_t)$,
(5) $\boldsymbol{S} \longrightarrow \boldsymbol{T} : \boldsymbol{M_2} = \{\boldsymbol{C_o}, \boldsymbol{R_o^{1*}}, \boldsymbol{V_1}, \boldsymbol{N_s^*}\}$.
(6) T Compute: $N_s = N_s^* \oplus K_i$; $R_o^1 = R_o^{1*} \oplus K_i$,
Verify V_1,
(7) Generate $\widehat{R_o} = \{\widehat{R_o^1}||\widehat{R_o^2}\} = OPUF(C_o)$,
$\widehat{R_o}$ is the OPUF response computed by the device
Check $FHD(\widehat{R_o^1}, R_o^1) < \Delta$ and proceed if it is true,
(8) Compute Session Key: $K_{ts} = H(N_t||N_s||R_o^1||\widehat{R_o^2})$.
(9) Compute: $X = \widehat{R_o^2} \oplus K_i$
(10) Generate challenge for next session: $\widetilde{C_o} = H(C_o||N_s||N_t)$,
Reconfigure $OPUF$ for next session: $\widetilde{OPUF}$,
Compute OPUF response with new configuration:
$\widetilde{R_o} = \widetilde{OPUF}(\widetilde{C_o})$; $\widetilde{R_o}^* = \widetilde{R_o} \oplus K_i$,
(11) Generate temporary ID for next session: $\widetilde{ID_t} = H(ID_t||\widetilde{R_o})$
(12) Compute: $V_2 = H(K_i||\widetilde{R_o}^*||N_s||X)$
(13) Store $\widetilde{ID_t}$ in device memory,
Replace K_{i-1} with K_i in memory attached to WPUF.
(14) $\boldsymbol{T} \longrightarrow \boldsymbol{S} : \boldsymbol{M_3} = \{\boldsymbol{\widetilde{R_o}^*}, \boldsymbol{X}, \boldsymbol{V_2}\}$.
(15) S Verify V_2,
(16) Compute: $\widehat{R_o^2} = X \oplus K_i$,
Check $FHD(\widehat{R_o^2}, R_o^2) < \Delta$ and proceed if it is true,
(17) Compute Session Key: $K_{ts} = H(N_t||N_s||R_o^1||\widehat{R_o^2})$.
(18) Compute: $\widetilde{C_o} = H(C_o||N_s||N_t)$; $\widetilde{R_o} = \widetilde{R_o}^* \oplus K_i$,
$\widetilde{ID_t} = H(ID_t||\widetilde{R_o})$; $K_{i+1} = H(K_i||R_w)$,
(19) Refresh database with $\widetilde{ID_t}, \widetilde{C_o}, \widetilde{R_o}, K_{i+1}, R_w$.

Authentication phase: To initiate the protocol, the device generates a nonce, computes the secret K_i, and the hash value V_0. It then sends message M_1 to the server (Steps 1-2). The server S checks the device's ID_t and retrieves the stored parameters in its database. Then, S verifies V_0; generates a nonce; encrypts the first part of R_o; computes the hash V_1, and sends the message M_2 to the device (Steps 3-5).

The device T receives M_2 and verifies V_1. It then generates the OPUF's response for the received challenge and splits it into two parts. As mentioned, OPUF has a noisy behavior; so, there might be a slight difference between the generated response and the one stored in the server's database. The server has sent only the first part of R_o via message M_2. Therefore, the device compares this part with the first part of its own response, by computing the FHD, and authenticates the server (Step 7). Next, the device T generates the session key K_{ts}, and reconfigures its OPUF. It then computes the new challenge $\widetilde{C_o}$ and its corresponding response $\widetilde{R_o} = \widetilde{OPUF}(\widetilde{C_o})$ and updates the ID_t for the next session (Steps 8-11). Finally, T stores the updated $\widetilde{ID_t}$, computes the hash value V_2, and sends message M_3 to the server (Steps 12-14).

The server S verifies V_2 and authenticates the device, by comparing the second part of the R_o stored in its database with the part received via M_3. At the end, it computes the session key and refreshes its database with the parameters updated for the next session (Steps 15-19).

III. Cryptanalysis of Gope et al.'s Scheme

We analyze Gope et al.'s scheme [2] in this section in order to demonstrate its weaknesses and vulnerabilities. A formal framework is presented to develop our arguments according to propositional logic.

To evaluate the protocol's performance, we define the adversary's capabilities:

Adversary model: Based on the DY threat model [12]:

$\boxed{A_1}$: An adversary $\mathcal{A}$ can eavesdrop on the device's communication channel with the server. Messages sent through this channel can be tampered with, replayed, or intercepted by an attacker.

Based on the CK threat model [13]:

$\boxed{A_2}$: An adversary $\mathcal{A}$ may obtain secret credentials, including session keys and session-specific temporary information.

Based on the fact that IoT devices are usually unprotected:

$\boxed{A_3}$: An adversary $\mathcal{A}$ may physically access a device and perform noninvasive attacks, such as reading the contents of the memory or applying challenges to a PUF chip and measuring its responses. On the other hand, due to the use of PUF chips, physical invasive attacks, such as node cloning and node tampering, are not possible.

A. Known Session-Specific Temporary Information (KSSTI) Attack

In this section, we show that Gope et al.'s scheme is vulnerable to the KSSTI attack. In this attack, by obtaining temporary session-specific information, such as random nonces, the adversary tries to compromise the session key or other secrets [14].

Claim 1: If the nonce parameter N_t of a particular session is somehow revealed to an adversary $\mathcal{A}$, he can obtain the secret session key K_{ts} of the current session as well as all previous and next session keys.

Proof: Obtaining current session key

$\boxed{A_2} \Rightarrow \mathcal{A}$ knows N_t in i^{th} session. (S_1)

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_1 in i^{th} session $\Rightarrow \mathcal{A}$ knows N_t^*. (S_2)

$(S_1, S_2) \Rightarrow \mathcal{A}$ computes: $K_i = N_t \oplus N_t^*$. (S_3)

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_2 in i^{th} session $\Rightarrow \mathcal{A}$ knows N_s^* and R_o^{1*}. (S_4)

$(S_3, S_4) \Rightarrow \mathcal{A}$ computes $N_s = N_s^* \oplus K_i$ and $R_o^1 = R_o^{1*} \oplus K_i$. (S_5)

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_3 in i^{th} session $\Rightarrow \mathcal{A}$ knows X and $\widetilde{R_o}^*$. (S_6)

$(S_3, S_6) \Rightarrow \mathcal{A}$ computes: $\widehat{R_o^2} = X \oplus K_i$. (S_7)

$(S_1, S_5, S_7) \Rightarrow \mathcal{A}$ obtains $K_{ts} = H(N_t||N_s||R_o^1||\widehat{R_o^2})$ in i^{th} session. ■

Proof: Obtaining next session key

$(S_3, S_6) \Rightarrow \mathcal{A}$ computes: $\widetilde{R_o} = \widetilde{R_o}^* \oplus K_i \Rightarrow$
$\mathcal{A}$ knows R_o in $(i+1)^{th}$ session. (S_8)

Go to the $(i+1)^{th}$ session:

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_2 in $(i+1)^{th}$ session $\Rightarrow \mathcal{A}$ knows $N_s^* \& R_o^{1*}$. (S_9)

$(S_8, S_9) \Rightarrow \mathcal{A}$ computes $K_{i+1} = R_o^{1*} \oplus R_o^1$ and $N_s = N_s^* \oplus K_{i+1}$. (S_{10})

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_1 in $(i+1)^{th}$ session $\Rightarrow \mathcal{A}$ knows N_t^*. (S_{11})

$(S_{10}, S_{11}) \Rightarrow \mathcal{A}$ computes: $N_t = K_{i+1} \oplus N_t^*$. (S_{12})

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_3 in $(i+1)^{th}$ session $\Rightarrow \mathcal{A}$ knows X. (S_{13})

$(S_{10}, S_{13}) \Rightarrow \mathcal{A}$ computes: $\widehat{R_o^2} = X \oplus K_{i+1}$. (S_{14})

$(S_8, S_{10}, S_{12}, S_{14}) \Rightarrow \mathcal{A}$ obtains K_{ts} in $(i+1)^{th}$ session. ■

Proof: Obtaining previous session key

$(S_5) \Rightarrow \mathcal{A}$ knows $\widetilde{R_o^1}$ in $(i-1)^{th}$ session. (S_{15})

Return to the $(i-1)^{th}$ session:

$\boxed{A_1} \Rightarrow \mathcal{A}$ has eavesdropped M_3 in $(i-1)^{th}$ session $\Rightarrow$
$\mathcal{A}$ knows $\widetilde{R_o}^*$ and X. (S_{16})

$(S_{15}, S_{16}) \Rightarrow \mathcal{A}$ computes: $K_{i-1} = \widetilde{R_o^{1*}} \oplus \widetilde{R_o^1}$ and $\widehat{R_o^2} = X \oplus K_{i-1}$. (S_{17})

$\boxed{A_1} \Rightarrow \mathcal{A}$ has eavesdropped M_1 and M_2 in $(i-1)^{th}$ session $\Rightarrow$
$\mathcal{A}$ knows N_t^* and N_s^* and R_o^{1*}. (S_{18})

$(S_{17}, S_{18}) \Rightarrow \mathcal{A}$ computes: $N_t = K_{i-1} \oplus N_t^*$ and
$N_s = K_{i-1} \oplus N_s^*$ and $R_o^1 = K_{i-1} \oplus R_o^{1*}$. (S_{19})

$(S_{17}, S_{19}) \Rightarrow \mathcal{A}$ obtains K_{ts} in $(i-1)^{th}$ session. ■

Claim 2: If the nonce parameter N_s of a particular session is somehow revealed to an adversary $\mathcal{A}$, he can obtain the secret session key K_{ts} of the current session as well as all previous and next session keys.

Proof: The proof of this claim is similar to the proof of *Claim1*, so we present a condensed form of it:

$\boxed{A_2} \Rightarrow \mathcal{A}$ knows N_s in i^{th} session. (S_1)

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_1 and M_2 and M_3 in i^{th} session $\Rightarrow$
$\mathcal{A}$ knows N_t^* and N_s^* and R_o^{1*} and X and $\widetilde{R_o}^*$. (S_2)

$(S_1, S_2) \Rightarrow \mathcal{A}$ computes: K_i and N_t and N_s and R_o^1 and $\widehat{R_o^2}$. (S_3)

$(S_3) \Rightarrow \mathcal{A}$ obtains $K_{ts} = H(N_t||N_s||R_o^1||\widehat{R_o^2})$ in i^{th} session. ■

$(S_2, S_3) \Rightarrow \mathcal{A}$ computes: $\widetilde{R_o} \Rightarrow \mathcal{A}$ knows R_o in $(i+1)^{th}$ session. (S_4)

Go to the $(i+1)^{th}$ session:

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_1 and M_2 and M_3 in $(i+1)^{th}$ session $\Rightarrow$
$\mathcal{A}$ knows N_t^* and N_s^* and R_o^{1*} and X and $\widetilde{R_o}^*$. (S_5)

$(S_4, S_5) \Rightarrow \mathcal{A}$ computes: K_i and N_t and N_s and R_o^1 and $\widehat{R_o^2}$. (S_6)

$(S_6) \Rightarrow \mathcal{A}$ obtains K_{ts} in $(i+1)^{th}$ session. ■

$(S_3) \Rightarrow \mathcal{A}$ knows R_o^1 in $(i-1)^{th}$ session. (S_7)

Return to the $(i-1)^{th}$ session:

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_1 and M_2 and M_3 in $(i-1)^{th}$ session $\Rightarrow$
$\mathcal{A}$ knows N_t^* and N_s^* and R_o^{1*} and X and $\widetilde{R_o}^*$. (S_8)

$(S_7, S_8) \Rightarrow \mathcal{A}$ computes: K_i and N_t and N_s and R_o^1 and $\widehat{R_o^2}$. (S_9)

$(S_9) \Rightarrow \mathcal{A}$ obtains K_{ts} in $(i-1)^{th}$ session. ■

By the above claims, we proved that exposing only one of the nonce parameters of a particular session, reveals all past and future secrets and session keys. So, this protocol is highly vulnerable to the KSSTI attack.

B. Lack of Perfect Forward Secrecy (PFS) and Perfect Backward Secrecy (PBS)

The PFS and PBS mean that knowing the secret parameters of the current session does not compromise the security of the previous and next sessions, respectively [14]. It is shown here that Gope et al.'s scheme cannot satisfy these features. In this protocol, the secrets of a session are K_i and R_o.

Claim 3: If the secret parameter K_i of the current session is known to an individual, he can obtain the secret session key K_{ts} of the current session as well as all previous and next session keys.

Claim 4: If the secret parameter R_o of the current session is known to an individual, he can obtain the secret session key K_{ts} of the current session as well as all previous and next session keys.

Proofs: Clearly, these two claims' proofs are very similar to those presented in *Claim1* and *Claim2*. Therefore, their details don't need to be repeated. Following is an outline of these proofs:

$\boxed{A_2} \Rightarrow \mathcal{A}$ knows K_i (or R_o) in i^{th} session. (S_1)

$\boxed{A_1} \Rightarrow \mathcal{A}$ eavesdrops M_1 and M_2 and M_3 in i^{th}, $(i+1)^{th}$, and $(i-1)^{th}$ sessions. (S_2)

$(S_1, S_2) \Rightarrow \mathcal{A}$ computes: N_t and N_s and R_o^1 and $\widehat{R_o^2}$ in i^{th}, $(i+1)^{th}$, and $(i-1)^{th}$ sessions. (S_3)

$(S_3) \Rightarrow \mathcal{A}$ obtains K_{ts} in i^{th}, $(i+1)^{th}$, and $(i-1)^{th}$ sessions. ■

C. Non-Invasive Physical Attacks

Non-invasive attacks are a class of physical attacks in which an attacker seizes an unprotected device and gains control over it for a limited time. Therefore, he can read the values stored in the memories and also perform CRP measurements without changing the physical structure and behavior of PUF chips. We show that Gope et al.'s scheme failed to withstand these attacks, despite using PUF chips.

Claim 5: If an attacker $\mathcal{A}$ seizes the device *T* for a limited time and reads its memory with no physical manipulation, he can obtain the session keys K_{ts} of all previous and next sessions.

Proof:

$\boxed{A_3} \Rightarrow \mathcal{A}$ seizes the device, and reads the values stored in its memories. (S_1)

$(S_1) \Rightarrow \mathcal{A}$ knows K_{i-1} in $(i-1)^{th}$ session. (S_2)

$(S_2, Claim3) \Rightarrow \mathcal{A}$ obtains K_{ts} in i^{th}, $(i-1)^{th}$, and $(i-2)^{th}$ sessions. ■

Claim 6: If an attacker $\mathcal{A}$ seizes the device *T* for a limited time and reads its memory with no physical manipulation, he can obtain the permanent secret R_w.

Proof:

$\boxed{A_3} \Rightarrow \mathcal{A}$ seizes the device, and reads the values stored in its memories. (S_1)

$(S_1) \Rightarrow \mathcal{A}$ knows C_w. (S_2)

$\boxed{A_3} \Rightarrow \mathcal{A}$ gains control of the device and performs CRP measurements using the WPUF chip. (S_3)

$(S_3) \Rightarrow \mathcal{A}$ obtains R_w. ■

Since the parameter R_w is a permanent secret in the protocol and never changes, this makes up a single point of failure for the system.

D. Replay and Denial of Service (DoS) Attacks

In a DoS attack, an attacker may want to disrupt the normal operation of the protocol by sending invalid or replayed messages to a participant.

Claim 7: The Gope et al.'s protocol is vulnerable to replay and DoS attacks

Proof:

$\boxed{A_1} \Rightarrow \mathcal{A}$ intercepts M_1 in the transmission path. (S_1)

$\boxed{A_1} \Rightarrow \mathcal{A}$ replays M_1 to the server *S*. (S_2)

$(S_1, S_2) \Rightarrow ID_t$ is valid for the server $\Rightarrow$ The server accepts M_1 as a valid message. (S_3)

$(S_3) \Rightarrow$ The server performs all the calculations in Step(4) of the protocol, and sends M_2 to the device *T*. (S_4)

$\boxed{A_1} \Rightarrow \mathcal{A}$ intercepts M_2 in the transmission path. (S_5)

$(S_5) \Rightarrow$ The device does not receive M_2, so the protocol is terminated. (S_6)

$(S_6) \Rightarrow$ The server does not update ID_t and its current value remains a valid value. (S_7)

$(S_7) \Rightarrow \mathcal{A}$ can repeatedly replay the M_1 with this scenario to the server. (S_8)

$(S_4, S_8) \Rightarrow \mathcal{A}$ can engage the computing and communication resources of the server and perform a DoS attack. ■

E. Weakness in Mutual Authentication Process

Due to OPUF's noisy behavior, the mutual authentication in Gope et al.'s scheme is achieved only based on the verification of one-half of the R_o on each side. So, the effective length of R_o reduces by half, and the computational complexity of the brute-force search for R_o decreases exponentially. This is a major weakness in the protocol.

Claim 8: If an adversary $\mathcal{A}$ obtains only one part of R_o, it can impersonate the server or device.

Proof:

$\boxed{A_2} \Rightarrow \mathcal{A}$ (as an illegal server) knows only R_o^1. (S_1)

$\boxed{A_1} \Rightarrow \mathcal{A}$ sends R_o^1 (Embedded in M_2) to the device *T*. (S_2)

$(S_2) \Rightarrow T$ Checks $FHD(\widehat{R_o^1}, R_o^1) < \Delta$ and proceed if it is true. (S_3)

$\boxed{A_1}, (S_3) \Rightarrow T$ sends $\widehat{R_o^2}$ (Embedded in M_3) to $\mathcal{A}$. (S_4)

$(S_4) \Rightarrow \mathcal{A}$ proceeds the protocol without checking $FHD(\widehat{R_o^2}, R_o^2) < \Delta$. (S_5)

$(S_5 \Rightarrow \mathcal{A}$ can impersonate the server. ■

We can give similar proof for the case where $\mathcal{A}$ knows only R_o^2 and wants to impersonate the device.

IV. Suggested Countermeasures

Based on the claims presented in Section III, we make suggestions to resolve the security issues of this protocol.

1-When implementing lightweight authentication protocols, the XOR operation is extremely useful. However, if not handled properly, it can seriously compromise the protocol's security. Gope et al.'s scheme hides several sensitive session parameters by XORing them with the same secret parameter K_i. This fact causes vulnerabilities in this protocol, according to the proofs presented for Claims 1 and 2. To resolve this issue, one-way hash functions can be used to generate different secret parameters from K_i, which can then be used in XOR operations.

2- In order to support PFS and PBS features in a protocol, it is essential to have a well-designed update procedure for session parameters. In Gope et al.'s scheme, the secret parameter K_i of the i^{th} session is explicitly used to hide the next session's secret credentials during transmission. According to the discussion presented to prove Claims 3 and 4, this fact causes the PFS feature to be violated. Applying a one-way hash function to K_i during the update procedure can solve this problem.

Increasing the number of secret parameters in each session and appropriately using them in the update procedure can effectively resolve the PBS issue. As shown in proofing Claims 3 and 4, due to the improper protocol design, the parameter R_w and the server's master key *MK* do not participate in updating the parameters. Therefore, modifying the protocol update procedure and involving new secret parameters can resolve this issue.

3- According to Claims 5 and 6, in Gope et al.'s scheme, WPUF fails to improve the protocol's physical security level in practice. Improper use of the response R_w, assuming it as a permanent secret, and explicitly storing challenge C_w in the device's memory are the main reasons for this issue. A first suggestion would be to send C_w to the device from the server each time rather than storing it in the device's memory. A second suggestion is to use R_w effectively in the authentication process and also update it after each session.

4- Based on proof of Claim 7, using the current time in the protocol can prevent the replay and DoS attacks.

5- To solve the weakness mentioned in Claim 8, the FHD should be computed at both parties using all bits of R_o, which, on the other hand, makes cheating possible between parties. Effective use of the bit commitment method [15] is a suggested solution to this issue.

V. Conclusion

In this letter, we analyzed a recent PUF-based authentication protocol presented by Gope et al. We presented a formal framework based on propositional logic to investigate security issues associated with this protocol. It is, of course, possible to utilize this framework for other similar protocols as well. Our analysis was then used to develop suggestions for resolving protocol vulnerabilities.

References

[1] P. Mall, R. Amin, A. K. Das, M. T. Leung, and K.-K. R. Choo, "PUF-based authentication and key agreement protocols for IoT, WSNs, and smart grids: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8205–8228, Jun. 2022.

[2] P. Gope, O. Millwood, and B. Sikdar, "A scalable protocol level approach to prevent machine learning attacks on physically unclonable function based authentication mechanisms for Internet of Medical Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1971–1980, Mar. 2022.

[3] A. M. A. Modarres and G. Sarbishaei, "An improved lightweight two-factor authentication protocol for IoT applications," *IEEE Trans. Ind. Informat.*, vol. 19, no. 5, pp. 6588–6598, May 2023.

[4] G. Bansal and B. Sikdar, "Location aware clustering: Scalable authentication protocol for UAV swarms," *IEEE Netw. Lett.*, vol. 3, no. 4, pp. 177–180, Dec. 2021.

[5] K. Lounis and M. Zulkernine, "Lessons learned: Analysis of PUF-based authentication protocols for IoT," *Digit. Threats Res. Pract.*, to be published. [Online]. Available: https://dl.acm.org/doi/10.1145/3487060

[6] Y. Cho, J. Oh, D. Kwon, S. Son, J. Lee, and Y. Park, "A secure and anonymous user authentication scheme for IoT-enabled smart home environments using PUF," *IEEE Access*, vol. 10, pp. 101330–101346, 2022.

[7] M. Adeli, N. Bagheri, H. Martín, and P. Peris-Lopez, "Challenging the security of 'A PUF-based hardware mutual authentication protocol,'" *J. Parallel Distrib. Comput.*, vol. 169, pp. 199–210, Jul. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731522001538

[8] C. Gu, C.-H. Chang, W. Liu, S. Yu, Y. Wang, and M. O'Neill, "A modeling attack resistant deception technique for securing lightweight-PUF-based authentication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1183–1196, Jun. 2020.

[9] L. Kraleva, M. Mahzoun, R. Posteuca, D. Toprakhisar, T. Ashur, and I. Verbauwhede, "Cryptanalysis of strong physically unclonable functions," *IEEE Open J. Solid-State Circuits Soc.*, vol. 3, pp. 32–40, 2023.

[10] M.-D. Yu, M. Hiller, J. Delvaux, R. Sowell, S. Devadas, and I. Verbauwhede, "A lockdown technique to prevent machine learning on PUFs for lightweight authentication," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 3, pp. 146–159, Jul.-Sep. 2016.

[11] Y. Gao, S. F. Al-Sarawi, D. Abbott, A.-R. Sadeghi, and D. C. Ranasinghe, "Modeling attack resilient reconfigurable latent obfuscation technique for PUF based lightweight authentication," 2017, *arXiv:1706.06232*.

[12] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.

[13] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2001, pp. 453–474.

[14] C. Boyd, A. Mathuria, and D. Stebila, *Protocols for Authentication and Key Establishment* (Information Security and Cryptography). Berlin, Germany: Springer, 2019. [Online]. Available: https://books.google.com/books?id=T8-8DwAAQBAJ

[15] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Hoboken, NJ, USA: Wiley, 2007.