QR-PUF: Design and Implementation of A RFID-based Secure Inpatient Management System Using XOR-Arbiter-PUF and QR-Code

Prosanta Gope *Senior Member, IEEE*, Yuening Wang, Zengpeng Li and Biplab Sikdar *Senior Member, IEEE*

***Abstract*—In recent years, Physical Unclonable Functions (PUFs) have played a major role in providing low-cost physical security for IoT devices such as Radio Frequency Identification (RFID) tags. PUFs take advantage of the physical properties of the device to build unique security primitives that can be used by authentication mechanisms. Meanwhile, the security and convenience of QR codes for device authentication on mobile devices has been widely recognized. The point-to-point communication makes it less vulnerable to interception and analysis by adversaries. In this article, we propose a *new* RFID-based secure inpatient management system for identifying a legitimate patient. Our proposed system uses an XOR Arbiter PUF to generate a secret key-stream and then uses the key-stream to construct a secure QR code for secure identification. Also, since PUFs are vulnerable to machine learning attacks, we propose a modeling attack resilience framework to enhance the security of the proposed protocol. Security analysis of the proposed scheme using ProVerif shows that the scheme is effective against a variety of imperative attacks on RFID devices. To show the applicability of the proposed scheme, we also provide a case study of an inpatient management system in hospitals.**

***Index Terms*—Inpatient management system, XOR Arbiter PUF, QR code, RFID, ProVerif.**

I. Introduction

With the rapid development of mobile devices and IoT systems, the interconnection between smart devices is increasing substantially. On the other hand, in many application scenarios, we need to identify the devices and their movement using advanced technologies such as Radio Frequency Identification (RFID) and QR-Code [1], [2], [3]. RFID technology links an electronic tag and a wireless signal reader through radio signals. When the tag enters the magnetic field and receives the special radio frequency signal sent by the reader, it can transmit the radio frequency through the antenna and send the information stored in its chip with the energy obtained by the induced current. RFID technology is able to penetrate non-metallic or non-transparent materials such as paper, wood and plastic. The versatility and convenience of RFID technology makes it widely used for authentication and product tracking. For instance, RFID systems are used in the fields of logistics and transportation. With the development of internationalization and e-commerce, customers' demands for high level services and competitive prices of logistics distribution is increasing. The traditional logistics systems consume a lot of human resources during storage and sorting, among other areas. Manual operations also lead to lower accuracy and higher loss of goods. However, with the help of RFID tags, goods can be automatically stored by the intelligent management system. When the goods pass through the reader at the exit of the warehouse, the system can also automatically assign them to specific transportation routes. Through the RFID system, the inventory and management centers are closely linked together, which minimizes the occurrence of errors and greatly saves manpower. However, because of contactless communication in RFID systems, the information security of users would be challenged if malicious devices interfere with the communication system. Therefore, the security of RFID devices and associated communication technology becomes important. In addition, due to the weak computing power of the lightweight RFID tag, attackers can launch a side-channel or invasion attack directly on the devices themselves. This allows attackers to simulate device tags to access private data from the server. In order to resolve these issues, the concept and technology of Physical Unclonable Functions (PUF) has been introduced. A PUF refers to the input to a physical entity and the resulting output of an unpredictable response, that exploits the random differences inherent in the device's physical structure. PUFs bring new security features to RFID technology.

P. Gope, Y. Wang are with Department of Computer Science, University of Sheffield, Regent Court, Sheffield S1 4DP, United Kingdom. (E-mail: prosanta.nitdgp@gmail.com/p.gope@sheffield.ac.uk) Z. Li is is with the School of Cyber Science and Technology, Shandong University, China. (E-mail: zengpeng@email.sdu.edu.cn) B. Sikdar is with the National University of Singapore. (E-mail: bsikdar@nus.edu.sg)

This work was supported in part by the Singapore Ministry of Education Academic Research Fund Tier 1 under Grant R-263-000-D63-114.

Corresponding author: Dr. Prosanta Gope

A. RFID-based Inpatient Management System

In general, inpatient information management in a hospital often consumes a considerable amount of time and manpower. According to authoritative data from the US Department of Health and Human Services and the US Agency for Healthcare Research and Quality from 2000 to 2008 [4], there were around 250,000 people in the United States who lose their lives each year because of improper care by medical staff. In 2009, the World Health Organization (WHO) publication regarding patient safety showed that about 10% of patients in developed countries admitted having suffered medical care errors or medical adverseness [5]. Some of these medical errors resulted from patients being wrongly identified and treated when the working areas were understaffed, which was avoidable. In order to solve such problems, many hospitals adopt digital

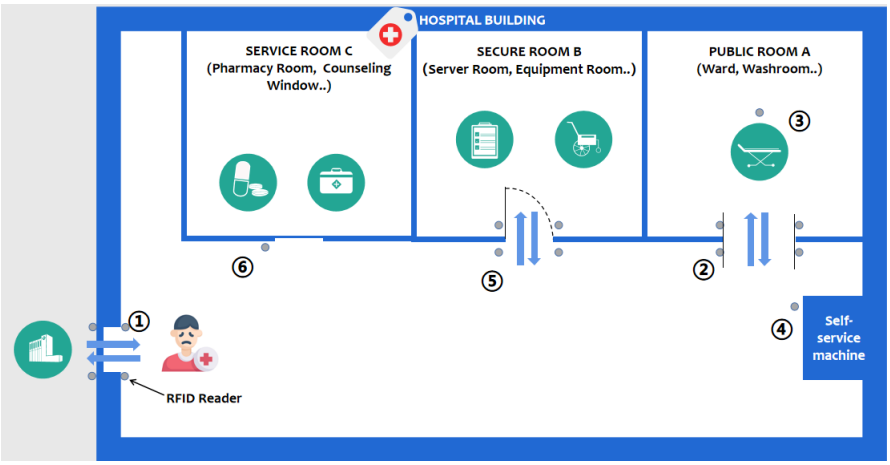

Figure 1. Mobile-RFID-based inpatient management system.

patient data management. However, the large-scale use of information and communications technology (ICT) has led to an increased risk of healthcare information breach. According to the healthcare breach report published by Bitglass in 2021 [6], the number of healthcare breaches that occurred in 2020 in the US reached 599. Hacking and information technology (IT) incidents accounted for 67.3% of the breaches, more than three times that of the next highest category. Besides, the number of breaches due to hacking sharply increased from less than 100 in 2014 to more than 400 in 2020. Obviously, information security is an urgent problem to be solved in order to ensure the secure management of patient information. Based on the authentication scheme proposed in this paper, we aim to build a secure inpatient management system (as shown in Fig. 1), where it is assumed that each user (such as a patient or medical staff) needs to manage a valid Mobile-RFID device (i.e., a RFID tag attached with a mobile device) to enter into different buildings/rooms of the hospital. The RFID-readers (installed in different entrances), attached with a server, determine whether the device (Mobile-RFID) is valid for their system and maintains a record for each user's movement in the server's database. Figure 1 shows a case study of a hospital building, where a user can move into different rooms (Room A, Room B, and Room C) of the building once he/she has a valid Mobile-RFID-device. The server attached with the readers will maintain all the information about the user's movement.

B. Security threats on RFID system

The safety risk of RFID systems comes mainly from the design idea of "system opening" [3]. The purpose of RFID design and application is to reduce the cost and improve efficiency, which reduces the scope for implementing adequate security measures. RFID security problems usually occur in various links such as data acquisition, data transmission, data processing and data storage. The common attacks are as follows.

Side-channel attack: The attackers analyze the power consumption leakage information or electromagnetic leakage information of the device to obtain the keys used in the communication between tag and the server. With access to the key, attackers can not only get information stored locally in devices but also access the private data in the system through communication with the server [7].

Counterfeiting attack: In a RFID system, the electronic tag communicates with the reader through a wireless signal. During the process of identity verification, the data interaction is completed through the radio frequency channel. After the reader, writer, and the tag transmit the identity information, they identify the part of the opposite party, and then proceed to the related operations after passing the verification. Since radio frequency signals are exposed in the air, any information transmitted during the communication between the two can be intercepted [3]. When attackers intercept the identity information of a legitimate user, they can use this information to impersonate the legitimate user in order to authenticate and make changes to the system.

De-synchronizing attack: In most RFID communication protocols, both server and RFID tag update their private secure information in every round. In this case, if attackers block the response from the server to the tag, it would cause the server to update while the tag does not. This makes the protocol inoperable.

Man-in-middle attacks: This attack is similar to the "replay attack" [8]. Attackers can conduct or control a communication with a legitimate tag or reader to authenticate the communication, and then forge the authenticated tag to proceed to the next stage of the transaction. This kind of attack can spoof the attacked system, anti-track, without exposing the attackers or others from organizing large-scale attacks. **Tracking attack**: In this attack, the adversary captures characteristic data of the tag by sending some simple querying commands and extracting unique information that can identify a tag (e.g., tag ID) by analyzing the data. The attacker can then track the carrier of the tag to invade the trade secrets or personal privacy [9].

Forward and backward threats: For some insecure RFID protocols, if attackers get the key information in the current communication from the tag (e.g., by brute force attack), then they can infer the backward or forward secret keys to access

the previous or future communications.

Machine-Learning (ML) Attacks on PUF: ML attacks on PUFs assume that an adversary can obtain and store a large number of exchanged CRPs, and then feed those CRPs into a machine learning algorithm in order to find the relation between challenges and responses. A successful ML attack provides the adversary an ability to predict the response for a future challenge. It allows the adversary to impersonate as a legal device to communicate with the server.

C. Related Work and Our Contribution

Because IoT devices are usually low-cost and resource-intensive, traditional authentication protocols for wireless networks cannot adapt to the application environment of IoT systems. In the past years, a number of anonymous authentication schemes based on PUFs have been proposed to protect privacy. In 2008, Bringer et al. [10] proposed a PUF-based authentication scheme. However, Sadeghi et al. [11] showed that it cannot guarantee the security against DoS attacks and counterfeiting attacks. In 2012, Kardas et al. [12] proposed a new RFID protocol using PUFs, but it cannot ensure forward secrecy and resilience against DoS attacks.

Recently with the development of machine learning, modelling attacks [13] have become a new threat to security protocols. In order to enhance the resilience against modeling attacks, Yu et al. [14] provided two security protocols. They restricted the number of CRPs that could be transported through the system to prevent attackers from acquiring large amounts of CRPs for modeling. Besides, to prevent attackers from measuring noise for modeling, their protocols do not allow repeated challenges. However, because of these measures, their protocols are only applicable to low density authentication, which means they are not suitable for systems that need to transfer data quickly and frequently, such as self-driving.

In 2019, Liang et al. [15] proposed a double PUF-based RFID protocol using the training model in [16]. The protocol is designed to challenge the PUF by both the verifier and the prover and ensured the machine-learning resilience. However, the protocol cannot ensure security against many other attacks such as counterfeiting attacks and replay attacks.

In this paper, we propose a PUF-based authentication protocol for Mobile-RFID systems. To develop the protocol, we use pypuf [17] to model an XOR Arbiter-PUF inside the RFID devices and extract the PUF output to generate a secure QR-Code through a the mobile-device attached with the RFID-Tag. In this way, the tag proves its legitimacy to the server. The security of the system has been analyzed and tested comprehensively by using the standard ProVerif platform. Finally, in this article we also propose a obfuscation method for resilience against ML-attacks on PUFs.

The rest of the paper is organized as follows: Section II provides a brief introduction to PUFs and QR codes. Section III presents and explains the two phases of our proposed protocol. Security analysis of the model is presented in Section IV. Section V present implementation details of the proposed scheme to prove that the proposed scheme is secure and efficient. Finally, Section VI concludes the paper.

Table I
SYMBOLS AND CRYPTOGRAPHIC FUNCTIONS

Symbol	Definition
T_X	The identity of tag T itself
$Temp_{T_X^i}$	The temporary identity of tag T for i-th round
$CRP(C_i, R_i)$	Challenge-response pair for i-th round
K_i	Session key for i-th round
P_T	Secure physical uncloneable functions for tag T
$h(.)$	One-way hash function
$\bigoplus$	Exclusive-OR operation
$\|$	Concatenation operation

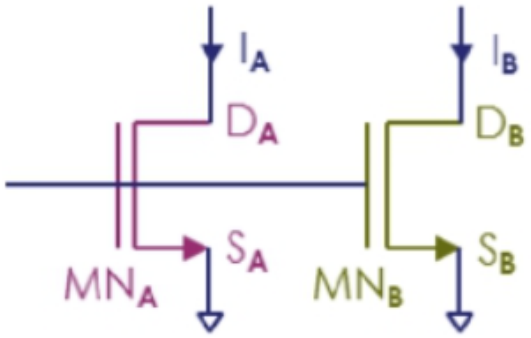

Figure 2. A simple PUF element

II. PRELIMINARIES

A. Physical Unclonable Function

The operating principle of a PUF is based on the uniqueness of the microscopic physical structure of the device that is caused by subtle variations during the industrial production process. In practice, the PUF system provides a specific response to a given challenge, called a challenge-and-response-pair (CRP). For example, Fig. 2 shows a simple PUF element which only has two transistors A and B which are produced using the same design and fabrication process. In reality, due to various uncontrollable, minute physical variations during manufacturing, A and B have slightly different electrical properties such as their threshold voltage. We denote the property as P, and set a rule that denotes cases when P_A is larger than P_B as 1 and the converse as 0. By combining a bunch of these elements, a PUF system is created. The PUF will output a random but unique binary string as a response to a certain electrical signal (or challenge). The response can be used in cryptographic systems as, for example, a private key to generate public keys, a key to encrypt other keys, or identification for devices [18], [19].

1) PUF Classification: In general, PUFs can be divided into two categories: strong PUFs and weak PUFs [20]. The terms "strong" and "weak" are not directly connected to the security against attacks. Strong PUFs are complete systems capable of operating independently and have a complicated underlying structure that is capable of generating a large

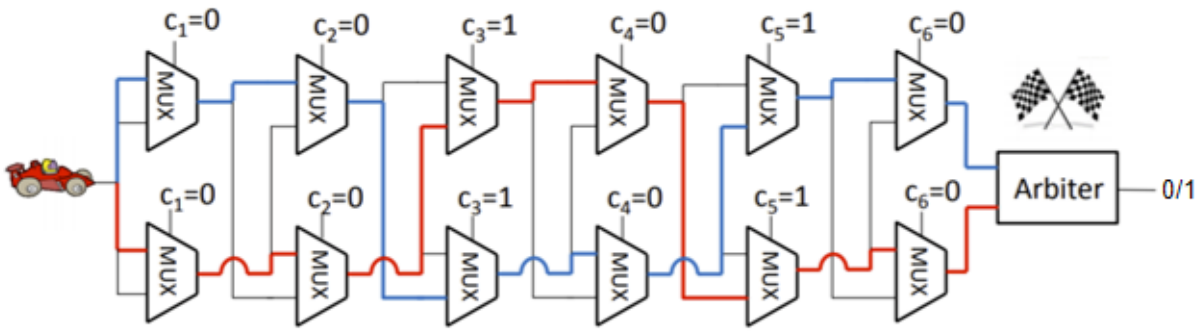

Figure 3. An example of Arbiter PUF.

number of CRPs. Strong PUFs have three main features [21]: (i) The physical structure must be impossible to be cloned or imitated (with indistinguishable behavior), making strong PUFs more challenging for manufacturers; (ii) As adversaries attack PUFs by measuring and analyzing CRPs, a complete measurement of all CRPs on strong PUFs must be impossible to be completed in limited time; (iii) Even with a large number of other known CRPs, the response of a strong PUF to a new challenge should still be difficult to predict.

Compared to strong PUFs, weak PUFs do not work alone as a complete security system but only deal with a few or even a single CRP, serving as a random key or "entropy" to a cryptosystem [22]. In many cases, this way of application makes weak PUFs more stable and adaptable than strong PUFs. Also, manufacturers do not have to focus as much on designing the physical structure of the system itself, which significantly reduces the cost of using PUFs. Developers are only required to design an appropriate way of encryption based on the characteristics of the PUF.

2) Arbiter PUF: Arbiter PUF is a PUF system based on circuit delay. As Fig. 3 shows, it is made up of two identical circuit layouts, many multiplexers, and an arbiter [14]. Two race signals propagate on these two layouts. Every multiplexer responds to a challenge to control the signal path. For example, when the challenge bit is 0, signals go straight, otherwise they cross over. When these two signals get to the arbiter, the response is generated depending on which signal is faster. Let ΔD be the delay difference between the top and bottom layouts. Ignoring the environmental variations like temperature, circuit changes and so on, the arbiter PUF may be modeled as a scalar multiplication as:

$$\Delta D = \vec{w} \cdot \vec{\Phi}$$

where $\vec{w}$ represents the vector of stage delays (the total signal delay difference from a multiplexer to the next multiplexer, i.e., a stage) and $\vec{\Phi}$ is the challenge vector whose elements take on values of 1 and -1 (correponding to challenge bits 1 and 0).

In this model, we can observe that the basic arbiter PUF is a linear function of the challenge. Such behavior is not secure against modelling attacks [23].

3) XOR Arbiter PUF: Arbiter PUFs may be strengthened by using a XOR arbiter PUF [24], [25]. There are two ways to design a XOR arbiter PUF based on the basic design in Fig. 3. The first one puts a challenge vector into multiple arbiter PUFs and does an XOR on the results to generate a single response. It transforms the model from an addition to a continued product, with more obfuscated stage delays to make it more secure. The second approach provides every arbiter PUF with an individual challenge vector. This makes the system much more complicated and thereby more secure against modeling attacks.

4) Formal Definition of PUF: To formally define and describe the security of PUF, we first introduce PUF class [26], denoted by P. P provides a creation function $P.Create()$ to set up a PUF instance puf_i. Since $P.create()$ is often a probabilistic function, we provide a randomized input $r^C = rand\{0,1\}$ to it. Thus, the class P can be described as a collection of all instances created:

$$\mathrm{P} \equiv \left\{puf_i \leftarrow P.Create(r_i^C) : \forall i, r_i^C = rand\{0,1\}\right\}$$

.

From a practical point of view, PUF class P represents the structure design of the PUF system and the PUF instance puf can be considered as the exact produced physical structure. Many PUF constructions are designed to be reconfigurable and can be modified by an external input. This makes a PUF system harder to analyze. Therefore, when required, we define a PUF instance with a variable parameter as $puf(x)$. Here, x represents what was called "challenge" above. The set of challenges which can be applied to an instance of a class P is denoted as x_P.

For a PUF instance, P also provides an evaluation function denoted as $puf.Eval()$ which generates a measurement result of the instance. Similarly, when the class P is reconfigurable, we write the evaluation function as $puf(x).Eval(r^E = rand\{0,1\})$. The measurement here represents the "response" of a PUF system. The collection of all responses that can be produced by class P is denoted as y_P.

For convenience, we abbreviate some statements as follows:

$$puf_i \leftarrow P.Create(r_i^C = rand\{0,1\}) \Rightarrow PUF \leftarrow P,$$

$y(x) \leftarrow puf(x).Eval(r^E = rand\{0,1\}) \Rightarrow Y(x) \leftarrow PUF(x).$

To sum up, a PUF can be formally defined as:

Definition 1. *A hardware device is called "PUF" if: (i) The instance $PUF \leftarrow P$ is deterministic for a specific set of challenges x_P and can be evaluated with each x at least once; (ii) The value of $Y(x) \leftarrow PUF(x)$ changes with different $x \in x_P$, i.e., $Y(x)$ is not a constant function.*

To define a "secure PUF", we introduce "attack models" [27] to analyze common attack environments. First, we assume that attackers have physical access to a PUF for only a limited time. After the access, the attacker tries to predict $Y(x) \leftarrow PUF(x)$ for a random $x \in x_P$. If the attacker has no knowledge of x_P and all other secret information generated in the lifetime or any other further information of the individual PUF device, then the attack model is called "outsider attack". If the attacker has no knowledge of x_P and all other secret information but has all other information as the manufacturer has, then the model is called "insider attack". Generally speaking, the software design of a PUF system is against outsider attacks. We define a secure PUF as:

Definition 2. *A PUF is secure if an outsider attacker can compute or physically copy the function $Y(x) \leftarrow PUF(x)$ for not more than a negligible fraction of challenges from x_P.*

Here, "compute" represents a computation independent from the PUF itself, corresponding to "mathematical cloning" while "physical copy" means the creation of a new device to duplicate the physical structure of the PUF, corresponding to "physical cloning".

B. Mobile RFID

Ordinary RFID structure often has a fixed reader and objects equipped with RFID tags, for example, smart cards. However, in a mobile RFID (M-RFID) system, the readers or interrogators cam be installed in mobile devices such as a mobile phone or a handheld scanner. Compared with traditional RFID, the main advantage of M-RFID is portability. It can cover a large area with a small number of mobile readers, which is more suitable when frequent identification is required.In addition, with hardware and software support, mobile devices can be both tags and readers. In this case, M-RFID can be involved in mobile telecommunication services. One of the implementations is Near-Field Communications (NFC) protocol that provides low-speed connection between two devices over a distance of 4 cm or less. M-RFID based on telecommunication can also be applied in information retrieval, data transmission, automated messaging, voice services, mobile payment and other fields.

C. QR Code

Unlike RFID which uses wireless communication, QR code is an end-to-end close-range visual recognition technology, which is widely used in mobile devices, especially smart phones. For example, according to a report from IResearch, QR code transactions in the Chinese market exceeded 10 trillion in the third quarter of 2020 [28]. The widespread use of QR codes is closely associated with their greater convenience. QR codes have evolved from the traditional one-dimensional bar code to a two-dimensional form that can carry more complex information with a positioning system, error correction mechanism, and unified version form. The structure of a version 2 QR code is shown in Fig. 4. There are function patterns spread over the code which are used to provide the location to the scanner. Also, there are places to store the format and version information. All the other places are filled with data code and error correction (EC) code. The reader can scan the QR code through the camera and read the information according to a number of geometric shapes corresponding to the binary data. The EC is an error correction mechanism based on Reed-Solomon codes which can generate a polynomial from a long code and can correct back the errors of the code itself. There are four error correction code levels corresponding to the ability to restore 7%, 15%, 20% and 30% of code words. Based on the EC mechanism, there are encryption solutions that write the confidential information into the EC [29]. In such cases, data code-words are replaced with errors from which secret messages can be recovered.

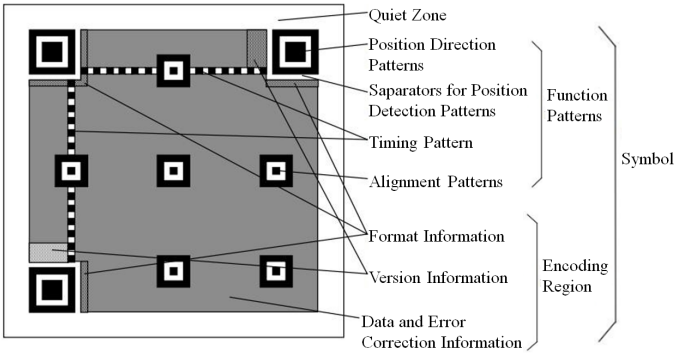

Figure 4. The structure of version 2 QR code.

Compared to RFID devices, QR codes lose some of the communication capability but gain a lighter and more secure user experience. In recent years, both RFID and QR code technologies have been used in logistics transportation, mobile commerce, security management, and other applications [30], [31], [32].

III. Proposed Scheme

The overall authentication process of the proposed scheme is divided into two parts: *setup phase* and *authentication phase*. Figs. 5 and 6 present the flow diagram and details of setup phase, respectively, and Figs. 7 and 8 present the flow diagram and details of the authentication phase. The line between the Mobile-RFID tag and the server (authenticator) represents the data communication between them, while the blocks show the actions inside each of these entities.

A. Setup Phase

The setup phase consists of following steps (all of the values are generated for authentication round 1):

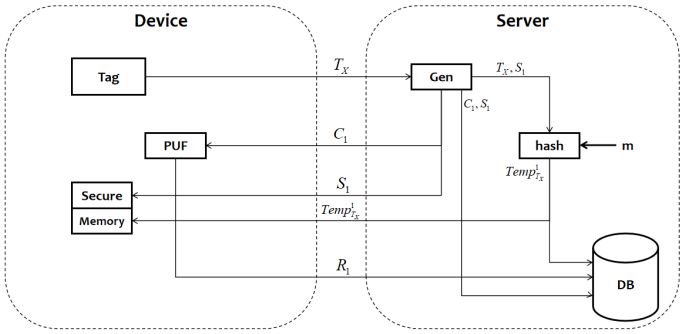

Figure 5. The flow chart of setup phase for the proposed scheme.

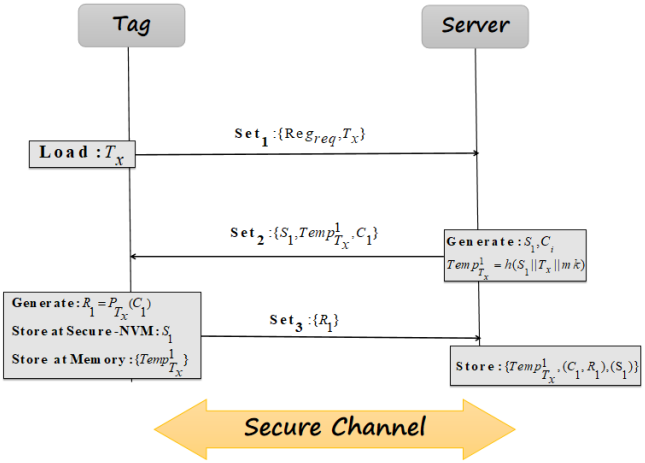

Figure 6. Details of the setup phase for the proposed scheme.

Step 1: The tag loads its tag ID T_X from the device and sends a registration request $Set_1 : \{Reg_{req}, T_X\}$ to the server through a secure channel to start the setup phase.
Step 2: The server receives Set_1 and generates a secret S_1 and a challenge C_1. Then, the server also generates a temporary ID, $Temp^1_{T_X} = hash(S_1||T_X||mk)$, where mk is the master key that serves as the unique ID of a server. After that, the server sends $Set2 : \{S_1, Temp^1_{T_X}, C_1\}$ to the tag.
Step 3: The tag inputs C_1 into its PUF for extracting a response R_1 ($R_1 = P_{T_X}(C_1)$), and then sends R_1 to the server, and finally stores the secret S_1 in a secure non-volatile-memory (NVM) and also stores $Temp^1_{T_X}$ in its physical memory.
Step 4: On the other hand, after receiving S_2, the server stores $Temp^1_{T_X}$, the CRP (C_1, R_1), and S_1 in its database. Note that unlike some other protocols such as [8] and [35], the setup phase of the proposed scheme does not need to build a large CRP database.

B. Authentication Phase

The i-th round of the authentication phase consists of following steps:
Step 1: The tag uses the temporary tag ID $Temp^i_{T_X}$ from its memory, constructs a message $MSG_1 : \{Req_{Entry}, Temp^i_{T_X}\}$, and then sends message MSG_1 to the server.
Step 2: Upon receiving message MSG_1, the server extracts the corresponding secret S_1 and the CRP (C_1, R_1) from its database based on $Temp^i_{T_X}$. Then, the server computes a temporary round key $K_i = hash(T_X||S_i)$ which is valid only for the current session/round. Next, the server encrypts C_i with K_i as $\Delta_i = E_{K_i}[C_i]$, uses Δ_i and K_i to compute $H_i = hash(\Delta_i||K_i)$, constructs a message $MSG_2 : \{\Delta_i, H_i\}$, and then sends MSG_2 to the tag attached with the mobile device.
Step 3: After receiving message MSG_2, the tag computes K_i by using the secret S_i loaded in Step 1 as $K_i = hash(T_X||S_i)$. After decrypting the message, the tag first extracts the PUF response R_i based on the challenge C_i and encrypts R_i with the round K_i, i.e., $X = E_{K_i}[R_i]$. Then, it converts X into a secure QR code X-QR using the mobile device attached with the tag. Based on the security credentials of the i-th round, the tag then generates a set of new credentials for the $(i+1)$-th round, i.e., $S_{i+1} = hash(S_i)$, $C_{i+1} = hash(C_i||S_{i+1})$, $K_{i+1} = hash(T_X||S_{i+1})$, and $Temp^{i+1}_{T_X} = hash(Temp^i_{T_X}||S_{i+1})$. After that, the tag extracts a PUF response R_{i+1} corresponding to the challenge C_{i+1}, and then computes $Y = E_{K_{i+1}} = [R_{i+1}]$. Finally, the mobile-RFID tag computes $H_{i+1} = hash(X||K_i||Y)$ and sends $MSG_3 : \{X - QR, Y, H_{i+1}\}$ to the server.
Step 4: After receiving message MSG_3, the server first scans the QR code X-QR, an then verifies the parameter H_{i+1}. If the verification is successful, the sever verifies the parameter X obtained from the QR code. Then, the server generates a set of security credentials for the $(i+1)$-th round, i.e., $S_{i+1} = hash(S_i)$, $C_{i+1} = hash(C_i||S_{i+1})$, $K_{i+1} = hash(T_X||S_{i+1})$, and $Temp^{i+1}_{T_X} = hash(Temp^i_{T_X}||S_{i+1})$. After that, the server decrypts Y to get R_{i+1} and stores all the new security credentials $\{Temp^{i+1}_{T_X}, (C_{i+1}, R_{i+1}), S_{i+1}\}$ in its database.

The authentication phase of our scheme ensures secure communication with forward security support, since most of the security credentials in the proposed scheme are used only once (for the current session). The security credentials for the next round are generated separately by the participants (device and server). Therefore, if an adversary obtains and infers the communication data in one round, they cannot infer either backward and forward communication, which provides a higher-level of security. In the proposed scheme, we assume the ideal-PUF conditions. However, the proposed scheme will also work in noisy PUF conditions. In such scenarios, we would need some error correction techniques such as fuzzy extractor, Hamming codes, etc. These details have been provided in our another work [35].

C. Proposed Approach for ML-attack Resilience

In modelling attacks, attackers are generally required to have access to a significant numbers of CRPs. In addition, in any modelling attack, an adversary closely observes the inevitable correlation between the input and output. Therefore, if the correlation can be obfuscated through appropriate

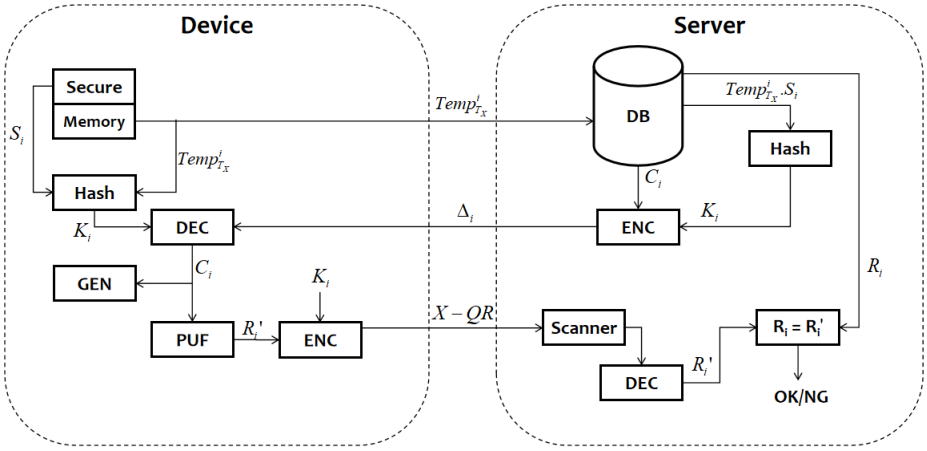

Figure 7. Flow chart of authentication phase for the proposed scheme.

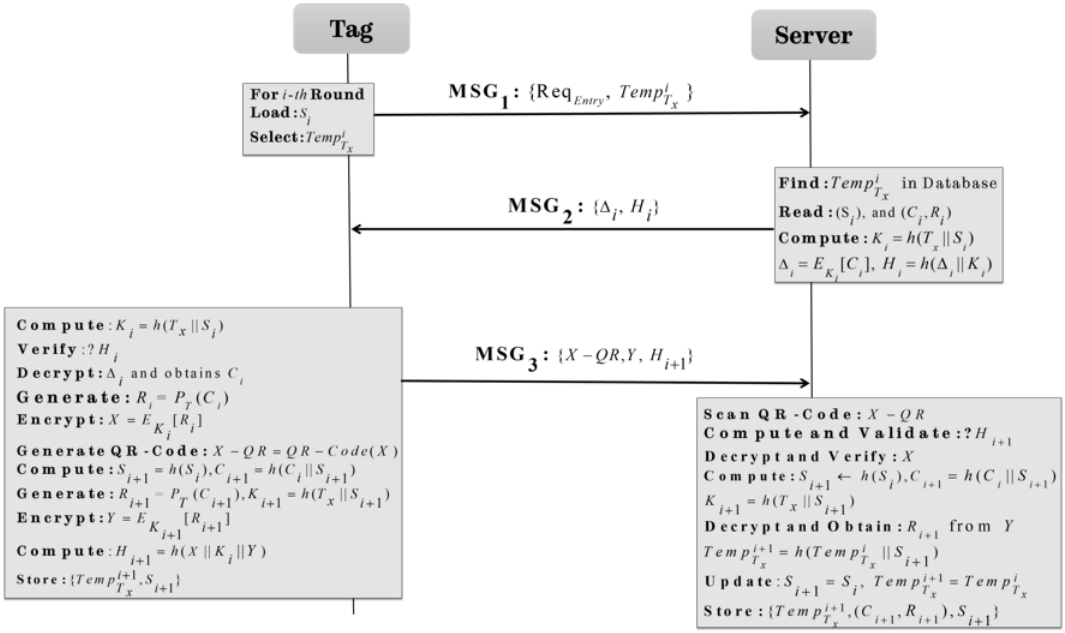

Figure 8. Details of the authentication phase for the proposed scheme.

methods, then modelling attack can be prevented. To improve the resilience of a strong PUF (such as an arbiter PUFs or any linear delay PUFs) against ML attacks, we propose a new obfuscation method/resilience framework that can enhance the security of the above protocol. The proposed framework can be divided into two phases: LFSR and Swap and 1's complement (SaC) obfuscation phase and the recycling shift phase, and assumes that the two phases operate within a closed environment. In the LFSR and SaC obfuscation phase, the original challenge C is go through a new obfuscation method LFSR and create an obfuscated challenge and then the obfuscated challenge c_{new} will be input into the recycling phase for further obfuscation and generated final obfuscated challenge f_c that will eventually output the response $r^{'}$. In this regard, after going through the LFSR obfuscation block, the obfuscated challenge is then divided into two parts and the position of the first half of the challenge is swapped with the second half of the challenge to generate a swapped challenge $C^{'}$. Then 1's complement is performed on the swapped challenge $C^{'}$ to create the obfuscation challenge C_{new}. Finally, the obfuscation challenge C_{new} is input into the recycling phase to generate the final obfuscated challenge f_C that will eventually output the response $R^{'}$. In the recycling shift phase, we calculate Hamming weight (i.e., the number of non-zero bits) of the challenge to determine the shifting bits. The advantage of using Hamming weight is that it is not a pre-

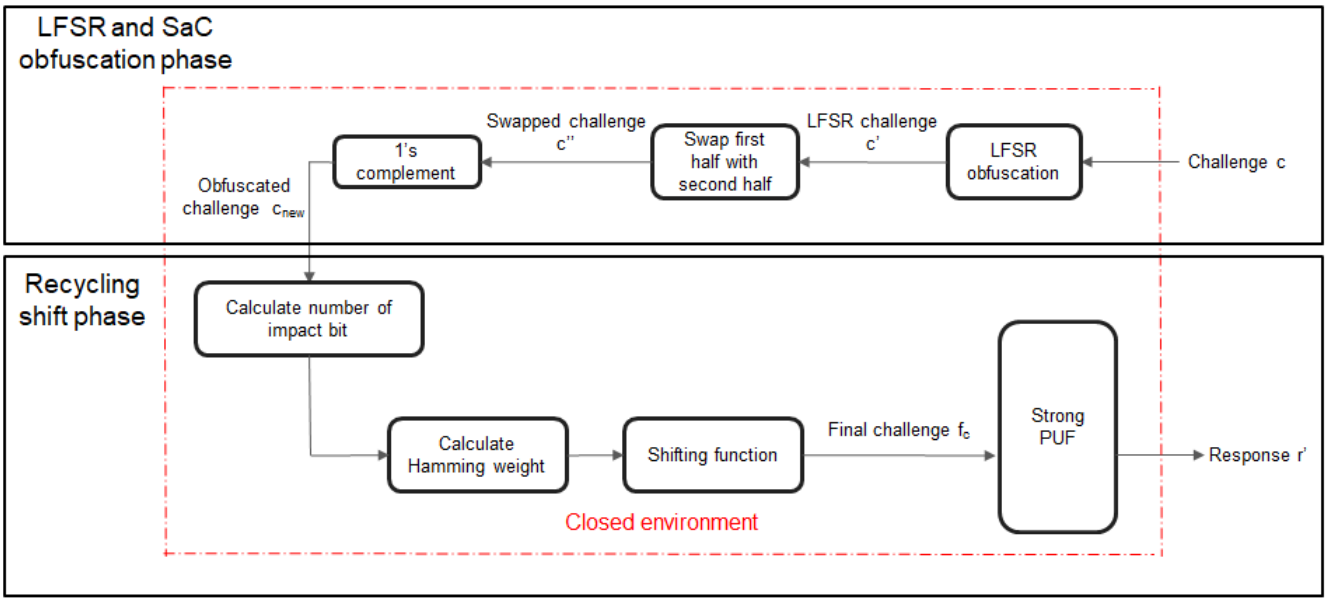

Figure 9. Proposed approach for ML-attack resilience.

defined key for encryption. Instead, it depends on a specific equation and input challenge. Thus, the calculated shifting value used in the algorithm afterwards has lower possibility of data leakage and is harder for the attacker to determined. In general. The Hamming weight is not regarded as the shift quantity. However, the Hamming weight of a certain segment of a challenge is considered as a binary number and has to be converted into decimal to determine the shift quantity. The reasons are as follows: (1) the effect of directly using the Hamming weight as the number of shifts is limited, and it has been verified that it has to be more than 20 digits to produce a noticeable effect. (2) after the conversion, only a few challenges are required to traverse all the displacement situations. The amount of shift is determined by applying the Hamming weight on several bits in front of the challenge and converting it to a decimal number. The reason for only using several bits in front of the challenge is because according to the algorithm, this is all it needs to represent all possible numbers of shifting. For example, 64 bits of challenge only need to look at 6 bits to represent all possible shifting numbers. After receiving the obfuscated challenge from the SaC phase, we first observe the length of the obfuscated challenge C_{new} to determine the number of bits z, which will be used for calculating Hamming weight.We then iterate through the first z bits that were previously determined by the challenge and apply an equation to convert each bit into a decimal number. The final decimal number b is the sum of each iteration. Then, the value of decimal number b is used as the shifting number in the obfuscated challenge c_{new}. Finally, the final challenge f_c is provided to the PUF that outputs a response $r^{'}$. The entire procedure flowchart of the resilience framework with SaC obfuscation is shown in Fig. 9. Algorithm 1 describes the cyclic shift algorithm. In order to show the effectiveness of the proposed scheme, we conduct a simulation in Python 3.8.12 on an Intel i7-9700k CPU @ 3.60GHz and 16 GB memory. The standard ML methods and XGBoost that are used in modelling attacks are constructed with the scikit-learn and XGBoost library in Python. Table II shows the effectiveness of the proposed scheme.

Algorithm 1: Recycling shift

Input: New challenge 2D array: c_{new}
Output: Final challenge 2D array: f_c

```
l = Column size of c_new ;
z = log(l, 2) ;
for each x ∈ [0, 1, 2, ..., z] do
    b = b + c_new[x] * 2^x
f_c = c_new shift right b bits ;
return f_c;
```

1) LFSR obfuscation: In general, a LFSR consists of a shift register and a linear feedback function. In this phase of our obfuscation method utilizes the few bits in challenge as the input of LFSR and generates a obfuscated challenge to break the linear relationship between challenge and response. In this regard, the random challenge generator generates (n+4) bits challenge C_1, and separate into 4 bits binary number C_2 and the a n bits challenge C_3. The 4 bits C_2 can be collected in two ways, in the first way we will simply take the first or last 4-bits of the challenge C_1. Whereas in the second way we randomly chose 4-bits in the challenge C_1 and combined into C_2. After C_2 is determined, it is converted from binary number to a decimal number, and will multiply with a predefined number to calculate the shift count that will input into the LFSR to enhance the protection, where shift count = predefined number X decimal number of C_2.

Table II
THE PERFORMANCE OF MODELLING ATTACK ON LFSR-SAC RESILIENCE FRAMEWORK IN TERMS OF CORRECT PREDICTION ACCURACY

PUF type	**Proposed LFSR-SaC resilience framework**	**No resilience**
APUF	54.1%	99.5%
3 XOR-APUF	47.4 %	97.3%
4 XOR-APUF	48.4%	96.2%
5 XOR-APUF	51.1%	92.5%
6 XOR-APUF	52.1%	89.0%
FF-3-XOR-APUF	52.4%	96.1%
FF-4-XOR-APUF	51.4%	94.0%
FF-5-XOR-APUF	50.0%	93.4%
FF-6-XOR-APUF	49.2%	92.3%

IV. SECURITY ANALYSIS

In this section, we analyze the security of the proposed solution from two perspectives. We first show how the proposed scheme ensures all the important security features as discussed in Section 1.1. Subsequently, we present our analysis results of this scheme from ProVerif and the Burrows-Abadi-Needham (BAN) logic.

A. Informal Security Analysis

In this informal analysis section, we focus on analyzing the security of the scheme against various attacks.

1) Confidentiality and Integrity: In the proposed scheme, $Temp^i_{T_X}$, S_i, K_i and C_i are all dynamically updated. This ensures that even if the attacker captures interactive data in the insecure channel, it is difficult to obtain valuable information. For example, in the scheme, R_i and R_{i+1} that serve as the authentication data are generated by the PUF and they are encrypted by a dynamic K_i, which provides data confidentiality. The validation of H_i and H_{i+1} ensures that any tampered message will be detected by the server or the tag, which ensures message integrity during the authentication process.

2) Forgery Attacks: In order to be successful against forgery attacks, the attacker must generate a correct response to the request from the server. In the proposed scheme, since the response R_i is generated by a strong PUF, attackers cannot obtain it directly by side-channel attacks. The only way to do that is to analyze the communication data. However, R_i is encrypted using K_i and K_i is generated inside the tag and the server by $hash(T_x||S_i)$. Besides, R_i and K_i are updated in every round of communication. Therefore the attacker cannot disguise as the tag. On the other hand, if an attacker tries to impersonate as a legitimate server, he/she would be required to produce the valid hash response H_i. However, since the adversary does not know the secret S_i, it will be difficult for him/her to generate a valid H_i.

3) Replay Attack: If the attacker pretends to be a legal tag and resends $MSG1 : \{Req_{Entry}, Temp^i_{T_X}\}$, he/she will not be successful because $Temp^i_{T_X}$ changes in every round. If the attacker resends $MSG2 : \{\Delta_i, H_i\}$, the message will be detected by the tag as expired. S_i changes every round so $K_i = hash(T_X||S_i)$ also changes every round. Therefore, when the tag verifies $H_i = hash(\Delta_i||K_i)$, the resent message from the attacker will be blocked. If the attacker resends $MSG3 : \{X - QR, Y, H_{i+1}\}$, for the same reason, the message will be blocked by the server.

4) Backward and Forward Security: The secret values such as K_i and R_i are valid only for a certain round. After that they are both updated to K_{i+1} and R_{i+1}. K_{i+1} is computed with the private values T_X and $S_{i+1} = hash(S_i)$ as $K_{i+1} = hash(T_X||S_{i+1)}$, and R_{i+1} is generated by the PUF. Therefore, even if the attacker manages to obtain K_i and R_i from a round, it will be difficult for him/her to infer K_{i-1}/K_{i+1} and R_{i-1}/R_{i+1}. In order to do that, the attacker needs a precise model of the XOR arbiter PUF inside the tag (which is difficult when X is large) and the secret T_X.

5) Tracking: Consider an adversary who wishes to intercept the communication between a tag and the server and track the tag's foot-print. In the proposed scheme, the tag uses a temporary identity $Temp^i_{T_X}$ during the execution of the authentication phase which is only valid for a specific round. Whereas, the real-identity of the tag is only known to the server. In addition, none of the parameters in the proposed authentication scheme are allowed to be sent twice. In this way, the proposed scheme is able to avoid tracking.

B. Performance Comparison and Discussion

In this section, we compare the proposed protocol with other traditional security protocols mentioned above such as the scheme of Bringer et al. [10] and Kardas et al. [12]. In addition, for comparing the security performance with other protocols with machine-learning resilience, we also consider the schemes of Yu et al. [14] and Liang et al. [15]. We consider several important security properties, as shown in Table III. As can be seen from the table, traditional protocols only provide a subset of the security properties achieved by the proposed protocol and also have poor resistance to modeling attacks. Moreover, [14] and [15] cannot guarantee security against many threats such as man-in-middle attacks and tracking attacks.

C. Formal Security Analysis

1) Proverif Simulation: ProVerif [33] is an widely-used automatic verifier for cryptographic protocols, which can effectively and comprehensively verify the security and robustness of the proposed scheme. To facilitate the verification, we simplified the protocol process to make it more appropriate for the ProVerif platform. We only consider this scheme as a one-round process with a setup phase and the first round of authentication phase. However, because the communication method of the subsequent process is exactly the same as that of the first round, and the process of generating security information is completely private, it does not affect the security verification of the whole scheme.

Table III
PERFORMANCE COMPARISON BASED ON THE SECURITY PROPERTIES

Schemes	SP1	SP2	SP3	SP4	SP5
Bringer et al.[10]	×	×	✓	✓	×
Kardas et al.[12]	✓	×	×	✓	×
Protocol #1 of Yu et al.[14]	✓	×	✓	×	✓
Protocol #2 of Yu et al.[14]	✓	×	✓	×	✓
Liang et al.[15]	×	×	×	×	✓
Proposed Scheme	✓	✓	✓	✓	✓

SP1:Man-in-Middle-Protection; **SP2**:DoS-Attack-Protection; **SP3**:Forward and Backward Secrecy; **SP4**:Untraceability; **SP5**:Resilience of Machine-Learning Attacks;

As shown in Fig. 10, the code uses the following definitions: c represents the public channel between the tag and the server and sc refers to the secure channel which is secret to the adversary. The tag ID Tx and the master key mk are private data stored locally in the tag and the server, and are thus defined as private values. The secret code S and challenge C are generated by the server. However, to simplify the verification process, we consider them as values stored in the server. Also, since S is not directly used in the communication and cannot be captured or analyzed by the adversary, we define it as a private value. For the encryption function in our scheme, we use symmetric encryption method and there is a pair of $senc()$ and $sedc()$ functions which represents encryption and decryption. Moreover, because hash functions are widely used with many different elements in security protocols [34], we also individually defined several hash functions to convert elements to hash values with fixed length.

The pre-definition part also states events in the protocol and the queries to test the relationships between them. To ensure the integrity and confidentiality of the information, we need to focus on two parts. The first one is whether the adversary can access the private data Tx, S and mk. These data are directly related to the security of the scheme and must not be obtained or analyzed by adversaries. Second, we validate the flow of events to prevent the adversary from replaying the communication messages. We mainly test $MSG2$ and $MSG3$ since these two messages contain secure information. For example, $query : inj-event(tagVerifyM2) ==> inj-event(serverSendM2)$ represents that for every time the tag receives $MSG2$ from the server, the server must have sent it only once.

```
free c : channel.
free sc : channel[private].
type key.

(*Values*)
free Tx : bitstring[private].
free S : bitstring[private].
free C : bitstring.
free mk : bitstring[private].

(*encryption*)
fun senc(bitstring,key):bitstring.
reduc forall m : bitstring, k : key; sedc(senc(m,k),k) = m.

(*hash*)
fun hash(bitstring):bitstring.
fun hashWithKey(bitstring,key):bitstring.
fun hashToKey(bitstring,bitstring):key.
fun hash2input(bitstring,bitstring):bitstring.
fun hash2inputWithKey(bitstring,key,bitstring):bitstring.
fun hash3input(bitstring,bitstring,bitstring):bitstring.
fun PUF(bitstring):bitstring[private].

(*events*)
event serverSendM2.
event tagVerifyM2.
event tagSendM3.
event serverVerifyM3.
event endSetup.
event startAuthentication.
event endAuthentication.

(*queries*)
query attacker(Tx).
query attacker(S).
query attacker(mk).
query inj-event(tagVerifyM2)==>inj-event(serverSendM2).
query inj-event(serverVerifyM3)==>inj-event(tagSendM3).
```

Figure 10. Pre-definition part in ProVerif.

After the pre-definition, we implement the main function of the tag and the server. As Fig. 11 shows, the tag loads its tag ID T_X and sends it through the secure channel sc to the server in step 1. After receiving a temporary id $TempT_X$, a secret S and a challenge C are generated by the server in step 2. The tag generates a response R using its PUF and sends it to the server in step 3. After the server receives this message, the event $endSetup$ is triggered in step 4, and the setup phase ends. Then, in step 5, the tag sends $TempT_X$ to the server to activate the authentication phase (which is $MSG1$). Next, in step 6, the server encodes the challenge C with a generated key K and sends it back with hash code H (which is $MSG2$). The event $serverSendM2$ is triggered here. In step 7, after verifying H, the event $tagVerifyM2$ is triggered. Then, the tag receives the feedback, generates a new response R and encodes it to $MSG3$. The encryption function here refers to the QR-code algorithm. Finally, after the event $tagSendM3$ triggered, the server verifies two responses and finishes the authentication phase.

The program is successfully executed as shown in Fig. 12. It is clearly shown that the queries are secure, the private values are secure so that adversaries cannot access them, and the event relationships are proved true so that the protocol can detect forged messages from adversaries. In conclusion, the protocol is secure in the ProVerif environment.

```
(*tag*)
let tag()=
        (*step1*)
        out(sc,Tx);

        (*step3*)
        in(sc,(S:bitstring,TempTx:bitstring,C:bit
string));
        let R = PUF(C) in
        out(sc,R);

        (*step5*)
        out(c,TempTx);

        (*step7*)
        in(c,(Delta:bitstring,H:bitstring));
        let K=hashToKey(TempTx,S) in
        let (=H)=hashWithKey(Delta,K) in
        event tagVerifyM2;
        let C=sedc(Delta,K) in
        let R = PUF(C) in
        let X = senc(R,K) in
        event tagSendM3;
        out(c,X).
```

```
(*server*)
let server()=

        (*step2*)
        in(sc,Tx:bitstring);
        let TempTx = hash3input(S,Tx,mk) in
        out(sc,(S,TempTx,C));

        (*step4*)
        in(sc,R:bitstring);
        event endSetup;

        (*step6*)
        in(c,TempTx:bitstring);
        let K=hashToKey(TempTx,S) in
        let Delta=senc(C,K) in
        let H=hashWithKey(Delta,K) in
        event serverSendM2;
        out(c,(Delta,H));

        (*step8*)
        in(c,X:bitstring);
        let (=R) = sedc(X,K) in
        event serverVerifyM3;
        event endAuthentication.
process
        tag()|server()
```

Figure 11. Protocol implementation process.

```
-- Query not attacker(Tx[]) in process 1.
Translating the process into Horn clauses...
nounif mess(sc[],Tx_2)/-5000
Completing...
Starting query not attacker(Tx[])
RESULT not attacker(Tx[]) is true.
-- Query not attacker(S[]) in process 1.
Translating the process into Horn clauses...
nounif mess(sc[],Tx_2)/-5000
Completing...
Starting query not attacker(S[])
RESULT not attacker(S[]) is true.
-- Query not attacker(mk[]) in process 1.
Translating the process into Horn clauses...
nounif mess(sc[],Tx_2)/-5000
Completing...
Starting query not attacker(mk[])
RESULT not attacker(mk[]) is true.
-- Query inj-event(tagVerifyM2) ==> inj-event(serverSendM2) in process 1.
Translating the process into Horn clauses...
nounif mess(sc[],Tx_2)/-5000
Completing...
Starting query inj-event(tagVerifyM2) ==> inj-event(serverSendM2)
RESULT inj-event(tagVerifyM2) ==> inj-event(serverSendM2) is true.
-- Query inj-event(serverVerifyM3) ==> inj-event(tagSendM3) in process 1.
Translating the process into Horn clauses...
nounif mess(sc[],Tx_2)/-5000
Completing...
Starting query inj-event(serverVerifyM3) ==> inj-event(tagSendM3)
RESULT inj-event(serverVerifyM3) ==> inj-event(tagSendM3) is true.
--------------------------------------------------------------
```

Figure 12. Proverif result.

2) BAN Logic: BAN logic is a set of rules for defining and analyzing information exchange protocols. It can help users determine the reliability of the information they exchange. BAN logic is based on the assumption that all information exchanges take place in public media which is easy to temper with. The main constructions of BAN logic are described as follow [35].

- P $|\equiv X$ represents P believes X. P acts as if X is true, and may assert X in other messages.
- P $\triangleleft X$ represents P sees X. P receives message X, and can read and repeat X.
- P $|\sim X$ represents P said X. At one time, P transmitted (and believed) message X, although P might no longer believe X.
- P $|\Rightarrow X$ represents P has jurisdiction over X. P's beliefs about X should be trusted.
- #(X) represents X has not previously been sent in any message.
- $\{ X \}_K$ represents formula X is encrypted by key K.
- P $\overset{K}{\longleftrightarrow} Q$ represents P and Q share a secret key K.
- $\overset{K}{\longmapsto} P$ represents P has a published public key K.
- P $\overset{K}{\rightleftharpoons} Q$ represents X is a secret known only to P, Q and possibly some trusted associates.

The following rules of interference required in our analysis:

- Message-Meaning Rule R1: $[(P \ |\equiv \ P \overset{K}{\longleftrightarrow} Q, P \triangleleft \{X\}_K)/(P \ |\equiv (|\sim X)]$.
- Nonce-Verification Rule R2: $[(P \ |\equiv \#(X), P \ |\equiv (Q \ |\sim X))/(P \ |\equiv (Q \ |\equiv X))]$.
- Jurisdiction Rule R3:$[(P \ |\equiv \ (Q \ |\Rightarrow \ X), P \ |\equiv \ Q \ |\equiv X)/(P \ |\equiv X)]$.
- Seeing Rules R4:$[P \triangleleft (X,Y)/P \triangleleft X], [(P \ |\equiv \ P \overset{K}{\longleftrightarrow} Q, P \triangleleft \{X\}_K)/P \triangleleft X]$.
- Fresh Rule R5: $[P \ |\equiv \#(X)/P \ |\equiv \#(X,Y)]$
- Belief Rule R6: $[P \ |\equiv (X,Y)/P \ |\equiv X]$.

To analyze the proposed scheme, we need to extend the conventional rules with the following:

- R7 :$[(P \ |\equiv P \overset{K}{\longleftrightarrow} Q, P \triangleleft f(X,Y))/(P \ |\equiv Q \ |\sim Y]$.
- R8 :$[(P \ |\equiv \overset{K}{\longmapsto} Q, P \triangleleft X)/(P \ |\equiv Q \ |\sim X)]$.

Since $Temp^i_{T_X}$ is related to the initial T_X only in the first round, we choose i as 1 for analysis. After the server is proved to believe in $Temp^1_{T_X}$, the security of following rounds can be proved iteratively. The protocol can be simply explained as follows:

- MSG1: $Sv \triangleleft Temp^1_{T_X}$.
- MSG2: $Tag_i \triangleleft (\{C_1\}_{K_1}, H_1)$.
- MSG3: $Sv \triangleleft (\{R_1\}_{K_1}, Y, H_2)$.

The target of the analysis of the proposed scheme is to prove $Sv \ |\equiv Temp^1_{T_X}$.

The initial security assumptions on the RFID tag Tag_i and the server Sv are described as follows.

- $Sv \ |\equiv \overset{T_X}{\longmapsto} Tag_i$.
- $Tag_i \ |\equiv Tag_i \overset{Temp^1_{T_X}, S_1, C_1}{\longleftrightarrow} Sv$.
- $Sv \ |\equiv Tag_i \overset{R_1}{\rightleftharpoons} Sv$.
- $Tag_i \ |\equiv Tag_i \overset{K_1}{\longleftrightarrow} Sv$ and $Sv \ |\equiv Sv \overset{K_1}{\longleftrightarrow} Tag_i$.

For each tag Tag_i, we can write $Tag_i \ |\equiv Sv \ |\sim MSG2$ and $Tag_i \ |\equiv \#(MSG2)$. So, according to R2 and R3, $Tag_i \ |\equiv Sv \ |\equiv MSG2$. Then, when Tag_i receives $MSG2$, we can infer the following statements:

$$\frac{Tag_i \ |\equiv (H_1, K_1)}{Tag_i \ |\equiv H_1},$$

$$\frac{Tag_i \ |\equiv (MSG2, K_1)}{Tag_i \ |\equiv MSG2},$$

$$\frac{Tag_i \ |\equiv Tag_i \overset{K_1}{\longleftrightarrow} Sv, Tag_i \triangleleft f(hash(\Delta_1||K_1), H_1)}{Tag_i \ |\equiv Sv \ |\sim H_1}.$$

Similarly, when Sv receives $MSG3$ from Tag_i, we can also infer the following statements:

$$\frac{Sv \ |\equiv (H_2, K_1)}{Sv \ |\equiv H_2},$$

```
Tx = "tagID"
mk = "masterKey"

#generate S[1],C[1],TempTx[1]
chars = string.ascii_letters + string.digits
S = [random.choice(chars) for i in range(8)]
S = "".join(S)
C = random_inputs(n=64, N=16, seed=int(random.random()))
TempTx = hash(S+(Tx+mk))

#PUF simulation
puf = XORArbiterPUF(n=64, k=8, seed=int(random.random()), noisiness=0)
rarray = puf.eval(C)
R = array2bin(rarray)

#write to memory
tag_1 = [[TempTx,S]]
server_1 = [[TempTx,challenge2str(C),R,S]]
with open('tag_memory.csv', mode='a', encoding='utf-8-sig') as csvfile:
    writer = csv.writer(csvfile)
    for line in tag_1:
        writer.writerow(line)
with open('server_memory.csv', mode='a', encoding='utf-8-sig') as csvfile:
    writer = csv.writer(csvfile)
    for line in server_1:
        writer.writerow(line)
```

Figure 13. The main code of setup phase.

```
Setup phase start

TagID Tx: tagID
Masterkey mk: masterKey

For the first round, secret code S1:
6u8kqQp0

The challenges C1:
[[-1 -1  1 ...  1 -1 -1]
 [-1 -1 -1 ...  1 -1  1]
 [ 1 -1  1 ... -1  1  1]
 ...
 [-1  1 -1 ...  1  1 -1]
 [ 1 -1  1 ...  1  1 -1]
 [-1  1 -1 ...  1 -1  1]]

The TempTx1:
3037364642573028312

The response R1 from XOR PUF:
0011000000100010

Writing in to memory...
Setup Phase end
```

Figure 14. The program output of setup phase.

$$\frac{Sv \mid\equiv (MSG3, H_2)}{Sv \mid\equiv MSG3},$$

$$\frac{Sv \mid\equiv \{R_1\}_{K_1}, Sv \mid\equiv Sv \overset{K_1}{\longleftrightarrow} Tag_i}{Sv \mid\equiv R_1},$$

$$\frac{Sv \mid\equiv \overset{MSG3}{\longmapsto} Tag_i, Sv \triangleleft MSG3}{Sv \mid\equiv Tag_i \mid\sim MSG3},$$

$$\frac{Sv \mid\equiv Tag_i \overset{K_1}{\longleftrightarrow} Sv, Sv \triangleleft f(hash(X||K_1||Y), H_2)}{Sv \mid\equiv Tag_i \mid\sim H_2}.$$

Then, according to the process, we can write that $Sv \mid\equiv Tag_i \mid\sim Temp^1_{T_X}$ since $Sv \mid\Rightarrow Temp^i_{T_X}$ and $Sv \mid\sim Temp^i_{T_X}$. Thus, we can infer following statements with R2, R3 and R7:

$$\frac{Sv \mid\equiv \overset{Temp^1_{T_X}}{\longmapsto} Tag_i, Sv \triangleleft Temp^1_{T_X}}{Sv \mid\equiv Tag_i \mid\sim Temp^1_{T_X}},$$

$$\frac{Sv \mid\equiv Tag_i \mid\Rightarrow Temp_{T_X}, Sv \mid\equiv Tag_i \mid\equiv Temp^1_{T_X}}{Sv \mid\equiv Temp^1_{T_X}}.$$

From the reasoning process above, we prove that the proposed scheme is reliable in BAN logic system.

V. Implementation

The QR-PUF system proposed in this article can be used in many scenarios that require fast and accurate authentication of identity information. In this section we show the implementation of XOR arbiter PUF simulation on pypuf and discuss the results of our protocol execution with Python. We use Python code to simulate the whole protocol, including XOR arbiter PUF simulation, security information generation and calculation, QR code generation and scanning, communication process simulation, etc. In this simulation, we use a 8-channel, 64-bit XOR arbiter PUF to generate the response (with no noise). Besides, we use two CSV files to simulate the memory of the tag and the server.

As shown in Fig. 13, there are 3 main steps in setup phase. First we define the initial tag ID, Tx, and master key, mk, and generate S_1, C_1, and $Temp^1_{T_X}$. Then, we input C_1 into pypuf for a response R_1. Finally, we write tag_1 and $server_1$ into the memory, separately. The program result of the setup phase is shown in Fig. 14.

The implementation code of authentication phase is shown in Fig 15. Corresponding to Fig. 11, there are 4 verifications in this phase: MSG_1, MSG_2, MSG_3 and the final response R_i. After extracting data from each memory, the program verifies the temporary ID $Temp_{T_X}$. Then we generate the key K_i in this round and encrypt the challenge matrix C_i with it. The second step verifies the hash code H_i generated by the tag and the server separately. After $MSG2$ is confirmed to be credible, we use the same pypuf model to compute the response R_i, encrypt it, and generate a QR code. Then, the tag and the server generate the next round of data in the simulation. The third verification is on H_{i+1}, the hash code computed by next round of data. In the end, the server decrypts the QR code and performs the last round of verification on response R_i.

The result of the first and second round authentication phase is shown in Fig. 16 and Fig 17 shows the generated QR code in round 1. After simulation of 7 rounds of authentication, the data stored in memory is shown in Fig. 18.

Acknowledgement

The work of Prosanta Gope was supported by The Royal Society Research Grant under grant RGS\R1\221183. The work of Biplab Sikdar is supported in part by the Ministry of Education, Singapore under grant R-263-000-D63-114.

VI. Conclusion

The security of Mobile RFID devices has become an increasingly serious problem. PUFs, because of their excellent

```
with open('tag_memory.csv','r') as tag_memory:
    reader = csv.reader(tag_memory)
    result = list(reader)
    TempTxi_t = result[-1][0]
    Si_t = result[-1][1]

with open('server_memory.csv','r') as server_memory:
    reader = csv.reader(server_memory)
    result = list(reader)
    TempTxi_s = result[-1][0]
    Ci_s = result[-1][1]
    Ri_s = result[-1][2]
    Si_s = result[-1][3]

if TempTxi_s != TempTxi_t:
    print("MSG1 - Wrong Temporary ID, stop authentication.")
    break
else:
    print("MSG1 - Temporary ID verified, loading data...")

# compute K
Ki_s = hash(TempTxi_s+Si_s)
print("The key K"+str(round)+": ")
print(Ki_s)

# encrypt challenges
Di = []
for i in range(64):
    Di.append(encrypt(Ki_s, Ci_s[i*16:(i+1)*16]))
Hi_s = hash("".join(Di)+str(Ki_s))

# verify Hi
Ki_t = hash(TempTxi_t+Si_t)
Hi_t = hash("".join(Di)+str(Ki_t))
if Hi_t != Hi_s:
    print("MSG2 = Wrong hashcode, stop authentication")
    break
else:
    print("MSG2 - Hashcode verified, generating data...")

# PUF simulation
for i in range(64):
    Ci_t=Ci_t+decrypt(Ki_t,Di[i])
Ci_t = str2challenge(Ci_t)
rarray = puf.eval(Ci_t)
Ri_t = array2bin(rarray)
X_t = encrypt(Ki_t,Ri_t)

# generate QrCode
qr = qrcode.QRCode(version=2,
error_correction=qrcode.constants.ERROR_CORRECT_L,
box_size=10, border=10, )
qr.add_data(X_t)
qr.make(fit=True)
img = qr.make_image()
img.show()
img.save('qrcode'+str(round)+'.png')
```

```
# Tag Generate next round data
Si1_t=hash(Si_t)
TempTxi1_t = hash(TempTxi_t+str(Si1_t))
Ci1_t = random_inputs(n=64, N=16, seed=int(str(Si1_t)[1:4]))
Ri1_t = puf.eval(Ci1_t)
Ri1_t = array2bin(Ri1_t)
Ki1_t = hash(Tx+str(Si1_t))
Y = encrypt(Ki1_t,Ri1_t)
Hi1_t = hash(str(X_t)+str(Ki_t)+str(Y))
tag_i = [[TempTxi1_t,Si1_t]]
with open('tag_memory.csv', mode='a', encoding='utf-8-sig',
newline='') as csvfile:
    writer = csv.writer(csvfile)
    for line in tag_i:
        writer.writerow(line)

# Decrypt QrCode
image = 'qrcode' + str(round) + '.png'
img = Image.open(image)
barcodes = decode(img)
for barcode in barcodes:
    X_s = barcode.data.decode("utf-8")

# Server generate next round data
Si1_s = hash(Si_s)
TempTxi1_s = hash(TempTxi_s + str(Si1_s))
Ci1_s = random_inputs(n=64, N=16, seed=int(str(Si1_s)[1:4]))
Ki1_s = hash(Tx + str(Si1_s))
Hi1_s = hash(X_s + str(Ki_s) + str(Y))
if Hi1_s != Hi1_t:
    print("MSG3 - Wrong hashcode, stop authentication.")
    break
else:
    print("MSG3 - Hashcode verified, decrypting data...")

# Authentication
decrypted_response = decrypt(Ki_s, X_s)
if Ri_s != decrypted_response:
    print("Wrong response, authentication fail.")
    break
else:
    print("Response verified, authentication success.")
Ri1_s = decrypt(Ki1_s,Y)

# Store new data
server_i = [[TempTxi1_s,challenge2str(Ci1_s),Ri1_s,Si1_s]]
with open('server_memory.csv', mode='a', encoding='utf-8-sig',
newline='') as csvfile:
    writer = csv.writer(csvfile)
    for line in server_i:
        writer.writerow(line)
```

Figure 15. The main code of authentication phase.

In the 1-th round, TempTx1 from tag:	In the 2-th round, TempTx2 from tag:
3037364642573028312	-87741694616166927
TempTx1 from server:	TempTx2 from server:
3037364642573028312	-87741694616166927
MSG1 - Temporary ID verified, loading data...	**MSG1 - Temporary ID verified, loading data...**
The key K1:	The key K2:
3286734653483427424	7699624145425643854
MSG2 - Hashcode verified, generating data...	**MSG2 - Hashcode verified, generating data...**
Response R1:	Response R2:
0011000000100010	1111010111010000
Generating QR-Code, close the picture to continue...	Generating QR-Code, close the picture to continue...
The data scanned from QR Code:	The data scanned from QR Code:
99_98_105_103_103_99_100_102_101_99_101_104_99_100_99_103_	104_103_106_106_102_99_100_98_101_102_100_99_101_102_100_99_
MSG3 - Hashcode verified, decrypting data...	**MSG3 - Hashcode verified, decrypting data...**
The decrypted response from QR Code:	The decrypted response from QR Code:
0011000000100010	1111010111010000
The response Ri stored in database:	The response Ri stored in database:
0011000000100010	1111010111010000
Response verified, authentication success.	**Response verified, authentication success.**

Figure 16. The program result of first two rounds of the authentication phase.

robustness and reconfigurability in lightweight devices, are popular as a solution to this problem. Considering the advantages and disadvantages of RFID and QR code technology in different fields, this article proposes a communication protocol scheme between Mobile-RFID tag and the authenticator (server) which can effectively protect against man-in-the-middle, forward and backward, and other attacks. Using a formal and comprehensive security analysis, we proved that the scheme provides reliable two-way authentication on public channels. We also designed an actual implementation of the scheme, and conduct a security analysis of the application use case. Therefore, the proposed scheme is a feasible and promising solution for the security of RFID devices.

Figure 17. The QR-Code generated in the first round.

TagMemory:

	TempTx	S
2	3037364642573020000	6u8kqQp0
3	-87741694616166900	3546021404899610000
4	6856891109276570000	-9017860878312980000
5	-4587930050190760000	-5038898561825900000
6	-3445039633461520000	-4969187715599300000
7	-2139573963293040000	-5887284935787710000
8	7883742936605490000	4093651862815110000
9	5885681965712430000	-8067005807640310000

ServerMemory:

	TempTx	Challenge	Response	S
2	3037364642573020000	1111101101010110	11000000100010	6u8kqQp0
3	-87741694616166900	1001111000000000001	1111010111010000	3546021404899610000
4	6856891109276570000	1001101001101000010	1010000111101	-9017860878312980000
5	-4587930050190760000	1110000101000101011	1001101000110110	-5038898561825900000
6	-3445039633461520000	0111101000101111110	1010110011010110	-4969187715599300000
7	-2139573963293040000	0101111000001011000	1111111110111110	-5887284935787710000
8	7883742936605490000	1011010111111100001	1001111100010	4093651862815110000
9	5885681965712430000	0011110101011011100	10010100110011	-8067005807640310000

Figure 18. Data in memory after 7 rounds.

REFERENCES

[1] M. Lehtonen, T. Staake, and F. Michahelles, "From identification to authentication–a review of rfid product authentication techniques," *Networked RFID Systems and Lightweight Cryptography*, pp. 169–187, 2008.

[2] Y.-G. Kim and M.-S. Jun, "A design of user authentication system using qr code identifying method," in *2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*. IEEE, 2011, pp. 31–35.

[3] K. Domdouzis, B. Kumar, and C. Anumba, "Radio-frequency identification (rfid) applications: A brief introduction," *Advanced Engineering Informatics*, vol. 21, no. 4, pp. 350–355, 2007.

[4] M. Makary, *Unaccountable: what hospitals won't tell you and how transparency can revolutionize health care*. Bloomsbury Publishing USA, 2013.

[5] W. H. Organization, *World health statistics 2009*. World Health Organization, 2009.

[6] C. T. Nguyen, "Examination of cloud privacy & security regulations of electronic health records," Ph.D. dissertation, Utica College, 2021.

[7] R. Xu, L. Zhu, A. Wang, X. Du, K.-K. R. Choo, G. Zhang, and K. Gai, "Side-channel attack on a protected rfid card," *IEEE Access*, vol. 6, pp. 58 395–58 404, 2018.

[8] P. Gope, O. Millwood, and N. Saxena, "A provably secure authentication scheme for rfid-enabled uav applications," *Computer Communications*, vol. 166, pp. 19–25, 2021.

[9] K. Fan, W. Jiang, Q. Luo, H. Li, and Y. Yang, "Cloud-based rfid mutual authentication scheme for efficient privacy preserving in iov," *Journal of the Franklin Institute*, 2019.
[10] Bringer, Chabanne, and Icart, "Improved privacy of the tree-based hash protocols using physically unclonable function," *LECT NOTE COMPUT SCI*, vol. 2008,5229, no. -, pp. 77–91, 2008.
[11] A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "Puf-enhanced rfid security and privacy," in *Workshop on secure component and system identification (SECSI)*, vol. 110, 2010.
[12] Suleyman, Kardas, , , Serkan, Celik, , , Muhammet, Yildiz, , , and Albert, "Puf-enhanced offline rfid security and privacy," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 2059–2067, 2012.
[13] Y. Jing, Q. Guo, H. Yu, H. Li, and X. Li, "Modeling attacks on strong physical unclonable functions strengthened by random number and weak puf," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, 2018.
[14] M. Yu, M. Hiller, J. Delvaux, R. Sowell, and I. Verbauwhede, "A lockdown technique to prevent machine learning on pufs for lightweight authentication," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 3, pp. 146–159, 2017.
[15] W. Liang, S. Xie, J. Long, K. C. Li, and K. Li, "A double puf-based rfid identity authentication protocol in service-centric internet of things environments," *Information Sciences*, vol. 503, 2019.
[16] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender puf protocol: A lightweight, robust, and secure authentication by substring matching," in *2012 IEEE Symposium on Security and Privacy Workshops*. IEEE, 2012, pp. 33–44.
[17] O. Näslund, "Lightweight and machine learning attack resistant physical unclonable functions," 2019.
[18] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*. IEEE, 2007, pp. 9–14.
[19] S. Sutar, A. Raha, and V. Raghunathan, "D-puf: An intrinsically reconfigurable dram puf for device authentication in embedded systems," in *2016 International Conference on Compliers, Architectures, and Sythesis of Embedded Systems (CASES)*. IEEE, 2016, pp. 1–10.
[20] U. Rührmair, J. Sölter, and F. Sehnke, "On the foundations of physical unclonable functions." *IACR Cryptol. ePrint Arch.*, vol. 2009, p. 277, 2009.
[21] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Secure lightweight entity authentication with strong pufs: Mission impossible?" in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 451–475.
[22] L. Santiago, V. C. Patil, C. B. Prado, T. A. Alves, L. A. Marzulo, F. M. França, and S. Kundu, "Realizing strong puf from weak puf via neural computing," in *2017 IEEE international symposium on defect and fault tolerance in VLSI and nanotechnology systems (DFT)*. IEEE, 2017, pp. 1–6.
[23] M. Ebrahimabadi, M. Younis, W. Lalouani, and N. Karimi, "A novel modeling-attack resilient arbiter-puf design," in *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*. IEEE, 2021, pp. 123–128.
[24] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and reliable xor arbiter puf design: An experimental study based on 1 trillion challenge response pair measurements," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.
[25] F. Ganji, S. Tajik, and J.-P. Seifert, "Why attackers win: on the learnability of xor arbiter pufs," in *International Conference on Trust and Trustworthy Computing*. Springer, 2015, pp. 22–39.
[26] M. Roel, "Physically unclonable functions: Constructions, properties and applications," *Katholieke Universiteit Leuven, Belgium*, 2012.
[27] J. B. Schmitt, *Measurement, Modeling, and Evaluation of Computing Systems and Dependability and Fault Tolerance: 16th International GI/ITG Conference, MMB & DFT 2012, Kaiserslautern, Germany, March 19-21, 2012, Proceedings*. Springer, 2012, vol. 7201.
[28] G. Yue, "The study of the application of o2o e-commerce model in china." in *WHICEB*, 2016, p. 51.
[29] M. M. S. Rani and K. R. Euphrasia, "Data security through qr code encryption and steganography," *Advanced Computing: An International Journal (ACIJ)*, vol. 7, no. 1/2, pp. 1–7, 2016.
[30] W. Xiao-Long, W. Chun-Fu, L. Guo-Dong, and C. Qing-Xie, "A robot navigation method based on rfid and qr code in the warehouse," in *2017 Chinese Automation Congress (CAC)*. IEEE, 2017, pp. 7837–7840.
[31] C. U. Reddy, D. V. P. Reddy, N. Srinivasan, and J. A. Mayan, "Bus ticket system for public transport using qr code," in *IOP Conference Series: Materials Science and Engineering*, vol. 590, no. 1. IOP Publishing, 2019, p. 012036.
[32] M. A. M'hand, A. Boulmakoul, H. Badir, and A. Lbath, "A scalable real-time tracking and monitoring architecture for logistics and transport in roro terminals," *Procedia Computer Science*, vol. 151, pp. 218–225, 2019.
[33] B. Blanchet, V. Cheval, X. Allamigeon, and B. Smyth, "Proverif: Cryptographic protocol verifier in the formal model," 2010.
[34] 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 Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1971–1980, 2021.
[35] P. Gope and B. Sikdar, "A comparative study of design paradigms for puf-based security protocols for iot devices: Current progress, challenges, and future expectation," *Computer*, vol. 54, no. 11, pp. 36–46, 2021.

Prosanta Gope (Senior Member, IEEE) is currently working as an Assistant Professor in the Department of Computer Science (Cyber Security) at the University of Sheffield, UK. Dr Gope served as a Research Fellow in the Department of Computer Science at the National University of Singapore (NUS). Primarily driven by tackling challenging real-world security problems, he has expertise in lightweight authentication, authenticated encryption, access control, security of mobile communications, healthcare, Internet of Things, Cloud, RFIDs, WSNs, Smart-Grid and IoT Hardware. He has authored more than 100 peer-reviewed articles in several reputable international journals and conferences and has four filed patents. He received the Distinguished PhD Scholar Award in 2014 from the National Cheng Kung University (Taiwan). Several of his papers have been published in high-impact journals such as IEEE TIFS, IEEE TDSC, IEEE TIE, IEEE TSG, etc. Dr Gope has served as TPC/Co-Chair member in several reputable international conferences such as IEEE TrustCom, IEEE GLOBECOM(Security-Track), ARES, ESORICS, etc. He currently serves as an Associate Editor of the IEEE Internet of Things Journal, IEEE Systems Journal, IEEE Sensors Journal, and the Journal of Information Security and Applications (Elsevier).

Yuening Wang received a MsC Degree in Artificial Intelligence and Cybersecurity from the University of Sheffield in 2021 and currently is a Doctoral Candidate at Beijing University Of Technology. He is interested in Lightweight Authentication, Physical Uncloneable Functions and currently researching in Heterogeneous Computing.

Zengpeng Li received a PhD degree from Harbin Engineering University (HEU), China, in 2018. During his doctoral program, he was a research assistant with the University of Auckland, New Zealand and Virginia Commonwealth University, USA, respectively. Currently, he is joining Nankai University (NKU), China. Prior to joining NKU, he has worked as a faculty member of Qingdao University (QDU), China, a postdoctoral research fellow at the Singapore University of Technology and Design (SUTD), Singapore, and a postdoctoral research associate at Lancaster University,

United Kingdom, respectively. His primary research interests are in cryptographic protocol and secure dis1362 tributed computing. In particular, his research efforts mainly focus on secure computing on encrypted data, verifiable computation, and pass1364 word-based cryptography.

Biplab Sikdar (Senior Member, IEEE) received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology Kanpur, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was a Faculty with Rensselaer Polytechnic Institute from 2001 to 2013, first as an Assistant and then as an Associate Professor. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include computer networks and security for IoT, and cyber physical systems. He served as an Associate Editor for the IEEE Transactions on Communications from 2007 to 2012. He is currently serving as an Associate Editor for the IEEE Transactions on Mobile Computing. He is a member of Eta Kappa Nu and Tau Beta Pi.