PUF-Based Device Authentication with Zero-Knowledge Proof in IoT

Tung-Tsun Lee, Shyi-Tsong Wu, Yao-Jen Liang[*]

Department of Electronic Engineering, National Ilan University, Taiwan
ttlee@niu.edu.tw, stwu@niu.edu.tw, yaojen@niu.edu.tw

Abstract

With the development of science and technology, the Internet of Things (IoT) had been integrated into the daily life of people. That makes the security of IoT a necessity and gains more attention. The Device authentication is an important issue in the security of IoT. In this paper, we propose a device authentication scheme based on both Physically Unclonable Function (PUF) and zero-knowledge proof. The proposed mutual authentication scheme reduces the memory load on the server and provides both data integrity and confidentiality during the authenticating process. We verify the proposed device authentication algorithm on the IoT platform Raspberry Pi using SRAM-PUF. The experimental results reveal that the proposed device authentication scheme is novel for IoT. It can resist brute force attack, replay attack, man-in-the-middle attack, machine learning attacks, and etc.

Keywords: PUF, Device authentication, IoT, Zero-knowledge proof, SRAM

1 Introduction

Due to the development of science and technology, the security of Internet of Things (IoT) has gained more attention and become fundamental [1-2]. The device authentication is an important issue in the security of IoT [3-4]. However, IoT is a resource-constrained environment and the conventional cryptographic authentication schemes are impractical for IoT [5-7].

Each device in IoT needs a unique ID for identification. PUF (Physically Unclonable Function) makes the chip to have unique and inimitable characteristics due to the variation of process in the chip generation process, which is suitable for the unique ID of a device [8]. In this paper, we utilize the SRAM-PUF and hash functions as the basis, to propose a PUF-based device authentication scheme.

The proposed PUF-based device authentication scheme is a mutual authentication scheme and provides both data integrity and confidentiality in the authenticating process. We implement the device authentication method on the IoT platform: Raspberry Pi. Compared with other device authentication schemes, the proposed scheme reduces the memory load in the server [9-11]. It can resist against brute force attack, replay attack, man-in-the-middle attack, machine learning attack and other known attacks.

The main contributions of the paper are summarized as follows:

- The server of IoT for the proposed PUF-based device authentication scheme is based on zero-knowledge proof that does not require storing all Challenge-Response Pairs (CRP) for authentication. Thus, it has the advantage of low memory.
- The server of IoT needs less momory for authentication, so the scheme is scalable for large IoT system easily.
- The proposed scheme is robust. It is a mutual authentication scheme and has both confidentiality, integrity. It can resist brute attack, replay attack, and machine learning attack.

The organization of remaining sections is as follows: Section 2 provides an overview of the related works. In Section 3, we present the remanence of SRAM PUF. The proposed PUF-Based device authentication scheme is introduced in Section 4. Section 5 includes the experimental results and a security analysis for the PUF-based device authentication scheme. Lastly, Section 6 presents the conclusions of this article.

2 Related Work

In this section, we introduce Physically Unclonable Functions (PUFs) and related works of device authentication using PUFs.

2.1 Physical Unclonable Function (PUF)

A Physical Unclonable Function (PUF) is a physical random function [8]. It uses the uncertainty and randomness of process variation during chip production to make the chip unique and inimitable. The unique characteristic of chip likes fingerprints, and through Challenge-Response Pairs (CRP) for device authentication. Different groups of cross-examinations will have unique responses that match their results for certification purposes [8]. Therefore, it is difficult for a third party to simulate an exact replica of a device equipped with a PUF without knowing all the CRPs. Additionally, the user can use this method to verify whether the device comes from the original factory.

Basically, PUF is divided into memory-based PUF and delay-based PUF according to circuit topology [12].

*Corresponding Author: Yao-Jen Liang; Email: yaojen@niu.edu.tw

DOI: https://doi.org/10.70003/160792642025092605010

2.1.1 Delay-Based PUF

The most representative of delay-based PUFs are Ring Oscillator (RO) PUF and Arbiter PUF. The characteristic is that the delay will be random due to the influence of factors such as MOSFET channel length, width and threshold voltage during the production process [12].

Arbiter PUF: As show in Figure 1, an arbiter PUF is composed of many multiplexers in series [12]. The signal paths through each pair of multiplexers are switched randomly. Due to the process variations, the delay caused by different paths is not the same. Finally, the arbiter compares which path's signal arrives first and produces an output, which is the response.

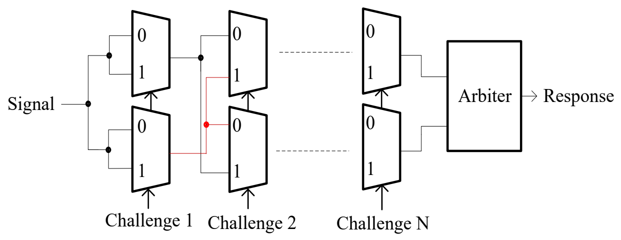


Figure 1. Arbiter PUF

Ring Oscillator (RO) PUF: As shown in Figure 2, RO PUF is composed of two $N * 1$ multiplexers, two edge detection circuits, a comparator and N oscillators with the same circuit structure [12]. Unpredictable process variation in the manufacturing process results in different frequencies resulting from different circuit delays for each oscillator. The two multiplexers select a pair of oscillator rings based on the excitation response. Then, the two edge detection circuits measure the selected oscillator ring. Finally, the comparator produces a response by comparing the two edge detection circuits.

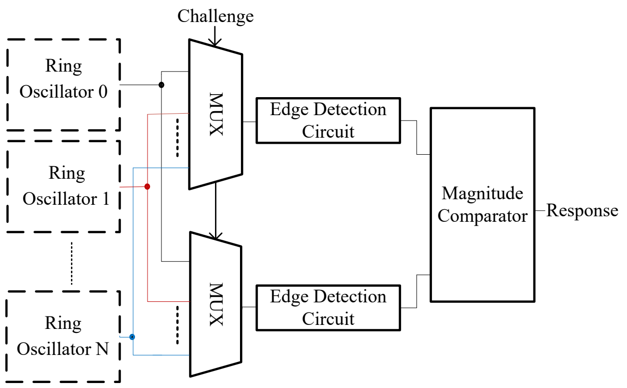


Figure 2. Ring oscillator PUF

2.1.2 Memory-Based PUF

Memory-based PUF refers to the random initial state of memory cells at the moment of device startup to extract device-related response pairs [12]. Unlike delay-based PUFs, it does not require specific additional design circuits to generate responses and can be used with standard memory cells, and thus saving circuit area.

SRAM PUF: As shown in Figure 3, SRAM PUF uses the enable value of the SRAM bit and is consists of a memory block implemented by two cross-coupled inverters [12]. As shown in the figure, its structure is symmetrical. Even if there is no external signal during startup, each bit will remain in its initial state. Once started, the voltage becomes unstable. Process variation in the manufacturing process, allowing the bit state to stabilize at 1 or 0.

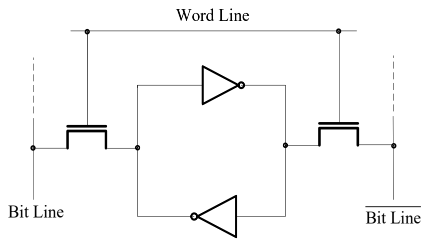


Figure 3. SRAM PUF

Butterfly PUF: As shown in Figure 4, a Butterfly PUF cell is a cross-coupled bistable circuit. The structure consists of two latches with their outputs cross-coupled [13]. When starting the PUF, set the enable signal to a high level, so that the butterfly PUF circuit is at an unstable operating point. After a few clock cycles, the enable signal is set to a low level, allowing one of the two possible stable states, 0 or 1, to appear on the output signal.

2.2 Device Authentication

There are many ways to perform user authentication, such as passwords, ID cards, passports, chips fingerprints, retina scans, and behavior, etc [13]. Device authentication is required to ensure that connected devices in the IoT can be trusted as they claim to be. Thus, each IoT device needs a unique identity that can be authenticated when the device tries to connect to a gateway or central server. With this unique ID, system administrators can track each device, communicate with it securely, and prevent it from executing unwanted processes. Assuming the device is behaving oddly, administrators can quickly cancel the device's privileges [14].

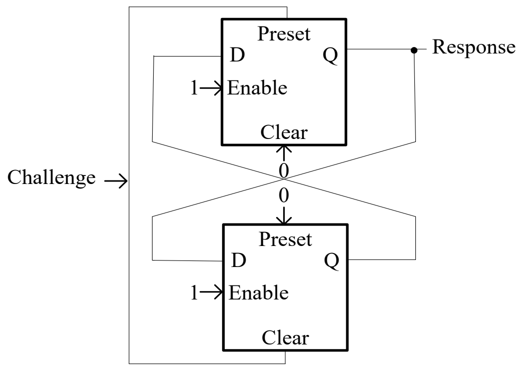


Figure 4. Butterfly PUF

As shown in Figure 5, the authentication protocol of PUF is applied to most security applications [13]. The PUF-based authentication process consists of two phases, the enrollment phase and the authentication phase.

In the enrollment/registration phase, the PUF device generates the corresponding response R_{Ci} according to the given challenge C_i. Next, the ECC (Error Correction Code) encoder generates the corresponding error correction code h_{Ci} according to the response R_{Ci}, and finally uses the hash operation to generate $K(R_{Ci})$ through the response R_{Ci}.

In the key regeneration or authentication phase, the PUF device receives the same challenge C_i and generates a corresponding response R_{Ci}'. Because PUF is affected by external noise, there will always be differences between the original R_{Ci} and the generated R_{Ci}'. However, R_{Ci}' can be restored to R_{Ci}" by using the error correction code h_{Ci}. Finally, the corrected response R_{Ci}" is used to generate the key $K(R_{Ci}")$ after the same hash operation. If the circuit remains unchanged, $K(R_{Ci}")$ will be the same as $K(R_{Ci})$.

2.3 Device Authentication Using PUF Method

The PUF-based device authentication had gained more attention for security in IoT. We introduce some methods for the device authentication based on PUFs. Ankur Jain, and et al. proposed a device authentication method in IoT using reconfigurable PUF [9]. In this scheme, PUF provides a unique hardware key depending on its specific device characteristics. This scheme requires less computing resources and is suitable for the IoT environment. However, the server needs to store all challenge-response pairs (CRP) , which causes a memory overload on the server. Additionally, the device does not authenticate the message from the server, making it vulnerable to both replay attack and the man in the middle attacks. Furthermore, the encryption of authenticating data uses an EXOR operation with a random number R_1. However, the R_1 will be broken in transmitting message $E(R_1$ exor ID), because ID had been transmitted in plaintext.

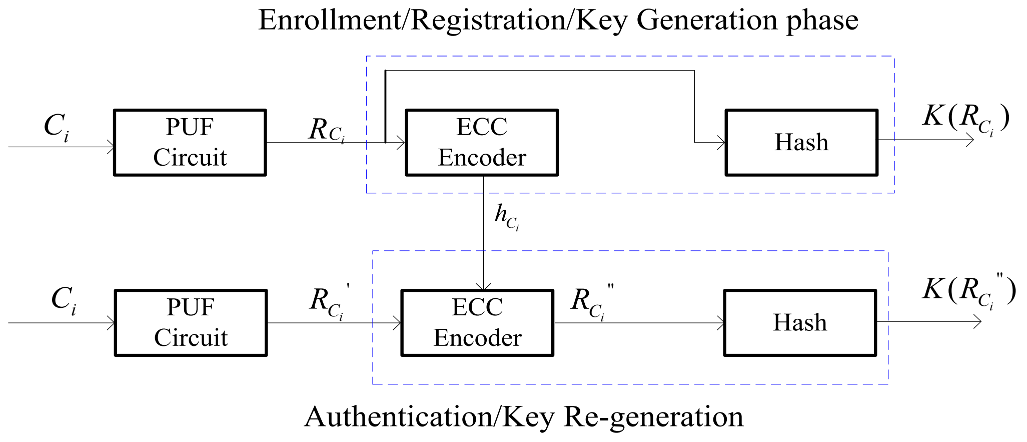


Figure 5. PUF based authentication protocol

In 2019, Byoungkoo Kim and et al. proposed a PUF-based IoT device authentication scheme [10]. In their scheme, only a single CRP needs to be stored and updated between the authentication server and the device. This minimizes the resources required for the authentication server and reduces security threats due to exposure of authentication keys. The scheme proposed by Kim et al. improves upon the disadvantage of traditional PUF authentication schemes, where the server needs to store all CRPs for authentication. Instead, this scheme stores and updates a single CRP by interaction with the device. In addition, it uses CRP to generate encryption keys that encrypt authentication messages, ensuring the confidentiality of transmission. However, the disadvantage of the scheme is that it does not use a hash function to ensure data integrity. Moreover, the server needs to store the used challenge list, which increases the memory load of the server.

Y. Yilmaz, and et al. proposed a device authentication protocol in IoT using a lightweight PUF [11]. This scheme, it has been implemented on IoT devices with limited resources. Compared with existing DTLS (Datagram Transport Layer Security), it shows better performance in terms of power consumption and resource usage. The scheme uses the neural network algorithm to build a PUF model within the device and the verifier without storing PUF challenge-response pairs (CRP) in the database. However, this PUF model is made by the neural network that requires a large amount of data and time to train. The output value of the PUF model after training is not always guaranteed to be correct. Additionally, the timestamp T_d is captured by the device if there is a time difference between the server and the devices, authentication may fail.

3 Remanence of SRAM PUF

3.1 PUF-Based Multiple Identities Set Up

For the uncontrollable nature of SRAM in the production process, it causes the SRAM cell output "1" or "0" with different probabilities, and ECC (Error-Correcting Code) is generally required for correcting errors to generate stable PUF output. However, ECC imposes a burden on computing resources on the device side [15]. Therefore, in this step, we use the algorithm of data remanence to find out the stable cells in SRAM.

First, we repeat the remanence test of SRAM 100 times, and the procedure is shown in the Figure 6. We describe the steps as follows:

1. Turn on the power. We write ‘1’ to all bit cells of the SRAM. Then turn off the power for a duration of *T*, where *T* = 300 ms in our implementation.
2. Turn on the power again to read data. Some unstable cells will flip to different states, and the bits that maintain ‘1’ at this time are called strong ‘1’. Store the address of strong ‘1’.
3. Similarly, write ‘0’ to all bit cells of SRAM, and turn off the power for a duration of *T*, where *T* = 300 ms.
4. Turn power on again, at this time the bit that maintain ‘0’ are considered strong ‘0’ and store the address of strong ‘0’.
5. Repeat these steps 100 times to analyze the PUF data and save the start address for every block.

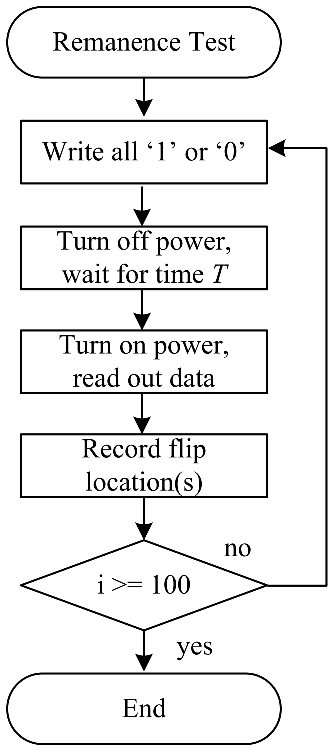


Figure 6. Flowchart of remanence test

In our implementation of device authentication, only 256-bit PUF secrets are read from SRAM. Regarding the utilization rate of SRAM, it only occupies a very small percentage of SRAM. To improve the utilization rate, we divide the SRAM into several small blocks to improve the utilization efficiency of PUF in SRAM. Each block of SRAM can provide multiple PUF secrets for different entities.

As shown in Figure 7, the storage space of SRAM is divided into several blocks, and each block can extract a 256-bit PUF secret. In the key generation phase, the function GetPUF( ) executes the procedure of PUF reading from the SRAM and retrieves the secret values of PUF from the stored start address for each block.

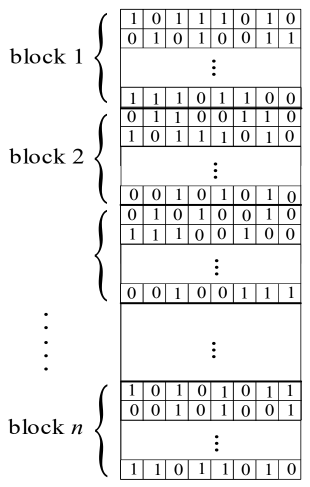


Figure 7. Division of the SRAM

3.2 SRAM PUF Experimental Evaluation

The IoT platform used in this paper is Raspberry Pi, and the system block diagram is shown in Figure 8. The SRAM PUF in our implementation is the 23LC1024 chip, connected via the synchronous serial communication protocol SPI (Serial Peripheral Interface), and linked to the server through Wi-Fi using a socket. The component side circuit diagram and the physical diagram of the Raspberry Pi are shown in Figure 9 and Figure 10.

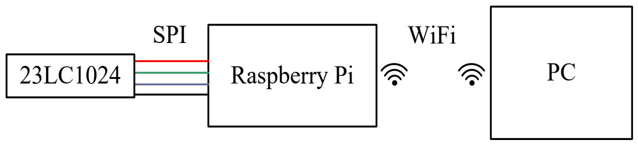


Figure 8. System block diagram

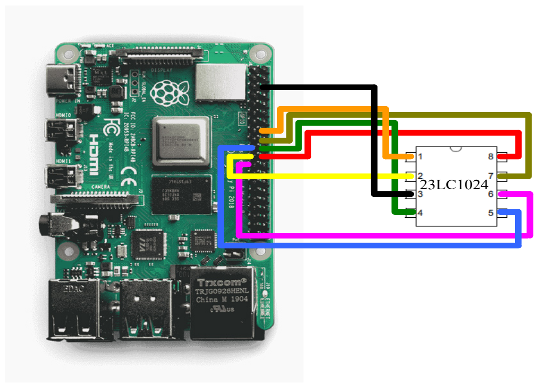


Figure 9. PUF experimental circuit diagram

Due to the inherent variability throughout of the entire lifecycle for SRAM, the state of each individual cell cannot be guaranteed. To address the volatility of SRAM and ensure stable responses, in this experiment, we employ the data remanence algorithm instead of ECC to identify stable cells in this experiment.

- **Remanence of SRAM PUF Evaluation**

Our experimental setup use a Raspberry Pi as the SRAM control and reading terminal device, implemented with the reading program written in C programming language. To verify whether the power-off duration affects the number of stable SRAM cells, we conducted separate power-up tests to analyze the number of stable '1' (1-skewed cells) and stable '0' (0-skewed cells) states in SRAM. In this experiment, we tested different power-off durations to evaluate the remanence of SRAM PUF. Multiple power-off durations were tested, as show in Table 1. In this table, the percentage of the 0-skewed cells(C_0) and 1-skewed cells(C_1) are calculated from Equation (1), respectably. Where N_0 represents the number of 0-skewed cells, and N_1 represents the number of 1-skewed cells:

$$C_0 = \frac{N_0}{N_0 + N_1}, \quad C_1 = \frac{N_1}{N_0 + N_1}. \tag{1}$$

Figure 10 shows the percentage of the cells that remain stable for 1-skewed cells and 0-skewed states. Besides, the shorter the power-off duration, the more stable cells obtained through the data remanence algorithm [16]. To ensure that consistently obtain a PUF value, we use a power-off duration of 300 ms. In our device authentication scheme, we obtain strong '1' and '0' states of the SRAM through the data remanence algorithm. These strong and stable cells serve as the foundation for the key of device authentication.

Table 1. Remanence tests of SRAM-PUF

(a) Number of 1-skewed cells

Power-off time (ms)	Number of cells	C_1
300	17505	81.93%
330	43264	73.16%
480	48128	72.03%
630	47360	74.00%
780	48640	73.64%
930	48384	73.83%

(b) Number of 0-skewed cells

Power-off time (ms)	Number of cells	C_0
300	3802	18.07%
330	15872	26.84%
480	18688	27.97%
630	16640	26.00%
780	17408	26.36%
930	17152	26.17%

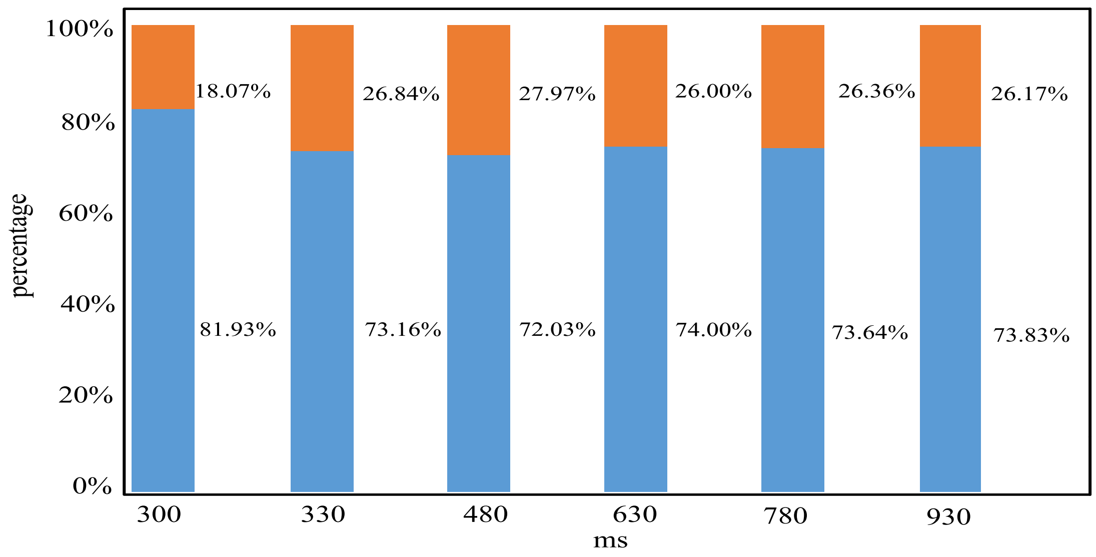


Figure 10. Distribution of "0" and "1" cells in the different power-off times of SRAM cells

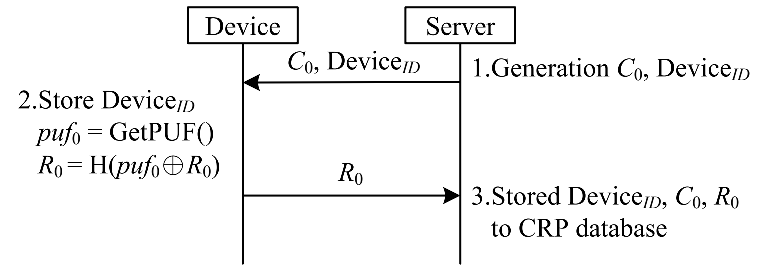
- **PUF Quality**

The uniqueness of a PUF is the primary indicator of its quality, as it signifies whether the responses generated by different PUFs using the same challenge are distinct or not. The Hamming distance (Inter-HD) is commonly used to indicate the level of uniqueness. It measures the number of positions where corresponding symbols differ between two PUF values of equal length [17]. The average Inter-HD is defined as follows:

$$\text{Inter-HD} = \frac{2}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \frac{HD(R_i, R_j)}{n} \times 100\%. \tag{2}$$

Where R_i and R_j (where $i \neq j$) represent the PUF chips i and j, respectively, and correspond to an n-bit response for a given challenge C. K denotes the number of chips. In an ideal scenario, the Inter-HD should be 50%, which means that if different chips are given the same challenge, their responses should differ by 50% [17].

In our proposed scheme, we divide 1M-bit PUF into 10 blocks, where $K = 10$, and calculate its Inter-HD. We read a 256-bit PUF values for each block, and $n = 256$ in Equation (2). Finally, the Inter-HD for our proposed scheme is 49.53%. This confirms the sufficient uniqueness of the PUF keys in our proposed scheme.


Device
Server
C_0, Device$_{ID}$
1.Generation C_0, Device$_{ID}$
2.Store Device$_{ID}$
puf_0 = GetPUF()
$R_0 = H(puf_0 \oplus R_0)$
R_0
3.Stored Device$_{ID}$, C_0, R_0 to CRP database

C_0 : Initial challenge
R_0 : Initial response
puf_0 : Original PUF's secret
Device$_{ID}$: Give the device a random number for identification purposes
CRP database : Database of Challenge-Response Pair

Figure 11. Enrollment phase

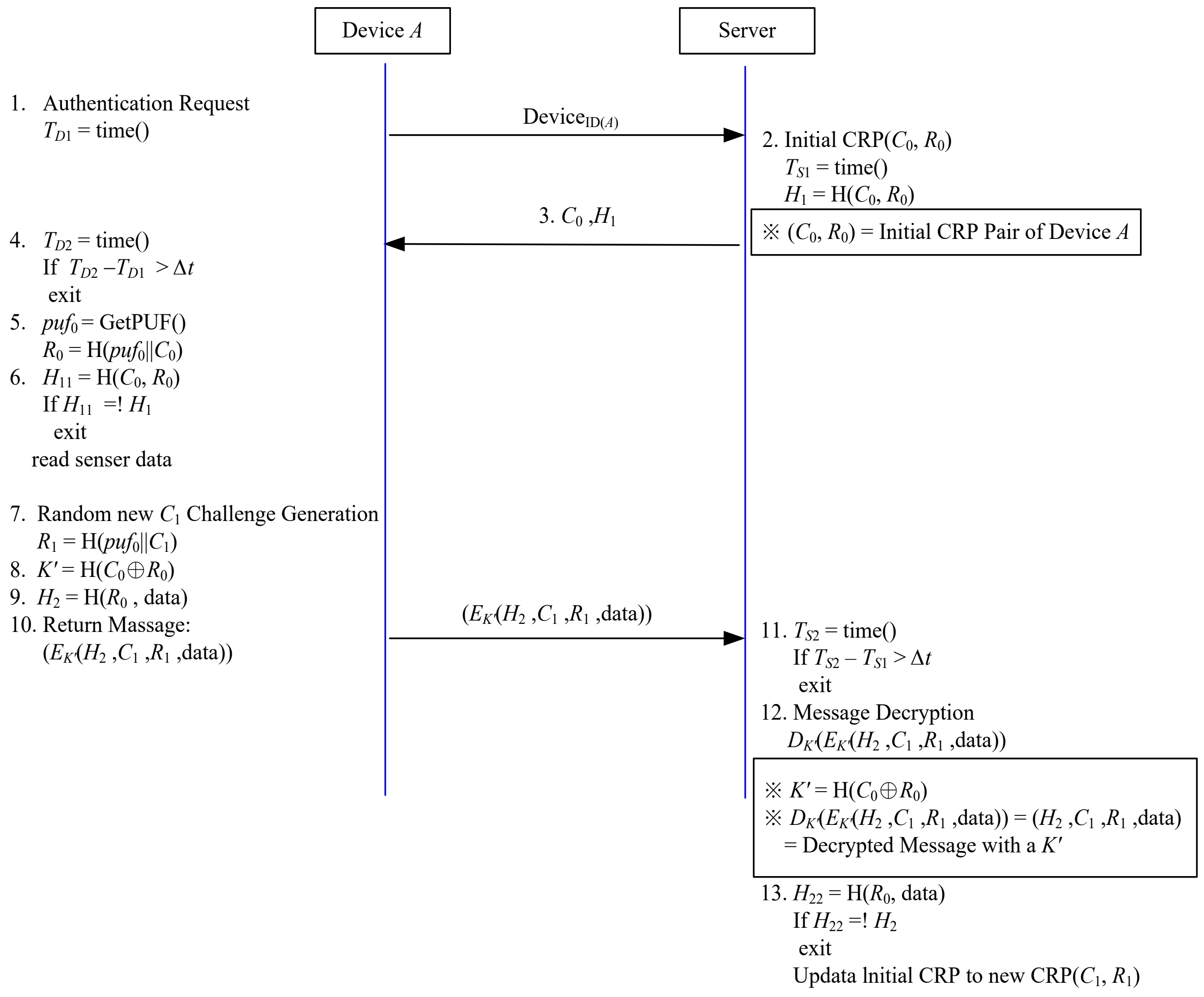

Device A
Server
1. Authentication Request
T_{D1} = time()
Device$_{ID(A)}$
2. Initial CRP(C_0, R_0)
T_{S1} = time()
$H_1 = H(C_0, R_0)$
3. C_0 ,H_1
※ (C_0, R_0) = Initial CRP Pair of Device A
4. T_{D2} = time()
If $T_{D2} - T_{D1} > \Delta t$
exit
5. puf_0 = GetPUF()
$R_0 = H(puf_0 || C_0)$
6. $H_{11} = H(C_0, R_0)$
If H_{11} =! H_1
exit
read senser data
7. Random new C_1 Challenge Generation
$R_1 = H(puf_0 || C_1)$
8. $K' = H(C_0 \oplus R_0)$
9. $H_2 = H(R_0$, data)
10. Return Massage:
($E_{K'}$(H_2 ,C_1 ,R_1 ,data))
($E_{K'}$(H_2 ,C_1 ,R_1 ,data))
11. T_{S2} = time()
If $T_{S2} - T_{S1} > \Delta t$
exit
12. Message Decryption
$D_{K'}$($E_{K'}$(H_2 ,C_1 ,R_1 ,data))
※ $K' = H(C_0 \oplus R_0)$
※ $D_{K'}$($E_{K'}$(H_2 ,C_1 ,R_1 ,data)) = (H_2 ,C_1 ,R_1 ,data)
= Decrypted Message with a K'
13. $H_{22} = H(R_0$, data)
If H_{22} =! H_2
exit
Updata Initial CRP to new CRP(C_1, R_1)

Figure 12. Proposed device authentication scheme based on PUF

4 The Proposed PUF-Based Device Authentication Scheme

Our scheme is designed to authenticate devices. It consists of two phases: the enrollment phase and the authentication phase, as shown in this section.

- **Enrollment Phase**

In this phase, the server stores a CRP required to authenticate the device. The steps are as follows and shown in Figure 11.

1. The server generates a challenge (C_0) and a Device$_{ID}$, and sends them through a secure channel to the device. In the enrollment phase, the information secure transmission channel is necessary.
2. Upon Receiving the challenge C_0 and Device$_{ID}$, the device stores the Device$_{ID}$ for future identification. Then, the device generates the original PUF secret puf_0 by calling GetPUF( ) and produces a response R_0 by the equation:

$$R_0 = \mathrm{H}(puf_0 \oplus C_0), \tag{3}$$

where H( ) is a hash function, in our implementation, H( ) is SHA-256.

3. After that, the device sends R_0 through a secure channel to the server. Finally, the server stores the initial tuple (C_0, R_0) to the CRP database.

In the enrollment phase, the tuple (C_0, R_0) is stored in the server's database. The length of both the challenge and response is 256-bit.

- **Authentication Phase**

The proposed device authentication method is shown in Figure 12 and explained as follows.

1. First, device A sends its ID (Device$_{\mathrm{ID}(A)}$) so that the server can identify itself. Then device A stores the timestamp T_{D1}.
2. The server obtains C_0 from its initial CRP according to the ID (Device$_{\mathrm{ID}(A)}$). Then, the server stores the timestamp T_{S1}. According to the initial CRP (C_0, R_0) of device A, H_1 is obtained by the hash function:

$$H_1 = \mathrm{H}(C_0, R_0). \tag{4}$$

3. The Server sends the data (C_0, H_1) to device A.
4. After receiving the data, device A compares the timestamp T_{D2} of the received (C_0, H_1), with the timestamp T_{D1} of the sent data, and confirms whether the time difference is within an allowable Δt delay. Otherwise, the communication is terminated.
5. Upon receiving the data (C_0, H_1), device A retrieves the original PUF's secret puf_0 from the enrollment phase and use C_0 to get the response R_0 using Equation (3).
6. Device A uses C_0, R_0 and a hash function to calculate H_{11}:

$$H_{11} = \mathrm{H}(C_0, R_0). \tag{5}$$

If H_{11} is equal to H_1, it means that the server (C_0, R_0) is correct, and the authentication is passed. Then the device A reads the data of sensor. Otherwise, the communication is terminated.

7. For subsequent device authentication, device A randomly generates a challenge C_1 and computes the corresponding response R_1 through the PUF:

$$R_1 = \mathrm{H}(puf_0 \oplus C_1). \tag{6}$$

8. Device A XORs the previous CRP(C_0, R_0) and applies a hash function to generate the key K' for encryption in transmission.

$$K' = \mathrm{H}(C_0, R_0). \tag{7}$$

9. Device A applies the hash function with R_0 and data to compute H2 for ensuring data integrity:

$$H_2 = \mathrm{H}(R_0, \text{data}). \tag{8}$$

10. Device A returns the encrypted data $E_{K'}(H_2, R_1, C_1, \text{data})$ to the server.
11. After receiving the data, the server compares the timestamp T_{S2} of the received data with the timestamp T_{S1} of the sent data to verify whether the time difference is within a reasonable Δt delay. Otherwise, the communication is terminated.
12. The server uses the key K' from Equation (7) to decrypt the data: $D_{K'}(E_{K'}(H_2, R_1, C_1, \text{data}))$.
13. The server uses the data and R_0 as inputs for the hash function to compute H_{22}:

$$H_{22} = \mathrm{H}(R_0, \text{data}). \tag{9}$$

H_{22} is compared with the received H_2. If the two are identical, this confirms that the information is correct and the integrity verification is successful. The sever then updates the initial CRP(C_0, R_0) to the new CRP(C_1, R_1).

In this paper, we implement the proposed authentication scheme on a Raspberry Pi, which serves as the IoT platform. We also describe the security analysis and efficiency analysis. The experimental setup consists of a Raspberry Pi 3 model B. The Raspberry Pi CPU is a 1.2 GHz 64-bit quad-core ARM. It has 1GB LPDDR2 RAM, a 16GB SD card, and runs the Raspbian Linux 10 (buster) operating system, version 10.11. The SRAM PUF in own implementation is 23LC1024 chip, with a capacity of 1M-bit.

5 Experimental Results and Security Analysis

In this section, we present the experimental results and security analysis. The experimental environment of this paper is the Raspberry Pi 3 Model B with 2GB of RAM. The experimental process and results are described in subsection 5.1. Security analysis is given in subsection 5.2, and subsection 5.3 provides the comparison with other schemes.

5.1 Experimental Process and Results

The algorithm of the proposed device authentication is implemented in the Raspberry Pi. This subsection deals with the experimental process and the results.

- **Operational process**

The operational process of the proposed device authentication scheme on the server-side is shown in Figure 13. The explaination is as follows:

Line 1-2: The server waits for the authentication request from the device side and displays the identification ID of the Device$_{ID}$.

Line 3-4: The server confirms that the identification ID, if Device$_{ID}$ exists and records the start timestamp T_{S1}.

Line 5-7: According to the specified identification ID, the server reads its initial CRP from the database, and obtains 256-bit H_1 through Equation (4). Send C_0, 256-bit, and H_1 back to the device side.

Line 8-9: After receiving the message from the device side, the server verifies the timestamp T_{S2}. Whether it is within a reasonable time delay.

Line 10: The server uses the Equation (7) to get the 128-bit key *K'*.

Line 11-12: Shows the received device-side authentication request is shown as $E_{K'}(H_2, R_1, C_1, \text{data})$.

Line 13-14: The plaintext $D_{K'}(E_{K'}(H_2, R_1, C_1, \text{data}))$ is obtained after decryption using the key *K'*.

Line 15-18: Using R_0 and data, the server calculates 256-bit H_{22} through Equation (9). If H_{22} matches H_2, the integrity verification is completed.

Line 19-21: C_1 and R_1 are saved for use in the next authentication.

The operational process of the device authentication scheme on the device-side is shown in the Figure 14. We explain as follows:

Line 1-3: The device sends its identification ID, Device$_{ID}$, to the server and gets start the timestamp T_{D1}.

Line 4-5: After receiving the data from the server at T_{D2}, the device side first checks whether the timestamp T_{D2} is valid. Otherwise, the communication is terminated.

Line 6-7: Upon receiving the data (C_0, H_1), the device uses GetPUF( ) to obtain the secret 256-bit puf_0, and the 256-bit response R_0 according to Equation (3).

Line 8-9: The device side computes 256-bit H_{11} through Equation (5) and compares it with the received H_1. If H_{11} matches H_1, the authentication is passed. Otherwise, the communication is terminated.

Line 10-12: A new 256-bit C_1 is randomly generated on the device side, and the corresponding 256-bit R_1 is computed as equation (4-18). Then, the device reads the data of sensor, and pads it with random number to reach 128-bit for encryption.

Line 13: The 256-bit H_2 is computed using Equation (8).

Line 14: The initial CRP values, C_0 and R_0, are used to derive the 128-bit key *K'* by Equation (7).

Line 15-16: The ciphertext $E_{K'}(H_2, R_1, C_1, \text{data})$ is sent to the server.

5.2 Security Analysis

The security of the proposed PUF based authentication scheme is dependent on the PUF secret puf_0 and CRP. The attacker's goal is to find out the puf_0 or the key. In this subsection, we analyze the security of our proposed scheme against known attacks.

To be implemented in IoT devices with resource constrains, the key in our scheme is 128-bit based on Equation (7). Its key space is 2^{128} and the security strength is sufficient in IoT against brute force attacks. Therefore, the proposed scheme is resistant to brute force attacks [18].

- **Man-in-the-middle attack**

In this attack, the purpose is to intercept and modify the the communication between the server and the device [19]. The attacker intercepts the authentication request from the device and sends a spoofed authentication request to the server. This means that the attacker acts as a middleman between the server and the device. If the system lacks security measures, it may not be aware of this attack, because the communication is not directly interrupted.

In our proposed scheme, an attacker would need to generate the correct H_2 to pass the authentication and send the spoofed request to the server. However, the H_2 is generated by Equation (8), which requires R_0 to be computed using Equation (3). If the attacker lacks the same SRAM, the computed puf_0 will be incorrect. As a result, the attacker cannot generate a valid R_0. Any malicious H_2 sent to the server will fail authentication.

- **Replay attack**

A replay attack occurs when an attacker captures transmitted authentication messages and retransmits them to gain unauthorized access [20]. The purpose of this attack is to intercept and resend previously sent messages to trick the system into granting access.

For our authentication scheme, timestamps are implemented in the authentication procedure, including T_{D1}, T_{D2}, T_{S1} and T_{S2}. This prevents hackers from resending previously sent messages and mitigates the risk of eavesdropping, message theft, or replaying malicious requests.

Our encryption key is dynamically generated based on Equation (7) using the CRP. Each CRP tuple is used only once and then discarded, ensuring that the key is unique for each authentication process. This means that even if an attacker intercepts a message and attempts to replay it, authentication will fail because the message has expired.

- **Machine learning attack**

PUF is a physically unique secret that can generate a large number of CRP pairs for authentication. However, an attacker could attempt to collect a subset of CRPs to construct a numerical model to predict PUF responses. A successful machine learning attack could break the security of PUF and render PUF-based security protocols ineffective [21].

For the proposed authentication scheme, we propose a lightweight hybrid chaotic encryption method to protect responses, preventing attacker from collecting CRP tuples. This makes it infeasible for machine learning algorithms to execute a machine learning attack against our PUF based authentication scheme. Moreover, our response R_1 is generated using Equation (6), which protects the PUF secret puf_0 by incorporating SHA-256 hashing.

- **Scalability**

The server of the proposed authentication scheme does not require storing all CRP for authentication. Thus, it has the advantage of low memory load of server. With this advantage, the proposed scheme in this study is one device for example, but it can be scalable for large-scale IoT. To allow a vast number of devices for authentication simultaneously, the IoT's platform needs more comutation power and more memory. The proposed PUF-based device authentication scheme demands less memory, and more memory can be released for authentication operation. thus, it has the advantage of scalability for large-scale IoT.

```
1     Start the socket service and wait for the client to connect...
2     Client ID : 45
3     Thu Mar  2 22:00:28 2023
4     start Timestamp
5     C0 = 0e6372720ecd4e029dc568e09564b1f886aec8ebaf8f491c8416bf7a19006020
6     H1 = 6a53afac12723e272f2e819c1b76b52fc1e3ed191ffa3e79ae540f2582bf8750
7     send C0 & H1
8     Thu Mar  2 22:00:28 2023
9     Timestamp check passed
10    key : c99ddeca8649d9228550bfcd40fa20ae
11    Ek(H2, R1, C1, data):
12    972d2709d7c2d89ba1fc7495d6748386af6978e11d3337bd114ba3394b7ab460
      dd0290f494a84c6ea9279a59924e704dea435053881e386c7b7652d7f8ca5d50
      321818218ca1589dce079603ca06b0cf4c3abdb84aa9ed5d9532385661437c44
      819e420f88eb2715bb8fc109dfe6c1a8
13    client_data decode:
14    fd62d75d0f770b3f546eef94d6134ef83cc48753627d3b31f71807efd8d19214
      c8b9d45dd613a4fe24bf6a711f4481547530c663eb7ddd69b2774f2def308c92
      d76f0365e5a94401a2363b671a64795f746e29701e854637bc506ca8848074ec
      7f09216650ee45c990a86d237b052c50
15    H2 = fd62d75d0f770b3f546eef94d6134ef83cc48753627d3b31f71807efd8d19214
16    data = 7f09216650ee45c990a86d237b052c50
17    H22 = fd62d75d0f770b3f546eef94d6134ef83cc48753627d3b31f71807efd8d19214
18    Integrity verification passed
19    Complete certification and update CRP
20    New C1 : d76f0365e5a94401a2363b671a64795f746e29701e854637bc506ca8848074ec
21    New R1 : c8b9d45dd613a4fe24bf6a711f4481547530c663eb7ddd69b2774f2def308c92
```

Figure 13. Device authentication data on the server-side

```
1     ID : 45
2     Thu Mar  2 22:00:28 2023
3     start Timestamp
4     Thu Mar  2 22:00:28 2023
5     Timestamp check pass
6     R0 =ae5a3b3196bd7f3605e6d4c2a92da81f8a36b7ebd662505e98ff56d570d85301
7     H1 =6a53afac12723e272f2e819c1b76b52fc1e3ed191ffa3e79ae540f2582bf8750
8     H11=6a53afac12723e272f2e819c1b76b52fc1e3ed191ffa3e79ae540f2582bf8750
9     Integrity verification passed!
10    C1 : d76f0365e5a94401a2363b671a64795f746e29701e854637bc506ca8848074ec
11    R1 : c8b9d45dd613a4fe24bf6a711f4481547530c663eb7ddd69b2774f2def308c92
12    data : 7f09216650ee45c990a86d237b052c50
13    H2 : fd62d75d0f770b3f546eef94d6134ef83cc48753627d3b31f71807efd8d19214
14    key : c99ddeca8649d9228550bfcd40fa20ae
15    result encrypted message :
16    972d2709d7c2d89ba1fc7495d6748386af6978e11d3337bd114ba3394b7ab460
      dd0290f494a84c6ea9279a59924e704dea435053881e386c7b7652d7f8ca5d50
      321818218ca1589dce079603ca06b0cf4c3abdb84aa9ed5d9532385661437c44
      819e420f88eb2715bb8fc109dfe6c1a8
```

Figure 14. Device authentication data on the device-side

5.3 Comparison

We analyze and compare the efficiency of the proposed scheme with other related works, shown in Table 2. The proposed authentication scheme provides confidentiality, integrity and mutual authentication. Our key space is 2^{128} and the security strength is ensured by our proposed lightweight hybrid chaotic encryption. Therefore, the proposed scheme is resistant to brute-force attacks. Besides, a hash function is utilized in our scheme to prevent unauthorized data tampering, which can resist man-in-the-middle attacks to tamper with the data. In order to prevent replay attack, we add a timestamp to ensure that the time duration of the transmission is within a reasonable range. In addition, encryption is used to protect our response R_1, and the attacker cannot collect our CRP tuple. This prevents an attacker from using the machine learning algorithms to execute a machine learning attack on our PUF-based authentication scheme. Moreover, our scheme is equipped with mutual authentication. The device-side can authenticate whether the server-side is legitimate by checking H_1 and H_{11}. The server-side can also authenticate device-side by verifying the integrity of the massage H_2 and H_{22}.

Next, we analyze and compare the efficiency of our proposed scheme with other relevant works, as shown in Table 3, where CO_D is the computation overhead for the device, CO_S is the computation overhead for the server, XOR is a logical EXOR operation, and AES and RC5 are block cipher operations. Hashing is a cryptographic algorithm that converts input data into a fixed-size alphanumeric string.

In the [9] and [10] schemes, the error correction code h_{c_i} needs to be stored on the device-side, and all CRP tuples need to be stored on the server-side, so the storage space required by the server-side and the device-side is large. However, scheme [9] uses only an 8-bit challenge and response, which reduces the storage space required on the server and device to a medium level. However, it results in lower security strength. On the other hand, the [11] scheme needs to store neural networks and the PUF model on both the device and the server, so it requires a large amount of space to store data. The [22] scheme uses a fingerprint authentication for each device and requires a list of stable cells and error correction codes h_{c_i} to be stored on the device-side, so it requires a large amount of storage space. On the server-side, it needs store a set of fingerprints for each device-side, which reduces the server's storage requirements.

Our proposed scheme, similar to the [23] scheme, uses a data remanence algorithm to obtain the PUF secret value, and the device-side needs to store a stable cell list. We only need to store one CRP tuple for each device on the server-side, which reduces the server's memory load.

Table 2. Comparison of security analysis

Schemes	Jain's [9]	Kim's [10]	Yilmaz's [11]	Rivera's [23]	Farha's [22]	Our scheme
Confidentiality	–	✓	✓	–	✓	✓
Integrity	–	–	–	✓	✓	✓
Mutual Authentication	–	–	✓	–	–	✓
Brute force attack	–	✓	✓	–	✓	✓
Man-in-the-middle attack	–	✓	–	✓	✓	✓
Replay attack	–	–	✓	✓	–	✓
Machine learning attack	–	–	–	✓	✓	✓

Table 3. Comparison of the PUF-based authentication schemes

Schemes	Jain's [9]	Kim's [10]	Yilmaz's [11]	Rivera's [23]	Farha's [22]	Our scheme
CO_D	2 XOR	1 AES	1 XOR + 1 RC5 + PUF-Model	1 XOR + 3 Hash	4 XOR + 1 Hash	1 XOR + 5 Hash
CO_S	2 XOR	1 AES	1 XOR + 1 RC5 + PUF-Model	1 XOR + 2 Hash	4 XOR + 1 Hash	1 XOR + 3 Hash
Device-side storage space	Medium	Big	Big	Big	Big	Big
Server-side storage space	Medium	Big	Big	Small	Small	Small

CO_D: Computation overhead for device
CO_S: Computation overhead for server

Finally, we compare our proposed scheme with other schemes in terms of required authentication time and communication cost per authentication, as shown in Table 4, where communication cost is the amount of data required to complete a authentication on the server-side and the device-side, and the computation cost is the time requiredconsumed it takes to complete a authentication on both the server-side and the device-side. In this experiment, we use the network protocol Wi-Fi 802.11ax.

Figure 15 depicts the comparison of communication costs in bits. Compared with [11, 23], and [22], we require a higher communication cost. It is because that both the challenge and response are 256-bit in our scheme, which provides higher security. After each successful authentication, the legitimate CRP tuple stored on the server-side must be updated. The purpose of this is to prevent the theft of important keys and ensure the security of the authentication system in case of an attack on the server. Besides, our proposed scheme provides valid mutual authentication, which enchances security strength.

Besides, we calculate the computation cost of the authentication phase and plot the comparison of computation costs in milliseconds, as shown in Figure 16. Our scheme uses two XOR operations and a hash function eight times. We use the data remanence algorithm to obtain the PUF secret values, which requires less computation cost than the scheme [10] and [22], which require error correction code h_{c_i} operations. Compared with the scheme in [23] that use the remanence algorithm but lacks mutual authentication, our scheme requires less computation cost while providing mutual authentication and improving security strength. Besides, our proposed scheme encrypts the data through a stream cipher to ensure its confidentiality. Compared with the scheme in [10] and [11] that use block ciphers, our scheme has lower computation costs and better efficiency.

Table 4. Experiment results of authentication

Schemes	Jain’s [9]	Kim’s [10]	Yilmaz’s [11]	Rivera’s [23]	Farha’s [22]	Our scheme
Communication cost	32-bit	2320-bit	784-bit	1024-bit	436-bit	1408-bit
Computation cost	100 ms	502 ms	171 ms	140 ms	168 ms	159 ms

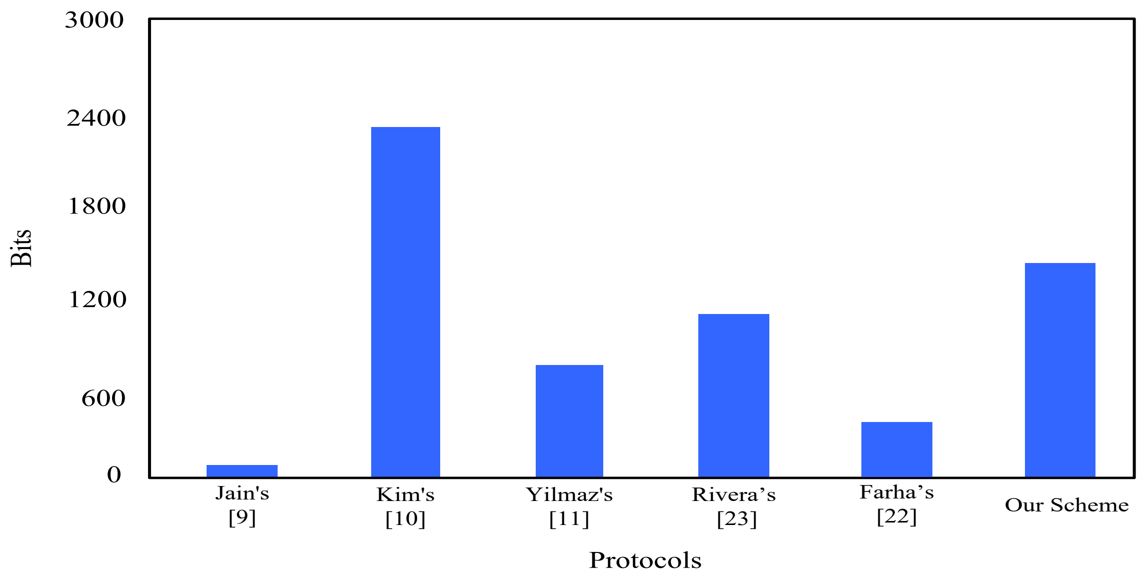


Figure 15. Communication cost

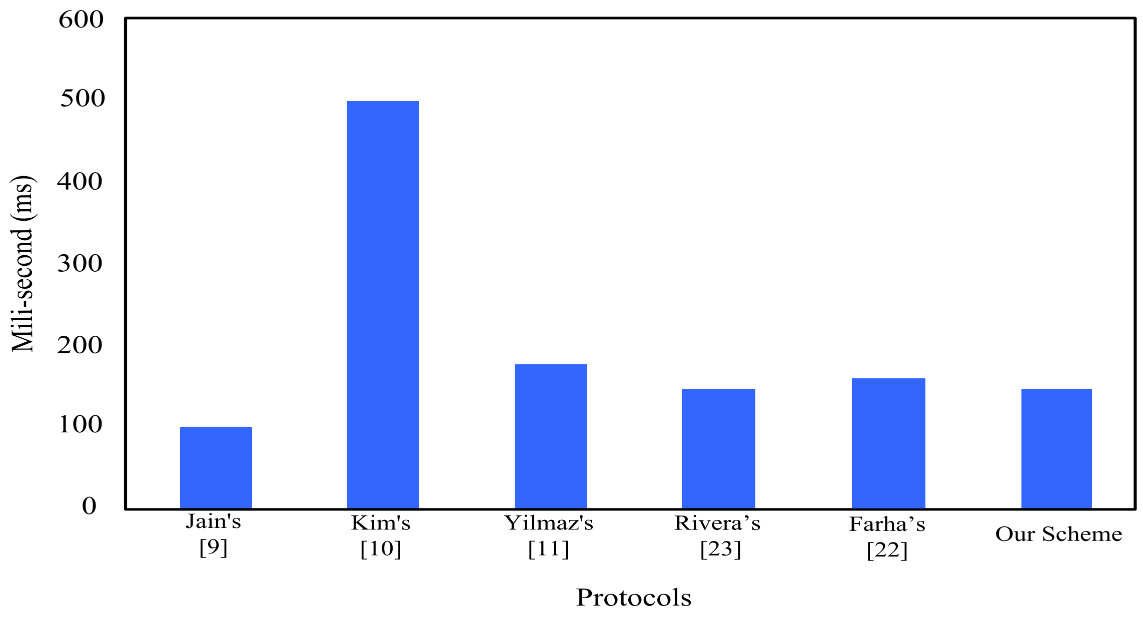


Figure 16. Computation cost

6 Conclusion

In this paper, we propose a PUF-based component authentication scheme for the Internet of Things. This method reduces the burden on server memory and protects the security of the transmitted data through encryption. For each authentication process, only a single CRP needs to be stored and updated. Timestamps are added to the transmitted data to ensure that the received data is within a reasonable time frame, preventing replay attacks. A hash function is added during transmission to ensure data integrity. The encryption and decryption keys of both parties change each time, ensuring that the communication content is secure and confidential. The PUF-based authentication with zero-knowledge proof is set to revolutionize IoT security by providing a highly secure, lightweight, and privacy-preserving authentication mechanism. Future work would focus on improving efficiency, standardization, and integration with emerging technologies, such as Artificial Intelligence (AI), blockchain, and etc.

Acknowledgments

The authors wish to thank Mr. Geoffrey Ting Sie Huo for his important assistance and the anonymous referees for their valuable and useful comments that have greatly enriched this paper.

References

[1] Y. Li, L. Qin, Q. Liang, Research on Wireless Sensor Network Security, *2010 International Conference on Computational Intelligence and Security*, Nanning, China, 2010, pp. 493-496. https://doi.org/10.1109/CIS.2010.113

[2] A. Banafa, Three major challenges facing iot, *IEEE Internet of things*, March, 2017.

[3] G. Rajendran, R. S. R. Nivash, P. P. Parthy, S. Balamurugan, Modern security threats in the Internet of Things (IoT): Attacks and Countermeasures, *2019 International Carnahan Conference on Security Technology (ICCST)*, Chennai, India, 2019, pp. 1-6. https://doi.org/10.1109/CCST.2019.8888399

[4] N. M. Karie, N. M. Sahri, W. Yang, C. Valli, V. R. Kebande, A Review of Security Standards and Frameworks for IoT-Based Smart Environments, *IEEE Access*, Vol. 9, pp. 121975-121995, September, 2021, https://doi.org/10.1109/ACCESS.2021.3109886

[5] N. A. Gunathilake, W. J. Buchanan, R. Asif, Next Generation Lightweight Cryptography for Smart IoT Devices: Implementation, Challenges and Applications, *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, 2019, pp. 707-710. https://doi.org/10.1109/WF-IoT.2019.8767250

[6] P. A. Patil, R. Boda, Analysis of Cryptography: Classical Verses Quantum Cryptography, *International Research Journal of Engineering and Technology*, Vol. 3, No. 5, pp. 1372-1376, May, 2016.

[7] T. Okamura, Lightweight cryptography applicable to various IoT devices, *NEC Technical Journal*, Vol. 12, No. 1, pp. 67-71, October, 2017.

[8] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, Controlled physical random functions, *18th Annual Computer Security* Applications *Conference, 2002. Proceedings*, Las Vegas, NV, USA, 2002, pp. 149-160. https://doi.org/10.1109/CSAC.2002.1176287

[9] A. Jain, A. M. Joshi, Device Authentication in IoT using Reconfigurable PUF, *2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, Manama, Bahrain, 2019, pp. 1-4. https://doi.org/10.1109/MENACOMM46666.2019.8988545

[10] B. Kim, S. Yoon, Y. Kang, D. Choi, PUF based IoT Device Authentication Scheme, *2019 International Conference on Information and* Communication *Technology Convergence (ICTC)*, Jeju, Korea, 2019, pp. 1460-1462. https://doi.org/10.1109/ICTC46691.2019.8939751

[11] Y. Yilmaz, S. R. Gunn, B. Halak, Lightweight PUF-Based Authentication Protocol for IoT Devices," *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, Costa Brava, Spain, 2018, pp. 38-43. https://doi.org/10.1109/IVSW.2018.8494884

[12] M. Al-Haidary, Q. Nasir, Physically Unclonable Functions (PUFs): A Systematic Literature Review, *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, Dubai, United Arab Emirates, 2019, pp. 1-6. https://doi.org/10.1109/ICASET.2019.8714431

[13] Z. A. Alizai, N. F. Tareen, I. Jadoon, Improved IoT Device Authentication Scheme Using Device Capability and Digital Signatures, *2018 International Conference on Applied and Engineering Mathematics (ICAEM)*, Taxila, Pakistan, 2018, pp. 1-5. https://doi.org/10.1109/ICAEM.2018.8536261

[14] B. M. S. B. Talukder, F. Ferdaus, M. T. Rahman, Memory-Based PUFs are Vulnerable as Well: A Non-Invasive Attack Against SRAM PUFs, *IEEE Transactions on Information Forensics and Security*, Vol. 16, pp. 4035-4049, 2021. https://doi.org/10.1109/TIFS.2021.3101045

[15] S. Shankar, A. S. Tomar, G. K. Tak, Secure Medical Data Transmission by using ECC with Mutual Authentication in WSNs, *Procedia Computer Science*, Vol. 70, pp. 455-461, 2015. https://doi.org/10.1016/j.procs.2015.10.078

[16] M. Liu, C. Zhou, Q. Tang, K. K. Parhi, C. H. Kim, A Data Remanence Based Approach to Generate 100% Stable Keys from an SRAM Physical Unclonable Function, *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Taipei, Taiwan, 2017, pp. 1-6. https://doi.org/10.1109/ISLPED.2017.8009192

[17] Z.-W. Lai, P.-H. Huang, K.-J. Lee, Using both Stable and Unstable SRAM Bits for the Physical Unclonable Function, *Journal of Electronic Testing*, Vol. 38, No. 5, pp. 511–525, October, 2022. https://doi.org/10.1007/s10836-022-06025-8

[18] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen, M. Médard, Centralized vs Decentralized Targeted Brute-Force Attacks: Guessing With Side-Information, *IEEE Transactions on Information Forensics and Security*, Vol. 15, pp. 3749-3759, 2020. https://doi.org/10.1109/TIFS.2020.2998949

[19] M. Data, The Defense Against ARP Spoofing Attack Using Semi-Static ARP Cache Table, *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, Indonesia, 2018, pp. 206-210. https://doi.org/10.1109/SIET.2018.8693155

[20] H. D. Syahla, D. Ogi, Implementation of Secure Parking Based on Cyber-Physical System using One-time Password Gong et al. Scheme to Overcome Replay Attack, *2021 International Conference on ICT for Smart Society (ICISS)*, Bandung, Indonesia, 2021, pp. 1-6. https://doi.org/10.1109/ICISS53185.2021.9533246

[21] S.-J. Wang, Y.-S. Chen, K. S.-M. Li, Modeling Attack Resistant PUFs Based on Adversarial Attack Against Machine Learning, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 11, No. 2, pp. 306-318, June, 2021. https://doi.org/10.1109/JETCAS.2021.3062413

[22] F. Farha, H. Ning, K. Ali, L. Chen, C. Nugent, SRAM-PUF-Based Entities Authentication Scheme for Resource-Constrained IoT Devices, *IEEE Internet of Things Journal*, Vol. 8, No. 7, pp. 5904-5913, April, 2021. https://doi.org/10.1109/JIOT.2020.3032518

[23] A. O. G. Rivera, D. K. Tosh, J. C. Acosta, L. Njilla, Achieving Sensor Identification and Data Flow Integrity in Critical Cyber-Physical Infrastructures, *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Tempe, AZ, USA, 2020, pp. 1-6. https://doi.org/10.1109/SmartGridComm47815.2020.9302937

Biographies

Tung-Tsun Lee received the B.S. and M.S. degrees in computer science and engineering from National Chiao Tung University, Taiwan, in 1983 and 1985, respectively. Since 1992, he has been a Lecturer with the Department of Electronic Engineering, National Ilan University, Taiwan. His research interests include security, image processing and computer networks.

Shyi-Tsong Wu was born in Jiaoxi, Yilan County, Taiwan. He received his Ph.D. in Electronic Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2005. He is now a Professor at the Department of Electronic Engineering, National Ilan University, Yilan City, Taiwan. His research interests include IoT security and applications, cryptography, and electronic circuits.

Yao-Jen Liang received the Ph.D. degree in communication engineering from National Taiwan University, Taipei, Taiwan, in 2010. He has more than six years' experience in industry. From 2010 to 2011, he was a Postdoctoral Fellow at the Institute of Information Science, Academia Sinica, Taiwan. From February 2011 to February 2012, he was a Visiting Scholar at the School of Electrical and Computer Engineering, Georgia Institute of Technology. He is currently an Associate Professor with the Department of Electronic Engineering, National Ilan University. His current research interests include wireless communications and statistical signal processing.