# A Secure PUF-Based Authentication Protocol for Remote Keyless Entry Systems in Cars

Rohini Poolat Parameswarath and Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—This paper first investigates various attacks against Remote Keyless Entry (RKE) systems in cars. Then, we propose an authentication protocol based on Physical Unclonable Functions (PUFs) to secure RKE systems. Detailed security analysis demonstrates that the proposed protocol prevents common attacks against RKE systems such as replay and RollJam attacks as well as the recently devised RollBack attack. To the best of our knowledge, this is the first protocol that addresses the RollBack attack. We use the concept of fuzzy extractor in the proposed protocol to address the noise in the PUF response due to the effect of environmental factors. The proposed protocol is also resilient to modeling attacks against PUFs. A performance analysis shows that the proposed authentication protocol is cost-effective.**

***Index Terms*—Authentication, Burrows–Abadi–Needham (BAN) logic, fuzzy extractor, Physical Unclonable Function (PUF), Remote Keyless Entry (RKE) systems, replay attack, RollBack attack, RollJam attack.**

## I. Introduction

Modern cars are far ahead of their predecessors in terms of the convenience they offer to users. The addition of more automotive electronics components to the cars to make them more user-friendly has resulted in an increase in attack surfaces as well [1]. Based on the attack surface, the attacks can be divided into physical access attacks and remote access attacks [1]. However, attacks that require physical access are operationally challenging [2]. Hence, this paper focuses on the attacks on Remote Keyless Entry (RKE) systems, a popular type of keyless entry systems, in cars that can be carried out remotely through short-range wireless access.

Keyless entry systems in cars eliminate the need to use physical keys to lock and unlock the car [3]. Keyless entry systems have two components: a key fob with the user and a receiver unit installed in the car. In RKE systems, pressing buttons on the key fob generates radio frequency (RF) signals that are sent to the car [4]. When the RF signals are received, the corresponding action, i.e., lock or unlock operation is triggered in the car. Static codes were used in the first generation of RKE systems. Capturing signals based on such static codes enables an adversary to replay the signals later to gain unauthorized access to the car. Hence, RKE systems with static code opened an attack surface where the adversary could act with minimum effort. Rolling codes were implemented to avoid such replay attacks. RKE systems that implement rolling codes generate a fresh code every time the key fob button is pressed. This code is unique and can be used only once to unlock the car [4]. In RKE systems, the KeeLoq rolling code scheme from Microchip Technology and the Hitag-2 scheme from NXP are extensively used. However, rolling code-based RKE systems are also vulnerable to attacks. The RollJam attack [5] is an attack devised against the rolling code-based RKE systems. In this attack, when a user presses the unlock button on the key fob, the adversary captures and stores the unlock signal. Then, the adversary jams the signal so that it does not reach the car. Since the first unlock attempt was unsuccessful, the user presses the unlock button again. The adversary captures, stores, and jams the second signal as well. At the same time, the adversary sends the first recorded signal to the car. Though the second signal is jammed, the car gets unlocked since the adversary has replayed the first captured signal. Thus, through this RollJam attack, the attacker captures the most recent signal to unlock the car. Since the car gets unlocked in the second attempt, the victim does not notice this attack. Through this attack, which can be realized by using relatively inexpensive equipment, it has been shown that even the rolling code-based RKE systems are susceptible to attacks. Further, a new attack called RollBack [6], [7] has recently been introduced. In this attack, the attacker captures and replays a few consecutive signals from the key fob. These signals can make RKE systems to roll back to a previous code thus unlocking the car.

Since an attacker can carry out these attacks by using relatively inexpensive hardware and software, it is crucial to have a preventive technique in place for such attacks. One of the essential requirements for RKE systems is to have an authentication protocol that makes it difficult for an adversary to execute the replay, RollJam, and RollBack attacks. Hence, we propose an authentication protocol based on Physical Unclonable Functions (PUFs) to secure RKE systems.

### A. Physical Unclonable Functions

Physical Unclonable Functions are circuit-based hardware security primitives that derive secrets from the physical characteristics of integrated circuits (ICs) [8]. They provide a challenge-response mechanism by mapping input challenges



This research was supported in part by Singapore Ministry of Education Academic Research Fund Tier 1 Grants R-263-000-E78-114 and R-263-001-E78-114.

R.P. Parameswarath and B. Sikdar are with the Department of Electrical and Computer Engineering, College of Design and Engineering, National University of Singapore, Singapore. (Email: rohini.p@nus.edu.sg, bsikdar@nus.edu.sg)

to responses. Instead of storing secrets in their memory, PUFs generate secrets based on the physical characteristics of the ICs, when required for a cryptographic operation. Two instances of PUF generate different responses for the same input. Cloning a PUF is almost impossible. Physical security, low cost, and power efficiency are some other characteristics of PUFs [9]. Due to these characteristics, PUFs have been widely used in security applications across various fields. PUFs can be used even in authentication scenarios where the devices involved are too resource-constrained to carry out expensive cryptographic operations.

Aging and environmental factors can have an impact on the response of PUFs. Since the PUF in the proposed protocol is used with a key fob where environmental conditions can affect the PUF response, there could be noise in the PUF response. We use the concept of fuzzy extractors [10] to address this noise issue in the PUF response. Though PUFs are unclonable, Machine Learning (ML) attacks that can model the PUFs have been reported in literature [11]. In the proposed protocol, we use a reconfigurable PUF to defend against such ML attacks [12].

### *B. Related Work*

In the past, researchers have examined automotive vulnerabilities and devised a variety of methods for attack detection and defence. In [2], the authors analyzed the external attack surfaces of cars. An adversary needs physical access, short-range wireless access, or long-range wireless access to the car to carry out attacks [2]. OBD port and entertainment systems such as CD players are two important physical interfaces to the car. Through physical access to the OBD port or to entertainment systems, an adversary may gain access to other systems since they are interconnected through the Controller Area Network (CAN) bus. Bluetooth, RKE systems, Radio Frequency Identification (RFID) systems, and WiFi are examples of attack surfaces for wireless interfaces that operate over short ranges. Broadcast channels and cellular networks are examples of long-range wireless channels that can be accessed from more than 1 km distance. A classification of attacks against autonomous vehicles and the solutions to protect from them was proposed in [13]. Chattopadhyay et al. analyzed different attack surfaces of vehicles in [14]. In [15], the authors presented a survey of attacks on vehicles and various defence techniques available. Attack surfaces on different car models that an attacker can exploit were surveyed in [16]. There have been studies focusing on keyless entry systems as well. The authors of [17] discussed different attacks against keyless entry systems.

Garcia et al. analyzed the security of some of the widely used RKE systems and demonstrated attacks against them in [4]. A discussion of the Hitag-2 algorithm for keyless entry systems was presented in [18]. Further, the authors of [18] demonstrated that the security of the Hitag-2-based RKE system can be compromised. An attack called RollJam attack against rolling codes-based RKE systems was presented in [5]. RollBack is a recent attack devised against RKE systems [6]. Compared to the RollJam attack, this attack reduces the adversary's effort. The details of the experiments and results are available in [6]. We have given details of RollJam and RollBack attacks in Section II D, Adversary Model.

While there have been studies on various vulnerabilities in RKE systems and attacks against RKE systems, several solutions were proposed to secure RKE systems from attacks as well. The authors of [3] proposed attack detection on keyless entry systems by differentiating legitimate unlock signals from malicious signals. To guard against the replay attacks in RKE systems, the authors of [19] presented a timestamp-based method. A technique based on time stamping and XOR encoding was presented in [20] to defend RKE systems against replay attacks. Ultimate KeeLoq technology [21] is based on a running timer to defend the system against replay attacks. Glocker et al. proposed a protocol based on symmetric encryption in [22]. In the solution proposed in [22], the requested instruction (lock/unlock) is sent in plain text. An adversary may modify this instruction resulting in an undesired outcome as the message integrity is not verified at the receiver. Though the timestamp-based solutions presented in [19] and [20] effectively protect RKE systems from common attacks, the key fob and the car receiver must have synchronized clocks for those solutions to work correctly. In [23], the asymmetric cryptographic technique Rivest–Shamir–Adleman (RSA) algorithm was used to build an authentication scheme for RKE systems. This solution is computationally expensive. Further, the solution in [23] also requires the key fob and the car receiver to have synchronized clocks. An authentication protocol secure against RollJam attacks is presented in [24]. However, this protocol has not addressed the RollBack attack. A PUF-based authentication protocol for keyless entry systems was proposed in [25]. Two factors, namely time-to-live (TTL) and nonce were used in [25] together with the PUF to ensure security. However, this protocol has certain drawbacks. The nonce and the PUF challenge are sent in plain text. An adversary may listen to these parameters. Also, the protocol does not mention the instruction (lock or unlock operation) to be executed. Instead, the successful execution of the protocol always results in the unlock operation of the car. To make the protocol complete and usable in a practical scenario, there should be an instruction field specifying the lock or unlock operation of the car. Further, the PUF is used with a key fob in [25] where environmental conditions can affect the PUF response. This noise issue is not addressed in [25]. Finally, there could be machine learning or modeling attacks against the PUF as well. These attacks are also not addressed in [25].

Most of the solutions mentioned above are based on computationally expensive operations. Since the authentication protocol must run each time the key fob button is pressed, the protocol should also be computationally efficient. To address the issues discussed above, we propose a lightweight authentication protocol based on PUFs for RKE systems. Researchers have proposed PUF-based authentication protocols in several domains including the Internet of Vehicles (IoV) [9], Internet of Things (IoT) [26], [27], Electric Vehicle (EV) charging [28], Internet of Drones (IoD) [29], and so on. Authentication protocols built on PUFs have also been

proposed for Wireless Sensor Networks (WSN) [30] and RFID systems [31].

This paper addresses the gaps in the existing literature by developing an authentication protocol that is secure against replay, RollJam, and RollBack attacks.

### C. Our Contributions

The key contributions of this paper are:

**1. Design of a lightweight mutual authentication protocol for RKE systems:** Since the communication channel between the key fob and the car is insecure, both parties want to ensure that the party on the other side is legitimate, not an adversary. To meet this requirement, we propose a mutual authentication protocol based on PUFs. The proposed protocol is computationally lightweight as well. The protocol addresses noise issues in the PUF response by using fuzzy extractors. The proposed protocol is designed in such a way that the PUFs used are robust against machine learning attacks as well.

**2. Protection from various attacks:** The proposed protocol is resilient to replay, RollJam, and RollBack attacks. To the best of our knowledge, no other protocol addresses the RollBack attack. The protocol also protects RKE systems from impersonation attacks by preventing the attacker from generating messages to impersonate a legitimate key fob.

**3. Security proof:** We provide security proofs and Burrows–Abadi–Needham (BAN) logic analysis to demonstrate that the proposed protocol is secure against various attacks.

The remaining sections of the paper are organized as follows. Section II discusses the basics of fuzzy extractor and PUF, the system model, the adversary model, and the security goals we want to achieve through the proposed protocol. In Section III, we present the proposed authentication protocol and in Section IV, we analyze its security using the BAN logic. The formal proof and informal security analysis are given in Section V. In Section VI, we provide a performance evaluation of the proposed protocol. The robustness of reconfigurable PUFs against machine learning attacks is analyzed in Section VII. The conclusions are given in Section VIII.

## II. Preliminaries

In this section, we first present the basics of fuzzy extractors and PUFs upon which the proposed authentication protocol is built. After that, we present the system model and the adversary model. Then, we discuss the security goals we want to achieve through the proposed authentication protocol.

### A. Fuzzy Extractor

The concept of fuzzy extractors can be applied to address the noise in the PUF response due to environmental factors. A fuzzy extractor is composed of two functions: a probabilistic key generation function, $FE.G()$, and a deterministic reconstruction function, $FE.R()$ [10], [32]. The two algorithms are given below:

- $FE.G()$: This function takes an input string $I$ and the outputs are a key $k \in \{0,1\}^n$ and helper data $hd \in \{0,1\}^*$:
$$(k, hd) \leftarrow FE.G(I). \quad (1)$$
- $FE.R()$: This function takes $I^{'}$ and the helper data $hd \in \{0,1\}^*$ as inputs and helps to recover the key $k$ if the Hamming distance between $I$ and $I^{'}$ is negligible:
$$(k) \leftarrow FE.R(I^{'}, hd). \quad (2)$$

Thus, fuzzy extractors help to store the randomness of an input and reproduce that randomness from another string that is close to the same input.

### B. PUFs

A PUF is a function that maps a challenge $P_i$ to a corresponding response $Q_i = PUF(P_i)$. During the chip manufacturing process, random variations are introduced in the physical structure of the integrated circuit that makes up the PUF. The response from the PUF is based on its underlying unique complex physical structure [8]. Hence, no two PUFs produce the same response for the same input. Due to this characteristic, PUFs have been widely adopted to build secure protocols. However, environmental conditions such as temperature variations may result in noise in a PUF's output. This is a limiting factor in the usage of PUFs as some of the bits in the PUF response may be incorrect for a particular challenge due to environmental conditions. The idea of a fuzzy extractor has been introduced to address such noise issues.

A $(d, n, l, h, \epsilon)$-secure PUF satisfies the following conditions [26]:

- Let $PUF_1$ and $PUF_2$ denote two PUFs and $P_i \in \{0,1\}^l$ denote the input. Then, $Pr[HD(PUF_1(P_i), PUF_2(P_i)) > d] \geq 1 - \epsilon$, where $HD$ is the Hamming distance.
(The inter-distance, i.e., the distance between the PUF outputs from two PUF devices for the same input is more than $d$.)
- Let $P_i$ be an input applied to a PUF, $PUF_1$. Let $PUF_1(P_i)$ and $PUF_1^*(P_i)$ be PUF responses at two instances from $PUF_1$. Then, $Pr[HD(PUF_1(P_i), PUF_1^*(P_i)) < d] \geq 1 - \epsilon$.
(The intra-distance, i.e., the distance between two PUF outputs from the same PUF device for the same input is less than $d$.)
- For a PUF, $PUF_1$, and for the inputs $P_1$, $P_2$, $\cdots$, $P_n$, $Pr[\widetilde{H}_\infty(PUF_1(P_i), PUF_1(P_j))_{1 \leq i,j \leq n, i \neq j} > h] \geq 1 - \epsilon$.
(The min-entropy of the PUF is always larger than $h$ with a high probability when the intra-distance is less than $d$ and the inter-distance is more than $d$.)

**Reconfigurable PUF:** An adversary may collect the challenge-response pairs of a PUF to create a machine learning model that can help predict the PUF's response to a new challenge. To address this issue of machine learning or modeling attack, we use a reconfigurable PUF in our

protocol that can change the PUF configuration [12], [33]. Reconfiguration allows updating the state of the PUF in such a way that the PUF responds differently to the same challenge before and after reconfiguration. We reconfigure the PUF after each authentication session. Since the challenge-response pairs of the PUF before the reconfiguration and the machine learning model built with it become obsolete after the PUF reconfiguration, the reconfiguration of the PUF eliminates the possibility of modeling attacks.

While reconfiguring a PUF, its state is updated [12], [33]. After reconfiguration, the security properties of the original PUF such as unclonability will still hold. However, after the reconfiguration, the PUF behaves as a new PUF in terms of the challenge and response pairs, making it difficult to create a mathematical model of the PUF. As an example, the D-PUF, an intrinsically reconfigurable Dynamic Random Access Memory (DRAM) based PUF [12] achieves reconfigurability by modifying the DRAM refresh-pause interval to change the challenge-response behaviour of the PUF.

The operation of a DPUF can be explained as follows [34]. In a DRAM PUF, reconfigurability is achieved by changing the DRAM refresh-pause interval. This will change the challenge-response behaviour of the PUF [12]. DRAM modules store data in 'bit-cells'. These cells are made up of capacitors and access transistors and can have a value of '0' or '1', based on the capacitor's charge state. Since these capacitors leak electrical charge gradually, they must be periodically refreshed to ensure data integrity. The interval between refreshes is called the refresh-pause interval. A D-PUF increases the refresh-pause interval causing random errors as bit-flips in some of the cells. Hence, when a bit string is stored in a block in a DRAM module and the refresh-pause interval is applied, the result will be a new bit string. Hence, varying the refresh-pause interval results in a new challenge-response pattern of the PUF, which is equivalent to reconfiguring the PUF.

### C. System Model

The system model consists of a key fob ($KF$) with the user and a receiver in the car. The $KF$ comes with at least two buttons to execute the lock and unlock operations. When the $KF$'s buttons are pressed, RF signals in the 315 MHz, 433 MHz, or 868 MHz band are generated and transmitted to the car receiver. After receiving the signal from the $KF$, the corresponding lock or unlock operation is triggered in the car.

### D. Adversary Model

In our adversary model, the adversary can intercept, record, jam, or replay the signals exchanged between the key fob and the car. The adversary may also execute a man-in-the-middle attack and modify the messages. The adversary may try to drop the messages between the car and the key fob to desynchronize the PUF challenge-response pairs and this will result in a Denial-of-Service (DoS) attack. If the parameters used in the proposed protocol such as the nonces and the PUF challenges are the same in every iteration of the protocol and not random, the signals exchanged will be the same always. Then, the adversary may capture the signals and send them later to unlock the car.

**Replay Attack:** Static code-based RKE systems always send the same signal to the car receiver. An adversary can launch a replay attack against such RKE systems by capturing a signal and transmitting it later to unlock the car. RKE systems that work on rolling codes are resilient to such replay attacks. However, they are still prone to the RollJam attack.

**RollJam Attack:** When a user presses the unlock button on the key fob, the adversary captures and saves the signal. Simultaneously, the adversary jams the unlock signal, preventing it from reaching the car. Since the first attempt was unsuccessful, the user tries to unlock the car again. The adversary captures, saves, and jams the signal again during the user's second attempt. At the same time, the adversary sends the first recorded signal to the car. Though the second signal is jammed, the car gets unlocked when the adversary replays the first captured signal. Through this RollJam attack, the attacker now has access to the most recent signal for unlocking the car. Here, the user does not realize that the key fob signals were captured and the RKE system was compromised. Instead, he/she thinks that the key fob did not work on the first attempt.

**RollBack Attack:** RollBack is another attack against RKE systems [6] that further reduces the attacker's effort. When the user presses the key fob button to unlock the car, the attacker captures, saves, and jams the signal. The car remains locked since it did not receive the unlock signal. Hence, the user presses the unlock button again. The next step in this attack is different from that in the RollJam attack. The attacker captures and saves the second signal as well but without jamming it. It must be noted that the attacker does not have to send the first recorded signal to the car as well. Later, by replaying the consecutive captured signals, the adversary can unlock the car at any time in some car models [6].

According to the researchers who devised the RollBack attack in [6], the exact root cause of the vulnerability that results in this attack has not been identified yet as the datasheets for RKE systems from all manufacturers are not available. They investigated the Microchip KeeLoq system since its datasheets were available. Whenever a new key fob is used, there is a key fob learning process in Microchip KeeLoq systems. In this process, with the first signal from the key fob, the key fob's serial number and rolling code counter are stored in the car receiver and the car receiver authenticates the signal. Then, the receiver waits for the second signal. With the second signal, the receiver checks whether the counter value is incremented. If this verification is successful, the receiver stores the current synchronization counter and other parameters. Then, it exits the learning mode. After that, the added key fob can be used in the future. Vehicles that are vulnerable to RollBack may be always in the learning mode without exiting [6]. In other words, when two consecutive signals are received, the car door gets unlocked if the counter value is incremented in the second signal. Also, before adding a key fob, it is not checked whether the serial number of the key fob already exists in the memory, i.e.,

whether it is an already added key fob. In that case, when the receiver receives two consecutive signals by replaying old signals from an already learned key fob, it allows adding that key fob again [6].

**Queries to Model the Attacks:** The queries that can be used to model the attacks performed by an adversary $A$ are given below:

***Record(m)***: Models $A$'s ability to capture a message *m*.

***Send(dest, m)***: Models the case when $A$ sends a message *m* to the destination $dest$ where $dest \in \{keyfob, receiver\}$.

***Jam(m)***: Models the query when $A$ jams a message *m*.

### *E. Security Goals*

The security goals of the proposed protocols are listed below:

**Authentication:** Before unlocking the car, the key fob and the car receiver must authenticate each other. This authentication prevents the scenario where an adversary unlocks the car.

**Integrity:** The integrity of messages must not be compromised. Both the key fob and the receiver should be able to confirm that the received messages have not been tampered with. If the messages are tampered with, the authentication process should be terminated.

**Resilience Against Common Attacks:** The proposed protocol should protect RKE systems from common attacks such as man-in-the-middle, replay, RollJam, and RollBack attacks.

## III. Proposed Authentication Protocol

In this section, we present the proposed mutual authentication protocol for RKE systems. The notations used in the proposed protocol and their descriptions are summarised in Table I.

TABLE I: Summary of notations

| Notation | Description |
|---|---|
| $ID_x$ | ID of a key fob |
| $PUF$ | Physical Unclonable Function |
| $P_i$ | An input challenge to PUF |
| $Q_i$ | Response from PUF |
| $(P, Q)$ | A challenge and the corresponding response of PUF |
| $FE$ | Fuzzy extractor |
| $A$ | Adversary |
| $N_i, N_k, N_c$ | Nonce values |
| $sk$ | A key shared between the key fob and the receiver |
| $\|$ | Concatenation operation |
| $\oplus$ | XOR operation |
| $H(X)$ | Hash of X |
| $M_n$ | $n^{th}$ Authentication message |

### *A. Assumptions*

The following are the assumptions made in this paper:

• Each $KF$ is equipped with a PUF that meets the conditions mentioned in Section II-B.

• The PUF output is unique and cannot be predicted [8].

• An attempt to tamper with the PUF makes it unusable [35].

### *B. Proposed Protocol*

The proposed protocol consists of two phases: setup and authentication. The setup phase is carried out only once. We assume that the $KF$ and the car receiver exchange messages through a secure channel during the setup phase. When the user presses the unlock button, the authentication process is initiated.

**Setup Phase:**

**Step 1:** The car receiver sends a challenge $P$ to the $KF$.

**Step 2:** Upon receiving $P$, the $KF$ generates the PUF output $Q$ for the input $P$ and sends its ID ($ID_x$), $Q$, and a reference parameter $ref$ to the receiver. The parameter $ref$ is required to denote the reconfiguration settings of the PUF when $Q$ is generated as the PUF is reconfigured during each authentication event. In a DRAM PUF, the refresh-pause interval is modified to reconfigure the PUF. The parameter $ref$ corresponds to the refresh-pause interval when $Q$ is generated.

**Step 3:** The car receiver generates a key $sk$ using a pseudorandom number generator and sends it to the $KF$. The car receiver stores $\{ID_x, (P, Q, ref), sk\}$ for further communication with the $KF$. The $KF$ stores $sk$.

**Authentication Phase:**

**Step 1:** When the user presses the $KF$ button, a nonce $N_i$ is generated at the $KF$. Then, the $KF$ computes $N_i^* = N_i \oplus sk$ and composes a message $M_1 = \{ID_x, N_i^*\}$ with its ID ($ID_x$) and $N_i^*$. After that, $KF$ sends $M_1$ to the car receiver indicating that it wants to communicate with the car receiver.

**Step 2:** The car receiver receives $M_1$. Then, it finds the challenge ($P_i$), the corresponding response ($Q_i$), the reference parameter ($ref_i$), and $sk$ for the $KF$ with ID, $ID_x$. The car receiver computes $N_i = N_i^* \oplus sk$ and generates a nonce $N_c$. Then, it calculates $M_c = (N_c \parallel N_i \parallel ref_i) \oplus sk$. Subsequently, it computes the Message Authentication Code (MAC) as $A_c = MAC(M_c \parallel N_i \parallel sk \parallel N_c \parallel P_i)$. We use the MAC function to confirm the integrity of the exchanged messages. The parameters $M_c$, $N_i$, and $sk$ in the MAC function ensure data integrity. The parameter $N_c$ ensures the freshness of the message. Then, it computes $P_i^* = P_i \oplus N_c$. After that, the receiver composes $M_2 = \{A_c, M_c, P_i^*\}$ and sends it to the $KF$.

**Step 3:** After receiving $M_2$, the $KF$ computes $M_c \oplus sk$ to extract $N_c$ and $ref_i$. Then, the $KF$ computes $P_i = P_i^* \oplus N_c$. After that, the $KF$ checks the message's integrity (i.e., the message is not modified) by verifying the received MAC. Then, it verifies that the received value, $ref_i$, corresponds to the current reconfiguration settings of the PUF. If $ref_i$ does not match the current reconfiguration settings, it indicates that there might have been an attempt by the adversary to desynchronize the PUF challenge-response pair. In that case, the reconfiguration settings of the PUF are changed to that corresponding to the received value, $ref_i$. Then, the $KF$ generates the PUF response $Q_i^{'}$ for the received challenge as $Q_i^{'} = PUF(P_i)$. After that, the $KF$

TABLE II: Authentication phase

| Key fob | | Car Receiver |
|---|---|---|
| Generate: $N_i$<br>$N_i^* = N_i \oplus sk$<br>$M_1 = \{ID_x, N_i^*\}$ | $\xrightarrow{M_1}$ | Find: $(P_i, Q_i, ref_i)$, and $sk$ corresponding to $ID_x$<br>$N_i = N_i^* \oplus sk$<br>Generate: $N_c$<br>$M_c = (N_c \parallel N_i \parallel ref_i) \oplus sk$<br>Compute: $A_c = MAC(M_c \parallel N_i \parallel sk \parallel N_c \parallel P_i)$<br>$P_i^* = P_i \oplus N_c$<br>$M_2 = \{A_c, M_c, P_i^*\}$ |
| Compute: $M_c \oplus sk$ to extract $N_c$ and $ref_i$<br>Compute: $P_i = P_i^* \oplus N_c$<br>Verify MAC<br>Verify: $ref_i$<br>Compute: $Q_i^{'} = PUF(P_i)$<br>Generate: $N_k$<br>$(k_i, hd_i) = FE.G(Q_i')$<br>$hd_i^* = hd_i \oplus sk$<br>Reconfigure: PUF for the $(i+1)th$ session<br>Compute:<br>$P_{i+1} = H(N_i \parallel N_k \parallel N_c)$<br>$Q_{i+1} = PUF(P_{i+1})$<br>$Q_{i+1}^* = Q_{i+1} \oplus sk$<br>$M_k = (N_c \parallel N_k \parallel Q_{i+1} \parallel ref_{i+1}) \oplus sk$<br>$cmd^* = cmd \oplus N_c$<br>$A_k = MAC(M_k \parallel N_k \parallel k_i \parallel cmd \parallel hd_i)$<br>$M_3 = \{A_k, M_k, cmd^*, Q_{i+1}^*, hd_i^*\}$ | $\xleftarrow{M_2}$ | |
| | $\xrightarrow{M_3}$ | Compute:<br>$hd_i = hd_i^* \oplus sk$<br>$Q_{i+1} = Q_{i+1}^* \oplus sk$<br>$cmd = cmd^* \oplus N_c$<br>Compute: $M_k \oplus sk$ to extract $N_k$ and $ref_{i+1}$<br>$k_i = FE.R(Q_i, hd_i)$<br>Verify: MAC<br>Execute: $cmd$<br>$P_{i+1} = H(N_i \parallel N_k \parallel N_c)$<br>Store: $P_{i+1}$, $Q_{i+1}$, $ref_{i+1}$ |

generates a nonce $N_k$. Subsequently, the $KF$ generates $k_i$ and $hd_i$ as $(k_i, hd_i) = FE.G(Q_i')$ and computes $hd_i^* = hd_i \oplus sk$. Then, it reconfigures the PUF as mentioned in Section II-B for the next authentication event. Then, the $KF$ generates the next challenge $P_{i+1} = H(N_i \parallel N_k \parallel N_c)$ and the corresponding response $Q_{i+1} = PUF(P_{i+1})$ using the reconfigured PUF. The parameter $ref_{i+1}$ denotes the reconfiguration settings of the PUF when $Q_{i+1}$ is generated.After that, the $KF$ computes $Q_{i+1}^* = Q_{i+1} \oplus sk$ and $M_k = (N_c \parallel N_k \parallel Q_{i+1} \parallel ref_{i+1}) \oplus sk$. Let the $cmd$ parameter specify the 'lock' or 'unlock' operation to be carried out in the car. The $KF$ computes $cmd' = cmd \oplus N_c$. Then, the $KF$ calculates the MAC as $A_k = MAC(M_k \parallel N_k \parallel k_i \parallel cmd \parallel hd_i)$. Finally, the $KF$ composes $M_3 = \{A_k, M_k, cmd^*, Q_{i+1}^*, hd_i^*\}$ and sends it to the car.

**Step 4:** When the receiver receives $M_3$, it computes $hd_i = hd_i^* \oplus sk$, $Q_{i+1} = Q_{i+1}^* \oplus sk$, $cmd = cmd^* \oplus N_c$, $M_k \oplus sk$, and $k_i = FE.R(Q_i, hd_i)$. Then, the receiver checks the integrity of the received message by verifying the MAC. If the MAC verification is successful, then the operation corresponding to the received command, $cmd$, is performed. Finally, the receiver computes $P_{i+1} = H(N_i \parallel N_k \parallel N_c)$ and stores $(P_{i+1}, Q_{i+1}, ref_{i+1})$.

The steps involved in the authentication phase are illustrated in Table II.

**Remark 1:** Consider the scenario when the user has accidentally pressed the $KF$ button when he/she is not in the vicinity of the car. Pressing a button on the $KF$ triggers the authentication process. The $KF$ sends the first message $M_1$. As the receiver does not receive $M_1$, the subsequent message $M_2$ is not sent by the receiver to the $KF$ and the authentication process will terminate. As a result, the message $M_3$ is also not sent by the $KF$. The car receiver still has valid parameters to use in the next round. Hence, even if the user accidentally presses the $KF$ button when he/she is not near the car, it will not affect the next valid attempt to unlock the car.

**Remark 2:** Nonces $N_i$, $N_c$, and $N_k$ used in the protocol are random numbers. They can be generated by pseudo-random number generators (PRNGs). PRNGs produce a sequence of random numbers determined by inputs called seed values. We assume that the key fob and the receiver are installed with seed values that are secure and not known to an adversary. Hence, the nonces are unpredictable [36]. Note that the security of the protocol depends on the nonces generated. Hence, nonces with high entropy should be used.

PRNGs must be seeded with sufficient entropy from reliable sources such as non-deterministic physical processes or unpredictable events [37]. Noise sources such as system data or human input can be used to generate the required entropy [37]. The National Institute of Standards and Technology (NIST) provides recommendations for generating random numbers using PRNGs in [37]. We recommend following the guidelines in [37] to generate nonces with high entropy.

**Remark 3:** Message authentication codes are used in the proposed protocol to ensure the integrity of the exchanged messages. The Keyed-Hash Message Authentication Code (HMAC) can be used for this purpose. The HMAC specification is given in RFC 2104 [38]. The HMAC function takes the symmetric key ($sk$) and the data string as inputs and produces the HMAC output value at the receiver. This is sent to the $KF$. The $KF$ computes the HMAC value using the data string and the same symmetric key and verifies whether the received HMAC value is correct or not. The same steps are repeated when the message is sent from the $KF$ to the receiver.

## IV. BAN LOGIC ANALYSIS OF THE PROPOSED PROTOCOL

In this section, we present a formal security analysis of the proposed protocol using BAN logic [39].

### A. Symbols and Rules

The symbols used in BAN logic analysis are provided in Table III.

TABLE III: Symbols used in BAN logic analysis

| Symbol | Meaning |
|---|---|
| $C \mid\equiv S$ | C believes the statement $S$ |
| $C \triangleleft S$ | C sees $S$ |
| $C \mid\sim S$ | C once said $S$ |
| $C \mid\Rightarrow S$ | C controls the statement $S$ |
| $\#(S)$ | $S$ is fresh |
| $C \overset{(k)}{\longleftrightarrow} D$ | A secret $k$ is shared between C and D |
| $\{X\}_Y$ | $X$ is encrypted using $Y$ |

The BAN logic rules used in the analysis of the proposed protocol are given below [26]:

1. Message-meaning rule R_1: $\frac{C\mid\equiv C\overset{(k)}{\longleftrightarrow}D, C\triangleleft\{S\}_k}{C\mid\equiv D\mid\sim S}$.

If C believes that C and D share a key $k$, and C sees a message $S$ encrypted with $k$, then C believes that D once said $S$.

2. Nonce-verification rule R_2: $\frac{C\mid\equiv\#(S), C\mid\equiv D\mid\sim S}{C\mid\equiv D\mid\equiv S}$.

If C believes that $S$ is fresh and that D once said $S$, then C believes that D believes $S$.

3. Jurisdiction rule R_3: $\frac{C\mid\equiv D\mid\Rightarrow S, C\mid\equiv D\mid\equiv S}{C\mid\equiv S}$.

If C believes that D controls $S$ and C believes that D believes $S$, then C believes $S$.

4. Seeing rule R_4: $\frac{C\triangleleft(X,Y)}{P\triangleleft X}$.

When C sees a concatenated message in which $X$ is a part, then C also sees $X$.

Seeing rule R_5: $\frac{C\mid\equiv C\overset{(k)}{\longleftrightarrow}D, C\triangleleft\{X\}_k}{C\triangleleft X}$.

If C and D share a key $k$, and C sees a message $X$ encrypted with $k$, then C also sees $X$.

5. Fresh rule R_6: $\frac{C\mid\equiv\#(X)}{C\mid\equiv\#(X,Y)}$.

If C believes a part $X$ of a formula is fresh, then C believes the whole formula that contains $X$ is fresh.

6. Belief rule R_7: $\frac{C\mid\equiv(X,Y)}{C\mid\equiv X}$.

If C believes a concatenated message in which $X$ is a part, then C believes $X$.

### B. BAN Logic Analysis of the Proposed Protocol

Through BAN logic analysis of the proposed protocol, we demonstrate that $N_c$, $N_k$, and $Q_{i+1}$ are shared secrets only between the $KF$ and the car receiver.

**Assumptions:**

The initial security assumptions about the $KF$ and the car receiver are given below:

$A_1$: $KF \mid\equiv \#(N_i)$: The $KF$ generates a fresh $N_i$ each time.

$A_2$: $Receiver \mid\equiv \#(N_c)$: The receiver generates a fresh $N_c$ each time.

$A_3$: $KF \overset{(sk)}{\longleftrightarrow} Receiver$: The receiver and the $KF$ share $sk$.

**Proof using BAN logic:**

When KF receives $M_2$, KF sees $M_c$:

$$P_1 : KF \triangleleft M_c.$$

Since only the receiver and the KF share $sk$ and $sk$ is used to construct $M_c$, by applying the message-meaning rule with $A_3$ and $P_1$:

$$P_2 : KF \mid\equiv Receiver \mid\sim M_c.$$

Since $N_i$ is a part of $M_c$ and $N_i$ is fresh, using the fresh rule R_6 with A1, we get:

$$P_3 : KF \mid\equiv \#(M_c).$$

Using the rule R_2 on $P_2$ and $P_3$, we can write:

$$P_4 : \frac{KF\mid\equiv\#(M_c), KF\mid\equiv Receiver\mid\sim M_c}{KF\mid\equiv Receiver\mid\equiv M_c}.$$

Using the rule R_7 on $P_4$, we get:

$$P_5 : \frac{KF\mid\equiv Receiver\mid\equiv(M_c, N_c)}{KF\mid\equiv Receiver\mid\equiv N_c}.$$

Hence, the $KF$ believes $N_c$ is received from the car receiver.

Similarly, When the car receiver receives $M_3$, the receiver sees $M_k$:

$$P_6 : Receiver \triangleleft M_k.$$

Since only the receiver and the KF share $sk$ and $sk$ is used to construct $M_k$, by applying the message-meaning rule with $A_3$ and $P_6$:

$$P_7 : Receiver \mid\equiv KF \mid\sim M_k.$$

Since $N_c$ is a part of $M_k$ and $N_c$ is fresh, using the fresh rule R_6 with A2, we get:

$$P_8 : Receiver \mid\equiv \#(M_k).$$

Using the rule R_2 on $P_7$ and $P_8$, we can write:

$$P_9 : \frac{Receiver\mid\equiv\#(M_k), Receiver\mid\equiv KF\mid\sim M_k}{Receiver\mid\equiv KF\mid\equiv M_k}.$$

Using the rule R_7 on $P_9$, we get:

$$P_{10} : \frac{Receiver|\equiv KF|\equiv(M_k,N_k)}{Receiver|\equiv KF|\equiv N_k}.$$

Hence, the car receiver believes $N_k$ is received from the $KF$.

Similarly, applying the belief rule R_7 with $P_9$, we also get:

$$P_{11} : \frac{Receiver|\equiv KF|\equiv(M_k,Q_{i+1})}{Receiver|\equiv KF|\equiv Q_{i+1}}.$$

Hence, the car receiver believes $Q_{i+1}$ is received from the $KF$.

Thus, we prove that $N_c$, $N_k$, and $Q_{i+1}$ are shared only between the $KF$ and the car receiver. This proves that the proposed authentication protocol is secure.

## V. Security Analysis

In this section, we present a formal security analysis and informal security analysis of the proposed protocol. Then, we compare the security properties of the proposed protocol with that of other protocols for RKE authentication.

### A. Formal Security Model and Analysis

We follow the security model in [40] to analyze the security of the proposed authentication protocol.

**Security Model:** Initially, a setup algorithm $Setup(1^k)$ is executed to generate a public parameter $p$ and a secret parameter $s$. The public parameter $p$ represents available public parameters such as the PUF output length and $s$ represents the PUF outputs. The setup phase is executed over a secure channel. However, the messages are exchanged over an insecure channel during the authentication phase. The $KF$ and the car receiver output either 1 (i.e., authentication is accepted) or 0 (i.e., authentication is rejected) as the outcome of the authentication. A communication session between the $KF$ and the receiver is represented by the session identifier $sid$. A session has a matching session if the messages are honestly transferred between the $KF$ and the receiver. Security of the protocol requires that the authentication results for both the $KF$ and the receiver become 1 if and only if the messages are transferred to each other without any man-in-the-middle attacks. From our adversary model, the adversary can control the communication channel between $KF$ and the receiver and can modify the exchanged messages.

TABLE IV: Security game

| |
|---|
| $(p, s) \leftarrow Setup(1^\lambda)$; |
| $(sid^*, M) \leftarrow A^{Launch,Send,Result,Reveal}(p, Receiver, KF)$; |
| $b :\leftarrow Result(sid^*, M)$; |
| $Output : b$ |

We consider a security game, $Game^{Sec}_{\pi,A}(\lambda)$, between a challenger $C$ and an adversary $A$ to analyze whether the proposed protocol $\pi$ provides security from man-in-the-middle attacks. $Game^{Sec}_{\pi,A}(\lambda)$ is illustrated in Table IV. After the first step Setup (), $A$ interacts with the $KF$ and the car receiver through the following oracle queries:

- $Launch(1^\lambda)$: Launch a new session.
- $Send()$: Send a message to the car receiver or the key fob.
- $Result(M; sid^*)$: Outputs whether the session $sid^*$ of $M$ is accepted or rejected where $M \in \{KF, receiver\}$.
- $Reveal(KF)$: Outputs the whole information stored in the memory of $KF$.

The advantage of $A$ against the protocol $\pi$, $Adv^{Sec}_{\pi,A}(\lambda)$ is the probability that the security game $Game^{Sec}_{\pi,A}(\lambda)$ outputs 1 when $sid^*$ of $M$ does not have a matching session.

***Definition 1***: An authentication protocol $\pi$ is secure against man-in-the-middle attacks if for any probabilistic polynomial time adversary $A$, if $Adv^{Sec}_{\pi,A}(\lambda)$ is negligible, i.e., $Adv^{Sec}_{\pi,A}(\lambda) \leq \epsilon$ for large $\lambda$.

***Theorem 1***: If the proposed protocol employs a $(d, n, l, h, \epsilon)$-secure PUF and a $(d, h, \epsilon)$-secure fuzzy extractor, it is robust against man-in-the-middle attacks.

*Proof.* The aim of the adversary $A$ is to win the security game $Game^{Sec}_{\pi,A}(\lambda)$. $A$ wins the game if the key fob or the car receiver accepts the session without a matching session, i.e., even if the messages are modified by $A$. For the proof, we consider a series of game transformations. Let $S_i$ be the advantage that $A$ wins the game, Game $i$. The game transformations are given below:

**Game 0.** This is the original game between the challenger and $A$.

**Game 1.** The challenger starts this game. The challenger aborts the game if the adversary cannot impersonate $KF$ to the car receiver or vice versa.

**Game 2.** Let $A$ establish up to $x$ sessions in the game. For $1 \leq l \leq x$, we modify the parameters of the session between the $KF$ and the receiver as given below:

- Game 2(l, 1): In the $l$-th session, the challenger evaluates the PUF response. $C$ aborts the game under the following conditions: the intra-distance $> d$, inter-distance $< d$, or if the PUF output does not have enough entropy.
- Game 2(l, 2): The fuzzy extractor output is changed to a random string in this game.

The strategy used in the security proof is to change exchanged messages to random strings, making it difficult for an adversary to distinguish the random strings from the exchanged messages. The adversary wins the game if he/she can differentiate between real messages and random strings or is successful in making the $KF$ or the car receiver accept the session even after modifying the messages. After completing the game transformations from Game 2(l, 1) to Game 2(l, 2), we apply this strategy recursively to the next sessions until the upper limit of $x$. The advantage of $A$ is negligible as shown through the following Lemmas:

- **Lemma 1.** $S_0 = S_1$.
  *Proof.* If $A$ wins the game, the $KF$ or the receiver accepts at least one session even though $A$ has modified the exchanged messages in that session. Hence, we get:
  $$S_0 = S_1. \tag{3}$$
- **Lemma 2.** If a $(d, n, l, h, \epsilon)$-secure PUF is used, then $S_1 = S_{2(l,1)}$ for any $1 \leq l \leq x$.
  *Proof.* Since we use $(d, n, l, h, \epsilon)$-secure PUF, its intra-distance is less than $d$ and its inter-distance is more than $d$. Also, since the PUF has min-entropy, its responses

are not correlated. Since the games in $S_1$ and $S_{2(l,1)}$ are based on the above conditions and the gap between them is bounded by $\epsilon$, we can write:

$$| S_1 - S_{2(l,1)} | \leq \epsilon. \quad (4)$$

- **Lemma 3.** If we use a $(d, h, \epsilon)$-secure fuzzy extractor, then an adversary cannot differentiate the games $S_{2(l,1)}$ and $S_{2(l,2)}$ for $1 \leq l \leq x$.
  *Proof.* The PUF used in the protocol has a min-entropy. Hence, the fuzzy extractor output, which is based on the output of the PUF, is a random string. Hence, an adversary cannot distinguish between $Game2(l, 1)$ and $Game2(l, 2)$. Therefore, the advantage of the adversary in distinguishing $Game2(l, 1)$ and $Game2(l, 2)$ can be written as:
  $$| S_{2(l,1)} - S_{2(l,2)} | \leq \epsilon. \quad (5)$$

We can see that the adversary has a negligible advantage in breaking the security when we transform $Game\ 0$ to $Game\ 2(x, 2)$ since these games are bounded by assumptions of secure PUF and fuzzy extractor. Combining (3), (4), and (5), we get $Game^{Sec}_{\pi,A}(\lambda) < 2\epsilon$ which is negligible. Hence, the adversary's advantage in executing man-in-the-middle attacks against the proposed protocol is negligible.

### *B. Informal Security Analysis*

In this subsection, we present the informal security analysis.

- **Mutual Authentication:** The $KF$ and the car receiver authenticate each other by verifying the integrity of the messages using the MAC. The $KF$ authenticates the car receiver by verifying the parameter $A_c = MAC(M_c \parallel N_i \parallel sk \parallel N_c \parallel P_i)$ in the message $M_2$. Similarly, the car receiver authenticates the $KF$ by verifying $A_k = MAC(M_k \parallel N_k \parallel k_i \parallel cmd \parallel hd_i)$ in the message $M_3$. Only a legitimate $KF$ and car receiver know $k_i$ and $sk$ to construct the correct parameters $A_k$ and $A_c$. Thus, the proposed protocol ensures mutual authentication between the $KF$ and the car receiver.
- **Protection Against Replay Attacks:** During each authentication event, the nonces used by the $KF$ and the car receiver ($N_i$, $N_c$, and $N_k$) are changed. The challenge-response pair of the PUF is also updated during each authentication event. Suppose an adversary captures $M_1$ and $M_3$ from a session. When an adversary sends the captured message $M_1$ in an attempt to execute the replay attack, the receiver sends a message back $M_2$. $M_2$ is based on the latest PUF challenge $P_i$ and a nonce $N_c$. The message $M_3$ that the adversary has captured contains parameters corresponding to the PUF challenge and $N_c$ in the previous authentication iteration. If the adversary replays the captured $M_3$, the MAC verification will fail at the receiver. As a result, an attacker cannot get authenticated by replaying the previously captured messages. Thus, the proposed protocol provides protection from replay attacks.
- **Protection Against Impersonation Attacks:** To impersonate a legitimate car receiver, the attacker $A$ needs to construct $M_c = (N_c \parallel N_i \parallel ref_i) \oplus sk$. However, $A$ does not know $sk$ to construct $M_c$. Hence, $A$ cannot compose a valid message $M_2$ to impersonate the receiver. Similarly, to impersonate a legitimate $KF$, $A$ needs to construct $M_k = (N_c \parallel N_k \parallel Q_{i+1} \parallel ref_{i+1}) \oplus sk$. $A$ does not know $sk$ to construct $M_k$. Hence, $A$ cannot compose a valid message $M_3$ to impersonate the $KF$ as well. Thus, the proposed protocol offers protection from impersonation attacks.
- **Protection Against RollJam Attack:** In the RollJam attack against the original RKE systems, the attacker $A$ stores and jams two consecutive signals from the $KF$ to the car. While capturing the second signal, $A$ forwards the first stored signal to the car receiver. $A$ sends the second stored signal to unlock the car later. The car receiver accepts the unlock signal from the $KF$ and unlocks the car.
  Now, let us consider the scenario where $A$ attempts to carry out the RollJam attack against the proposed authentication protocol. $A$ records and jams two consecutive signals $M_1^i$ and $M_1^{i+1}$ in the $i$-th and $(i+1)$-th authentication iterations from the $KF$. $A$ also sends $M_1^i$ to the receiver while jamming $M_1^{i+1}$. When $A$ sends $M_1^i$ to the car receiver, the car receiver sends the corresponding message $M_2^i$ with the parameter $A_c = MAC(M_c \parallel N_i \parallel sk \parallel N_c \parallel P_i)$ to the key fob. The $N_i$ used by the car receiver in constructing $A_c$ corresponds to the first signal $M_1^i$ of the $i$-th round since $M_1^i$ is the signal received by the receiver. When the $KF$ receives $M_2^i$, it computes the MAC and verifies it against $A_c$ received in $M_2^i$. The value computed by the key fob is $MAC(M_c \parallel N_{i+1} \parallel sk \parallel N_c \parallel P_i)$ whereas the received $A_c = MAC(M_c \parallel N_i \parallel sk \parallel N_c \parallel P_i)$ since the $KF$ uses $N_{i+1}$ corresponding to the signal $M_1^{i+1}$ (which is the latest signal sent by the $KF$) and the receiver uses $N_i$ corresponding to the signal $M_1^i$ (which is the latest signal received by the receiver). Thus, the verification of $A_c$ will fail. Hence, the authentication process will terminate. Thus, the attacker's attempt to send the first signal by jamming the second signal will fail and the user will detect the presence of the attacker. As a result, the proposed protocol eliminates the RollJam attack against RKE systems.
- **Protection Against RollBack Attack:** In the RollBack attack, the attacker $A$ captures two consecutive signals from the $KF$ to the car. $A$ then sends the two captured signals to unlock the car at a later point in time. With the proposed protocol, when $A$ sends the captured signals to the car receiver to execute the RollBack attack, the car receiver sends $M_2$ to the attacker. $A$ cannot produce the PUF response $Q_i^{'}$ for the input challenge $P_i$ and hence cannot calculate $k_i$ and $hd_i$. $A$ will not be able to extract $N_c$ by doing $M_c \oplus sk$ as he/she does not know $sk$. As a result, $A$ cannot generate $A_k = MAC(M_k \parallel N_k \parallel k_i \parallel cmd \parallel hd_i)$ and $M_3 = \{A_k, M_k, cmd^*, Q^*_{i+1}, hd^*_i\}$ and hence cannot continue with the rest of the steps and the authentication process will terminate. Thus, the proposed protocol is

TABLE V: Comparison of security properties

| Features | Greene et al. [19] | Greene et al. [20] | Glocker et al. [22] | Parameswarath et al. [23] | Gade et al. [25] | Proposed Protocol |
|---|---|---|---|---|---|---|
| SP1 | Yes | Yes | Yes | Yes | Yes | Yes |
| SP2 | No | No | Yes | No | Yes | Yes |
| SP3 | Yes | Yes | Yes | Yes | Yes | Yes |
| SP4 | Yes | Yes | No | Yes | Yes | Yes |
| SP5 | No | No | No | No | Yes | Yes |
| SP6 | No | No | No | No | Yes | Yes |
| SP7 | No | No | Yes | No | Yes | Yes |
| SP8 | Yes | Yes | No | No | No | Yes |
| SP9 | Yes | Yes | Yes | Yes | No | Yes |
| SP10 | No | No | No | No | No | Yes |
| SP11 | No | No | No | No | No | Yes |
| SP12 | - | - | - | - | No | Yes |
| SP13 | - | - | - | - | No | Yes |
| **SP1:** Key fob Authentication; **SP2:** Mutual Authentication; **SP3:** Protection From Replay Attack; | | | | | | |
| **SP4:** Protection From RollJam Attack; **SP5:** Message Integrity; | | | | | | |
| **SP6:** Physical Security; **SP7:** Does not Require Clock Synchronization; **SP8:** Protection From Eavesdropping Attack; | | | | | | |
| **SP9:** Completeness (Provides both lock and unlock options); **SP10:** Addresses RollBack Attack; **SP11:** Security Proof; | | | | | | |
| **SP12:** Resilience to ML Attack; **SP13:** Resilience to Impact of Environmental Factors | | | | | | |

resilient to the RollBack attack.

- **Clock Synchronization is not Required:** To protect against replay attacks, a popular strategy employed in authentication protocols is to use timestamps. For the timestamp-based mechanism to work correctly, both the sender and the receiver should have synchronized clocks. The proposed protocol generates fresh nonces during each authentication event and does not depend on timestamps to confirm the freshness of messages. Hence, the proposed protocol does not require the $KF$ and the car receiver to synchronize their clocks.
- **Protection Against Physical and Cloning Attacks:** The PUF generates responses based on the input challenges. PUF responses are not stored in its memory. Hence, an attacker cannot get any PUF response from the PUF by doing a physical attack. The PUF will become useless if there is any physical tampering with it. Such a physically tampered PUF will not produce the correct response for a challenge sent by the car receiver. Further, PUFs are unclonable [41]. Hence, an attacker cannot clone a PUF to generate the PUF responses. Thus, the proposed protocol is resistant to physical and cloning attacks.
- **Protection Against Modeling or Machine Learning Attacks:**
  To carry out a machine learning attack, an adversary collects a large number of challenge-response pairs $\{(P_1, Q_1), (P_2, Q_2), \ldots, (P_n, Q_n)\}$ of the PUF. Then, he/she generates a model to predict a response $Q_{n+1}$ for a new challenge $P_{n+1}$. In the proposed protocol, the PUF is reconfigured during each iteration of key fob authentication using the reconfigurability feature (e.g., by changing the refresh-pause interval) of the PUF. Hence, the PUF's responses before and after reconfiguration for the same challenge will be different. As a result, the PUF responses collected before the reconfiguration and the model built with them are obsolete after the PUF reconfiguration. This will make $A$'s attempt to collect the PUF's challenge-response pairs to build a machine learning model ineffective. Hence, the proposed authentication protocol is robust against modeling attacks.
- **Resilience Against Desynchronization and DoS Attacks:**
  An adversary may attempt to carry out a DoS attack by desynchronizing the secrets (e.g., PUF challenge-response pair) between the $KF$ and the receiver by jamming the message $M_3$ sent by the $KF$. In the proposed protocol, after reconfiguring the PUF corresponding to the settings $ref_{i+1}$, the $KF$ computes the PUF response $Q_{i+1} = PUF(P_{i+1})$. When $M_3$ sent by the $KF$ is received by the receiver, $Q_{i+1}$ and $ref_{i+1}$ are updated at the receiver. Consider the scenario where an attacker jams the message $M_3$ sent by the $KF$ in an attempt to desynchronize the secrets between the $KF$ and the receiver. To protect the system from such desynchronization attacks, when the $KF$ receives the message $M_2$, it verifies that the received value, $ref_i$, corresponds to the current reconfiguration settings of the PUF. If $ref_i$ does not match the current reconfiguration settings of the PUF, it indicates that there might have been an attempt by the adversary to desynchronize the challenge-response pair between the $KF$ and the receiver by jamming $M_3$. In such a scenario, the reconfiguration settings of the PUF are changed to that corresponding to the received value, $ref_i$. Then, the $KF$ generates the PUF response corresponding to the received challenge and runs the $FE.G()$ function to find the key and the helper data. It does not store any of these parameters. Hence, even if the attacker drops the message $M_3$ so that the PUF challenge, the response, and the reconfiguration parameter will not be updated at the receiver, the synchronization of secrets between the $KF$ and the receiver will not be affected. Thus, the protocol is resilient against desynchronization and corresponding DoS attacks.

### C. Comparison of Security Properties

Next, we compare the proposed protocol with other existing schemes for RKE security based on the security properties it achieves. Table V provides an overview of

the comparison of the security properties. The protocols proposed in [19], [20], [22], [23], and [25] do not address the RollBack attack. The proposed protocol is the first protocol designed to safeguard against the RollBack attack. Though [19], [20], and [23] offer some important security features, the proposed protocol offers the additional property of mutual authentication. The proposed protocol also uses MACs to confirm the integrity of the messages whereas most other protocols do not ensure message integrity. With the proposed protocol, the $KF$ and the receiver can detect if the messages they receive have been modified by an adversary. The $KF$ and the receiver are required to synchronize their clocks for the protocols mentioned in [19], [20], and [23] to work correctly. The proposed protocol does not require the clocks in the $KF$ and the car to be synchronized with each other. The proposed protocol ensures physical security of the $KF$ as well by the use of PUFs, and this is not provided by [19], [20], [22], and [23]. The protocols in [22], [23], and [25] are vulnerable to eavesdropping attacks as the parameters are sent in plain text and the adversary may listen to them. Also, the protocol in [25] is not complete since it does not provide an option to mention the lock operation. Successful operation of the protocol proposed in [25] always results in the unlock operation. The proposed protocol is resilient to ML attacks against PUFs. Also, by using a fuzzy extractor, the protocol ensures that it is resilient to the noise in the PUF response due to environmental factors. The PUF-based protocol in [25] is not resilient against ML attacks and is susceptible to noise due to environmental factors. Thus, the proposed protocol offers better security features compared to other similar protocols for RKE authentication.

## VI. Performance Analysis and Comparison

In this section, we evaluate the computation cost of the proposed protocol and compare it with the computation cost of other protocols for RKE authentication.

### A. Computation Cost

First, we find the computation cost during the authentication phase.

TABLE VI: Number of operations and execution time

| **Operation** | **Key fob** | **Car Receiver** |
|---|---|---|
| $XOR$ | 7 | 7 |
| $PUF$ | 2 | 0 |
| $Concatenation$ | 9 | 8 |
| $Hash$ | 1 | 1 |
| $MAC$ | 2 | 2 |
| $FE.G()$ | 1 | - |
| $FE.R()$ | | 1 |
| $Reconfig()$ | 1 | - |
| **Computation Time using Python (ms)** | 0.83 | 0.34 |

Let us denote the number of hash, MAC, XOR, PUF, fuzzy extractor key generation, fuzzy extractor key reconstruction, reconfiguration, and concatenation operations as $N_h$, $N_{MAC}$, $N_x$, $N_{PUF}$, $N_{FEG}$, $N_{FER}$, $N_{re}$, and $N_c$, respectively. The $KF$ executes $7N_x$+$2N_{PUF}$+$9N_c$+$N_h$+$N_{FEG}$+$2N_{MAC}$+$N_{re}$ operations while the car receiver executes $7N_x$+$8N_c$+$N_h$+$2N_{MAC}$+$N_{FER}$ operations to complete one iteration of the authentication process. Two MAC operations are considered to account for the MAC generation and MAC verification.

To simulate the proposed protocol, the operations have been executed in Python on a Raspberry Pi. Let $t_h$, $t_{MAC}$, $t_x$, $t_{FEG}$, $t_{FER}$, and $t_c$ represent the time taken by hash, MAC, XOR, fuzzy extractor key generation, fuzzy extractor key reconstruction, and concatenation operations, respectively. From the simulations, $t_x$ is 10.9 µs, $t_h$ is 16.9 µs, $t_{MAC}$ is 86 µs, $t_{FEG}$ is 22 µs, $t_{FER}$ 43 µs, and $t_c$ is 4.9 µs. To deploy in the $KF$, we consider a PUF that gives a response of 128 bits in 0.21 µs. The calibration circuit can adjust the refresh time to get the required retention failure probability [42]. We can set the calibration circuit and set the DRAM refresh period to reconfigure the PUF. We consider the time to reconfigure the PUF $t_{recon}$ as 0.5 ms in our calculation. The proposed protocol executes two PUF operations in the $KF$ during each authentication event. However, physical security is achieved by using PUFs. Hence, the two PUF operations are justified.

Next, we compare the computation cost of the proposed protocol with protocols in [19], [20], [22], [23], and [25] for RKE security. There is one AES encryption operation in the $KF$ and one AES decryption operation in the car receiver in [19]. Note that we do not use AES encryption in the proposed protocol. Let us denote these encryption/decryption operations as $N_{enc}$. From the simulations, the time taken by encryption/decryption $t_{enc}$ is 0.93 ms. Similarly, there is one RSA signature generation in the $KF$ and one RSA signature verification in the receiver in [23]. Let $N_{sign}$, $N_{verify}$, $t_{sign}$, and $t_{verify}$ represent the number of RSA signature generation operations, number of signature verification operations, the time taken by RSA signing, and the time taken by the signature verification operation, respectively. From the simulation, $t_{sign}$ is 0.95 ms and $t_{verify}$ is 0.97 ms. A comparison of the computation cost of the proposed protocol and other protocols is given in Table VII. Similarly, the computation costs of the proposed protocol and other protocols are plotted in Figure 1. These results lead us to the conclusion that the proposed protocol provides additional security properties with reasonable computation cost compared to other existing RKE authentication protocols.

### B. Communication Overhead

Table VIII shows a comparison of the communication overhead of the proposed protocol with [19], [20], [22], [23], and [25]. We observe that the communication overhead of the proposed protocol is reasonable. The communication costs of protocols in [22] and [23] are higher than that of the proposed protocol. The communication costs of the proposed protocol and other protocols are plotted in Figure 2.

TABLE VII: Comparison of computation cost

| Scheme | Key fob | Receiver |
|---|---|---|
| **Greene et al. [19]** | $2t_c$+ $t_{enc}$ = 0.9398 ms | $t_{enc}$ = 0.93 ms |
| **Greene et al. [20]** | $t_x$+$t_{enc}$ = 0.94 ms | $t_x$+$t_{enc}$ = 0.9409 ms |
| **Glocker et al. [22]** | $6t_{enc}$ = 5.58 ms | $5t_{enc}$ = 4.65 ms |
| **Parameswarath et al. [23]** | $2t_c$+$t_{sign}$ = 0.9598 ms | $2t_c$+$t_{verify}$ = 0.9798 ms |
| **Gade et al. [25]** | $t_{PUF}$ +$2t_h$ = 0.034 ms | $t_{PUF}$+$2t_h$ = 0.034 ms |
| Proposed Protocol | $7t_x$+$2t_{PUF}$+$9t_c$+$t_h$+$t_{FEG}$+ +$2t_{MAC}$+$t_{recon}$ = 0.83 ms | $7t_x$+$8t_c$+$t_h$+ $2t_{MAC}$+$t_{FER}$ = 0.34 ms |

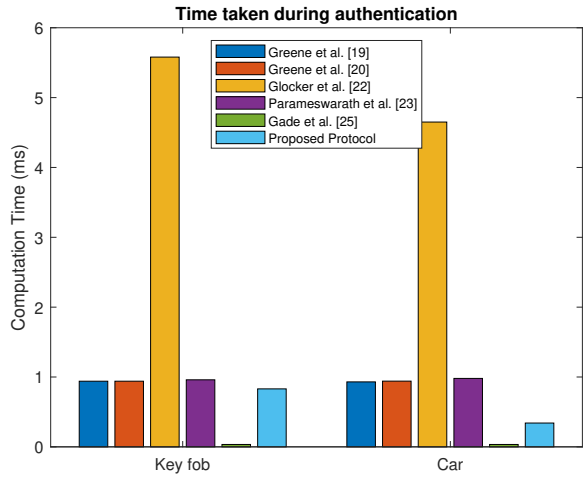


Fig. 1: Computation costs based on simulation in Python.

TABLE VIII: Communication overhead

| Protocol | Number of bytes |
|---|---|
| Greene et al. [19] | 16 |
| Greene et al. [20] | 16 |
| Glocker et al. [22] | 49 |
| Parameswarath et al. [23] | 129 |
| Gade et al. [25] | 22 |
| Proposed Protocol | 34 |

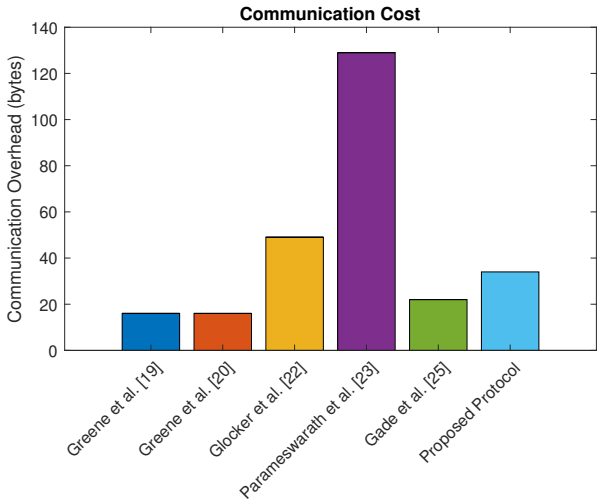


Fig. 2: Communication cost.

### C. Other Metrics

The energy/power consumption, area, and delay metrics of a PUF are vital points to be considered while using it in authentication protocols. We consider the DRAM PUF presented in [42] where a 64Kb DRAM PUF array was fabricated in a 1.2V, 65nm LP CMOS process. The circuit area is 350x300 µm and the energy consumption is 0.89 pJ/bit. This PUF can generate $10^{32}$ challenge-response pairs from a 1 Kbit array. Next, we consider the time required to reconfigure the PUF. After writing '0' and '1' to the DRAM, we can check if the bit has flipped after a certain retention time has elapsed. The storage node capacitance and the leakage current determine the retention time. The reference voltage can be modified so that a certain number of cells flip for a given retention time. The DRAM refresh period is 100 $\mu$s to 10 ms [42]. The calibration circuit can adjust the refresh time to get the required retention failure probability [42]. Hence, we can see that the area, power consumption, and reconfiguration time of PUFs are negligible. With the advancement in technology, these parameters will be reduced further.

## VII. Analysis of Robustness of Reconfigurable PUFs Against Machine Learning Attacks

In this section, we consider a DRAM PUF to analyze the robustness of reconfigurable PUFs against machine learning attacks. We employed the experimental settings of [33] for the analysis. A DRAM PUF generates entropy from power-up and cell refresh for transforming challenges into responses, together with the reconfiguration characteristic as mentioned in Section II-B. The PUF challenge-response pairs were collected from the Pandaboard ES Revision B3 [43]. It contains a Texas Instruments OMAP4460 processor with 8Gb/1GB POP LPDDR2 DRAM device (Elpida P/N EDB8064B1PB-8D-F) [43]. The DRAM can be accessed using two external memory interfaces. A Linux kernel module was implemented to modify the memory controller to turn off the DRAM refresh. After a certain time interval, the DRAM refresh was enabled again and the memory contents were collected. After the time interval, sufficient charge had leaked from some cells so that their logical bits were flipped. The positions of the flipped bits are unique and that act as the PUF response [44]. In total, 300 responses were collected.

The robustness of several PUFs against machine learning attacks was explored in [45]. The machine learning algorithm, Logistic Regression (LR), was very effective and accurately predicted the challenge-response pairs for different PUF models [45]. Hence, we modelled the DRAM PUF in Python

using LR. We also modelled it using Naive Bayes. We trained the models and tried to predict test data. Then, we calculated the accuracy. The accuracy of the predicted results in these two cases was very low at 18.14 % and 15.2 %, respectively. Hence, we can conclude that machine learning attacks on a reconfigurable PUF are not effective.

## VIII. Conclusion

Attackers have always targeted RKE systems since the attacks can be carried out easily on RKE systems by using relatively inexpensive hardware and software. In addition to the basic replay attack which can be applied against static code-based RKE systems, RollJam attacks against rolling code-based systems also have been reported in the literature. The RollBack attack, which further reduces the adversary's efforts, is a newly devised attack against RKE systems. Hence, it is crucial to protect RKE systems from these attacks. We proposed a PUF-based mutual authentication protocol that protects RKE systems from replay, RollJam, and RollBack attacks. To make the solution usable in a practical scenario, we considered certain factors while designing the protocol. The key fob is exposed to environmental elements such as temperature and hence the PUF response can be affected. We designed the protocol considering this fact and mitigating the effect of environmental influence by using the concept of fuzzy extractor. As it has been shown in the literature that PUF-based protocols are vulnerable to machine learning attacks, we designed the protocol to make it secure against such attacks as well. Further, we analyzed the reconfigurable PUF and its robustness against machine learning attacks. This analysis demonstrated that by reconfiguring the PUF, it becomes secure against modeling attacks. We also showed that the proposed protocol is lightweight and provides more security features compared to other existing schemes. To summarize, we demonstrated that PUF-based authentication protocol with careful design elements (such as modeling attack resistance and immunity to environmental conditions) is a promising solution to ensure the security of RKE systems.

## References


[1] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of can bus security challenges," *Sensors*, vol. 20, no. 8, p. 2364, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/8/2364

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*, vol. 4, no. 447-462. San Francisco, 2011, p. 2021.

[3] K. Joo, W. Choi, and D. H. Lee, "Hold the door! fingerprinting your car key to prevent keyless entry car theft," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/hold-the-door-fingerprinting-your-car-key-to-prevent-keyless-entry-car-theft/

[4] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlides, "Lock it and still lose it - on the (in) security of automotive remote keyless entry systems," in *Proceedings of the 25th USENIX Security Symposium*, 2016, pp. 929 – 944.

[5] Defcon 23 rolljam attack. [Online]. Available: https://samy.pl/defcon2015/

[6] L. Csikor, H. W. Lim, J. W. Wong, S. Ramesh, R. P. Parameswarath, and M. C. Chan, "Rollback: A new time-agnostic replay attack against the automotive remote keyless entry systems," *ACM Transactions on Cyber-Physical Systems*, vol. 8, no. 1, pp. 1–25, 2024. [Online]. Available: https://doi.org/10.1145/3627827

[7] Rollback - a new time-agnostic replay attack against the automotive remote keyless entry systems. [Online]. Available: https://infocondb.org/con/black-hat/black-hat-usa-2022/rollback-a-new-time-agnostic-replay-attack-against-the-automotive-remote-keyless-entry-systems

[8] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[9] M. N. Aman, U. Javaid, and B. Sikdar, "A privacy-preserving and scalable authentication protocol for the internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1123–1139, 2021.

[10] Y. Dodis, J. Katz, L. Reyzin, and A. Smith, "Robust fuzzy extractors and authenticated key agreement from close secrets," in *CRYPTO*, vol. 4117. Springer, 2006, pp. 232–250.

[11] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 237–249.

[12] S. Sutar, A. Raha, D. Kulkarni, R. Shorey, J. Tew, and V. Raghunathan, "D-puf: An intrinsically reconfigurable dram puf for device authentication and random number generation," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 17, no. 1, pp. 1–31, 2017.

[13] V. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2016, pp. 164–170.

[14] A. Chattopadhyay, K.-Y. Lam, and Y. Tavva, "Autonomous vehicle: Security by design," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2020.

[15] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," *Computers & Security*, p. 102269, 2021.

[16] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *black hat USA*, vol. 2014, p. 94, 2014.

[17] A. I. Alrabady and S. M. Mahmud, "Analysis of attacks against the security of keyless-entry systems for vehicles and suggestions for improved designs," *IEEE transactions on vehicular technology*, vol. 54, no. 1, pp. 41–50, 2005.

[18] R. Benadjila, M. Renard, J. Lopes-Esteves, and C. Kasmi, "One car, two frames: attacks on hitag-2 remote keyless entry systems revisited," in *11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17)*, 2017.

[19] K. Greene, D. Rodgers, H. Dykhuizen, K. McNeil, Q. Niyaz, and K. A. Shamaileh, "Timestamp-based defense mechanism against replay attack in remote keyless entry systems," in *2020 IEEE International Conference on Consumer Electronics (ICCE)*, 2020, pp. 1–4.

[20] K. Greene, D. Rodgers, H. Dykhuizen, Q. Niyaz, K. Al Shamaileh, and V. Devabhaktuni, "A defense mechanism against replay attack in remote keyless entry systems using timestamping and xor logic," *IEEE Consumer Electronics Magazine*, vol. 10, no. 1, pp. 101–108, 2020.

[21] Ultimate keeloq technology. [Online]. Available: https://www.microchip.com/en-us/solutions/wireless-connectivity/rf-remotes/ultimate-keeloq-technology

[22] T. Glocker, T. Mantere, and M. Elmusrati, "A protocol for a secure remote keyless entry system applicable in vehicles using symmetric-key cryptography," in *2017 8th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2017, pp. 310–315.

[23] R. P. Parameswarath and B. Sikdar, "An authentication mechanism for remote keyless entry systems in cars to prevent replay and rolljam attacks," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 1725–1730.

[24] R. P. Parameswarath and B. Sikdar, "A puf-based lightweight and secure mutual authentication mechanism for remote keyless entry systems," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 1776–1781.

[25] S. Gade, U. Chatterjee, and D. Mukhopadhyay, "Pakamac: A puf-based keyless automotive entry system with mutual authentication," *Journal of Hardware and Systems Security*, vol. 6, no. 3-4, pp. 67–78, 2022.

[26] P. Gope and B. Sikdar, "Lightweight and privacy-preserving two-factor authentication scheme for iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 580–589, 2019.

[27] T. A. Idriss, H. A. Idriss, and M. A. Bayoumi, "A lightweight puf-based authentication protocol using secret pattern recognition for constrained iot devices," *IEEE Access*, vol. 9, pp. 80 546–80 558, 2021.

[28] P. Gope and B. Sikdar, "An efficient privacy-preserving dynamic pricing-based billing scheme for smart grids," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–2.

[29] P. Gope and B. Sikdar, "An efficient privacy-preserving authenticated key agreement scheme for edge-assisted internet of drones," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 13 621–13 630, 2020.

[30] P. Gope, A. K. Das, N. Kumar, and Y. Cheng, "Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 4957–4968, 2019.

[31] M. Akgun and M. U. Caglayan, "Puf based scalable private rfid authentication," in *2011 Sixth International Conference on Availability, Reliability and Security*, 2011, pp. 473–478.

[32] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM journal on computing*, vol. 38, no. 1, pp. 97–139, 2008.

[33] P. Gope and B. Sikdar, "A privacy-aware reconfigurable authenticated key exchange scheme for secure communication in smart grids," *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5335–5348, 2021.

[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] M. S. Kirkpatrick, S. Kerr, and E. Bertino, "System on chip and method for cryptography using a physically unclonable function," U.S. Patent 8,750,502, Jun 10, 2014.

[36] "Pseudorandom number generator," [Accessed: Sep 2023]. [Online]. Available: https://csrc.nist.gov/glossary/term/pseudorandom_number_generator

[37] E. B. Barker, J. M. Kelsey *et al.*, *Recommendation for random number generation using deterministic random bit generators (revised)*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007.

[38] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," *IETF RFC 2104*, February 1997.

[39] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Trans. Comput. Syst.*, vol. 8, no. 1, p. 18–36, feb 1990. [Online]. Available: https://doi.org/10.1145/77648.77649

[40] A. Aysu, E. Gulcan, D. Moriyama, P. Schaumont, and M. Yung, "End-to-end design of a puf-based privacy preserving authentication protocol," in *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*. Springer, 2015, pp. 556–576.

[41] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[42] Q. Tang, C. Zhou, W. Choi, G. Kang, J. Park, K. K. Parhi, and C. H. Kim, "A dram based physical unclonable function capable of generating $>10^{32}$ challenge response pairs per 1kbit array for secure chip authentication," in *2017 IEEE Custom Integrated Circuits Conference (CICC)*, 2017, pp. 1–4.

[43] "Pandaboard System Reference Manual," Online,https://www.cs.utexas.edu/~simon/378/resources/PandaBoardES.pdf, [Accessed: Sep 2023].

[44] A. Schaller, W. Xiong, N. A. Anagnostopoulos, M. U. Saleem, S. Gabmeyer, B. Škorić, S. Katzenbeisser, and J. Szefer, "Decay-based dram pufs in commodity devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 3, pp. 462–475, 2019.

[45] U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas, "Puf modeling attacks on simulated and silicon data," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, 2013.

**Rohini Poolat Parameswarath** received a Master of Technology degree in Software Engineering from the National University of Singapore (NUS), Singapore in 2009. She is a cyber security researcher at the Department of Electrical and Computer Engineering, NUS, Singapore. Currently, she is pursuing a PhD with research focusing on protocols for security and privacy in vehicular environments. Before joining NUS, she was part of the cyber security research team at the Singapore University of Technology and Design, Singapore. She worked as a software engineer in multinational companies before embarking on her career in cyber security research. Her papers have been published in IEEE/ACM transactions and journals, as well as in security and communication-related conferences. She is passionate about finding solutions to the current challenges in the cybersecurity landscape. Her research interests include identifying security and privacy issues in domains such as the Internet of Things (IoT), cyber-physical systems, and vehicular networks and designing secure and privacy-preserving authentication protocols to defend against such threats.

**Biplab Sikdar** (Senior Member, IEEE) is a Professor in the Department of Electrical and Computer Engineering at the National University of Singapore, where he also serves as the Head of the Department of Electrical and Computer Engineering. He 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, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was an Assistant Professor from 2001-2007 and Associate Professor from 2007-2013 in the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute. He is a recipient of the NSF CAREER award, the Tan Chin Tuan fellowship from NTU Singapore, the Japan Society for Promotion of Science fellowship, and the Leiv Eiriksson fellowship from the Research Council of Norway. His research interests include IoT and cyber-physical system security, network security, and network performance evaluation. Dr. Sikdar is a member of Eta Kappa Nu and Tau Beta Pi. He has served as an Associate Editor for the IEEE Transactions on Communications, IEEE Transactions on Mobile Computing, and IEEE Internet of Things Journal and is an IEEE COMSOC and VTS Distinguished Lecturer and ACM Distinguished Speaker.