Physically Secure Mutual Authentication for IoT

Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar
Department of ECE,
National University of Singapore, Singapore

***Abstract*—Many devices in the Internet of things (IoT) have special and challenging design requirements including limited size, energy, storage, and processing capabilities. Moreover, many IoT devices may be deployed in the open and in public places, making them vulnerable to physical and cloning attacks. These characteristics dictate that any security protocol designed for IoT devices should not only be efficient but should also provide security even if an IoT device is captured by an adversary. To solve this issue, we present mutual authentication protocols for IoT devices that are not only efficient but also secure against physical and cloning attacks. To provide security to physically unprotected devices, the proposed protocols use physical unclonable functions (PUFs) and avoid storing sensitive information on the device. A security and performance analysis of the proposed protocols is presented.**

I. Introduction

IoT systems will change the way we work, think, and interact with objects as well as with each other. The explosive growth of autonomously communicating "devices" has caused the devices to outnumber human beings by a ratio of 1.5 to 1 [1]. IoT devices typically have a low cost, with limited processing power and energy. Any protocol or application designed to run on IoT devices, including those for security, thus needs to be very efficient in terms of computational complexity and energy requirements. Moreover, traditional security protocols for the Internet were designed for physically protected devices such as personal computers. However, IoT devices may be easily accessible by an adversary. Thus, the simple nature, no physical protection and constrained resources put security at the forefront for designing IoT systems.

As an alternative paradigm for providing security primitives, physical or physically unclonable functions have gained popularity in the security domain and their practicality has been demonstrated by recent works. PUFs are a result of the manufacturing process of Integrated Circuits (ICs) which introduces random physical variations into the micro structure of an IC, making it unique. These variations in the micro structure of an IC cannot be controlled, making them virtually impossible to clone or duplicate. PUFs are ICs which use their internal structure to provide a one-way function that can not be duplicated. The fact that PUFs are hard to predict but easy to construct and evaluate makes them a good choice for use as security primitives for IoT devices.

The focus of this work is to design an authentication protocol for IoT systems, which is not only secure against other forms of compromise but also provides superior protection against physical and cloning attacks. For this purpose, we use a challenge-response based mechanism using PUFs. The use of PUFs serves the purpose of protecting a device from being cloned. Our design requirements also include not storing any sensitive information on an IoT device to avoid leakage of data or security keys. We desire to meet these requirements with scalability, i.e., keeping the storage requirements at the server to the minimum.

The rest of the paper is organized as follows. In Section II we discuss the related work. Section III presents a brief introduction to PUFs, the network model, assumptions and attack model. The proposed mutual authentication protocols are presented in Section IV. Section V proves the correctness of our protocols. The security and performance analysis of the proposed protocols are presented in Sections VI and VII. Finally we conclude the paper in Section VIII.

II. Literature Review

Existing literature for authentication in IoT systems are either too computationally complex or require sensitive information to be stored in the IoT device [2], [3], [4], [5]. The existing literature on using PUFs for authentication is not extensive. Most of the work is focused reliably computing a PUF response to a challenge [6], [7]. Similarly, some literature describe techniques for implementing authentication protocols on reconfigurable hardware for the purpose of intellectual property (IP) protection [8], [9]. In other work, PUFs are used for designing protocols for wireless sensor networks (WSNs) and radio frequency identification (RFID) systems [6], [10], [11]. However, most of these protocols store secret keys on the device's memory. Moreover, most of the PUF based protocols are not scalable in the sense that they require the server to store a large number of parameters for each device. Similarly, existing authentication protocols based on smart cards also require some information to be stored on the card. In contrast, the proposed protocols do not store any secret information on the device's memory and are scalable at the same time.

In a related work, the authors of [12] propose an authentication protocol based on PUFs. The protocol does not require any secrets on the device and is also scalable. However, their protocol requires a user to input a password for authentication as well as enrollment with the server. Since most IoT devices are not operated by humans, this protocol is not suitable for IoT systems. This paper is an extension of our previous work [13], and provides a mechanism for two way device authentication as well as a formal verification of the protocols.

978-1-5090-5569-2/17/$31.00

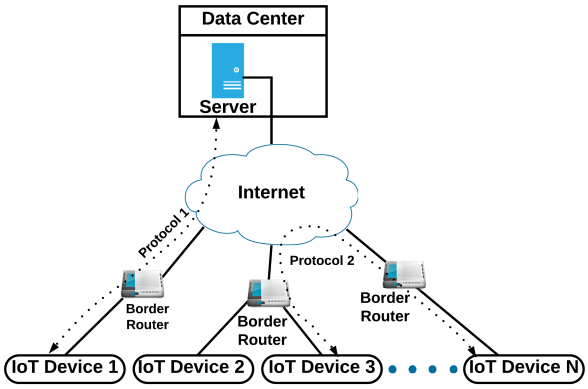

Fig. 1: Network Model

III. Background, Network Model and Assumptions

A. Preliminary Background

In this section we present a short description of PUFs. A PUF can be considered as a function that maps a set of challenges to a set of responses based on a unique physical micro structure. A PUF has the following properties:

- Output depends on a physical system.
- Easy to evaluate and construct.
- Output is unpredictable and looks like a random function.
- Virtually impossible to duplicate or clone a PUF [14].

A PUF is characterized by a challenge-response pair (CRP). It is an IC which takes a string of bits as an input challenge and produces a string of bits called the response. The response R of a PUF P to a challenge C can be represented as follows:

$$R = P(C). \tag{1}$$

B. Network Model

We assume each IoT device is equipped with a PUF. The IoT devices are connected with a data center using the Internet through border router elements such as 6LoWPAN. Figure 1 describes our network model.

C. Assumptions

In this paper, we make the following assumptions regarding the system:

a. An IoT device consists of a microcontroller attached to a PUF. As the PUF output depends on its unique physical characteristics, any attempt to tamper with the PUF changes the behavior of the device and renders the PUF useless.
b. An IoT device is considered to be a system on chip, and it is not possible to tamper with the communication between the micro controller and its PUF [15], [16].
c. IoT devices have limited resources while the servers in the data center are considered secure and have no such limitation.
d. We denote the ID of an IoT device, XOR operation, hash of X, and concatenation by ID_x, $\oplus$, $H(X)$, and $\|$, respectively. Moreover, the challenge and response for the i-th round of the protocol are denoted by C^i and R^i, respectively. An expression Ex evaluated using the values from a received message is represented by $[Ex]_{REC}$.

D. Attack Model

We assume that the IoT devices may be deployed out in the open and are not physically protected. An adversary can easily access the device and subject it to physical or cloning attacks. We assume an adversary can compromise one or more network entities and can inject packets, eavesdrop, initiate a session, replay older messages and mimic other devices. We assume that the adversary aims to launch an undetectable attack to authenticate itself with the server or any of the IoT devices.

IV. Proposed Mutual Authentication Protocols

In this section we describe the proposed mutual authentication protocols. Separate protocols for mutual authentication are presented for two scenarios: (i) mutual authentication between an IoT device and a server in the data center, and (ii) mutual authentication between two IoT devices.

The initial CRP for a device is obtained by the server when a device is first deployed in the field. An operator inputs a password into the device and the device can exchange an initial random CRP with the server using a one time password (OTP) authentication mechanism. ID_A, and the CRP (C^i, R^i) for each IoT device are stored at the server. However, an IoT device does not need to store anything.

A. Protocol 1: IoT Device and Server Mutual Authentication

When an IoT device wants to authenticate itself with the server, the proposed mutual authentication protocol is shown in Figure 2. Steps for the protocol are as follows:

1) The IoT device sends a randomly generated nonce along with its identity to the server as shown in message 1 in Figure 2.
2) The server searches its memory for ID_A, and the authentication request is rejected if the search fails. Otherwise, the server reads the corresponding CRP (C^i, R^i) for this device from its memory. The server then sends message 2 to ID_A as shown in Figure 2. Message 2 contains a message authentication code (MAC) to ensure data integrity and freshness. The last parameter in this MAC i.e., N_A is the freshness identifier for the source (the server in this case). The remaining parameters ensure data integrity. The same approach is followed for message freshness, source identifier, and data integrity throughout the protocol.
3) IoT device ID_A uses its PUF to generate R^i using the challenge C^i. The device then computes N_A using R^i as follows

$$N_A = R^i \oplus \left[R^i \oplus N_A\right]_{REC}. \tag{2}$$

ID_A verifies the MAC using the parameters in its memory and if the verification fails, authentication is terminated. Otherwise, the IoT device constructs the new challenge $C^{i+1} = H(N_A \parallel N_B)$ using a new random

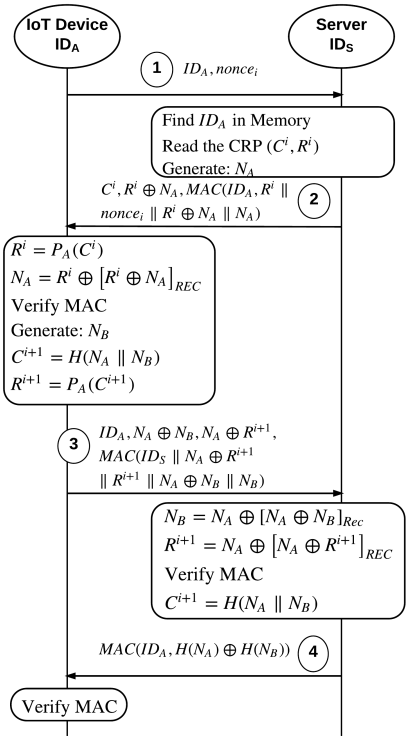

Fig. 2: Mutual authentication for IoT device and server.

number N_B. ID_A then inputs C^{i+1} into its PUF to obtain the new response R^{i+1}. This new CRP (C^{i+1}, R^{i+1}) will be used for future authentications. The IoT device then sends message 3 to the server.

4) The server computes N_B and R^{i+1} using the stored secret N_A. The server then verifies the MAC and sends message 4 to ID_A if the verification is successful. Otherwise, the authentication is rejected.
5) ID_A verifies the MAC received in message 4. If the verification fails the authentication is terminated. Otherwise, authentication is considered complete.

N_A and N_B can also be used to construct a session key between ID_A and the server as follows: $H(N_A) \oplus H(N_B)$. It should be noted that any compromise in this secret key does not jeopardize the security of the system. An adversary can not obtain R^i from the secret key, and thus cannot construct "valid data".

B. Protocol 2: Mutual Authentication for Two IoT Devices

In this section we present a mutual authentication and key exchange protocol for the scenario when two IoT devices want to authenticate each other and form a session. Let us assume IoT device ID_A wants to establish a session with another IoT device ID_B. Figure 3 shows the mutual authentication protocol for this scenario. The steps for the proposed protocol are as follows:

1) The IoT device that wants to initiate authentication sends its ID and a nonce to the IoT device with which it wants to communicate, i.e., ID_B, as shown in Figure 3.

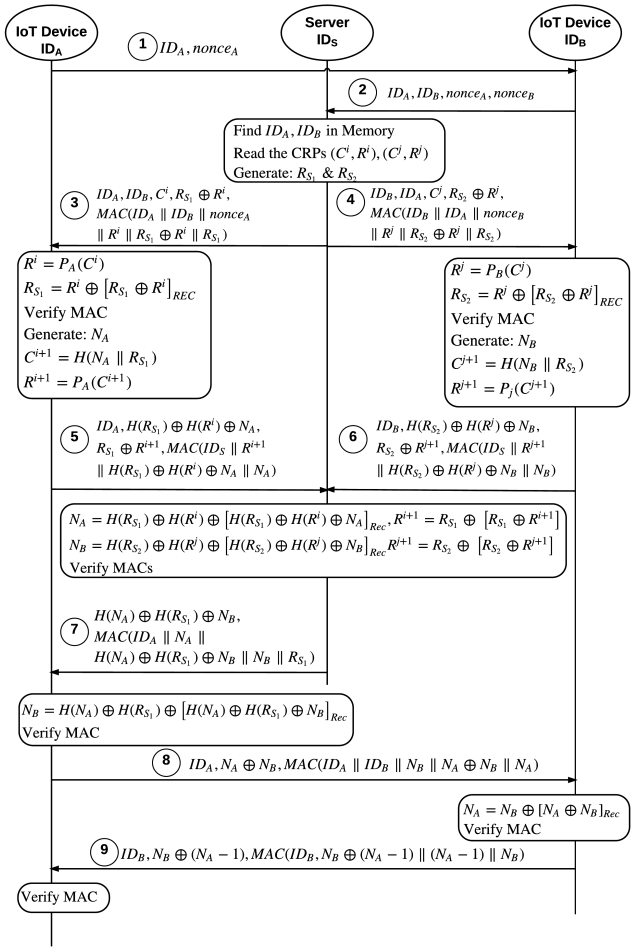

Fig. 3: Mutual authentication of two IoT devices.

2) ID_B sends message 2 to the server which contains the IDs and nonces of the two IoT devices.
3) The server searches its memory and reads the respective CRPs (C^i, R^i), and (C^j, R^j) for ID_A and ID_B. The server then sends message 3 to ID_A and message 4 to ID_B.
4) ID_A and ID_B generate R^i and R^j, respectively, using their respective PUFs. ID_A and ID_B compute R_{S_1} and R_{S_2} using R^i and R^j respectively. Both IoT devices verify the MACs, and if the verification fails, the concerned IoT device does not respond and terminates the current authentication attempt. Otherwise, the IoT devices generate random numbers N_A and N_B, and compute their respective new CRPs using their PUFs. ID_A sends message 5, while ID_B sends message 6 to the server.
5) The server computes N_A, N_B, R^{i+1}, R^{j+1} and verifies the respective MACs for data integrity and freshness. If the verification fails the server rejects the authentication. Otherwise, the server sends the random number generated by ID_B, i.e., N_B in message 7 to ID_A.
6) IoT device ID_A computes N_B using N_A and R_{S_1}. ID_A verifies the MAC in message 7, and if the verification fails the authentication is terminated. Otherwise ID_A sends its

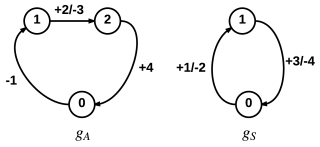

Fig. 4: Directed Graph/FSM for Protocol 1

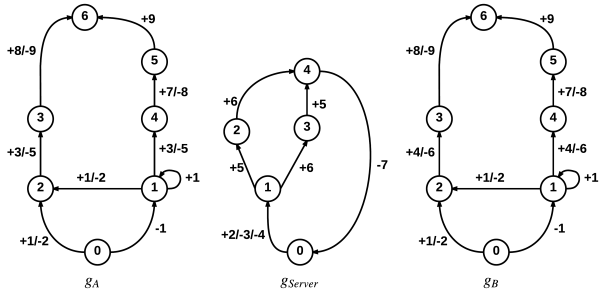

Fig. 5: Directed Graph/FSM for Protocol 2

own random number, i.e., N_A to ID_B in message 8.

7) IoT device ID_B computes N_A using N_B in its memory and verifies the MAC in message 8. If the verification fails the authentication request is rejected. Otherwise, ID_B sends $N_A - 1$ back to ID_A in message 9.
8) ID_A verifies $N_A - 1$ and the MAC received in message 9. If the verification fails the authentication is rejected. Otherwise, the verification is complete and ID_A and ID_B have successfully authenticated each other.

Similar to Section IV-A, the two IoT devices can now use N_A and N_B to establish a secret symmetric key.

V. Protocol Verification

In this section we present a formal verification of the correctness of the proposed protocols. To prove correctness we show that the proposed protocols possess the following properties [17]:

1) **Completeness**: The protocol is able to accept all valid inputs.
2) **Deadlock Freeness**: The protocol does not enter a state such that it stays in that state indefinitely.
3) **Livelock or Tempo-blocking freeness**: The protocol does not enter into an infinite loop.
4) **Termination**: When starting from the initial state, the protocol is always able to reach a well-defined final state.
5) **No non-executable interactions**: The protocol only contains transmission, reception, and interaction paths that are realized under normal operating conditions.

We use the technique proposed in [17] to prove the correctness of the proposed protocols. In this technique we first create a directed graph for each entity of a protocol being verified. The directed graph can be considered as a finite state machine (FSM) for that entity. In protocol 1 we have two entities (an IoT device and a server), and the directed graphs for these entities are shown in Figure 4. Similarly, the directed graphs for the three entities of protocol 2 are shown in Figure 5. In these figures, g_A, g_B, and g_S represent the directed graphs for the FSMs of ID_A, ID_B, and the server, respectively. In this section we refer to ID_A, and ID_B as A and B, respectively, in the figures. The state of a protocol machine is represented by the labeled circles. -m (respectively, +m) on the directed arcs represent a transmission (reception) of message m. Moreover, +m/-n represents the reception of message m followed by the transmission of message n. For example, one run of protocol 1 corresponds to the following interaction paths for g_A and g_S:

- g_A: [0] -1[1] +2/-3[3] +4[0]
- g_S: [0] +1/-2[1] +3/-4[0]

where "[]" represents a state in Figure 4. The above sequence of activities for IoT device ID_A can be interpreted as: ID_A starts in state 0, sends message 1 to the server and enters state 1, receives message 2 and transmits message 3 to enter state 3, and finally receives message 4 to enter state 0 again. The sequence of activities for the server can be interpreted in a similar fashion. Note that the final state for both ID_A and the server for protocol 1 is S_0, while the final states for ID_A, ID_B, and the server for protocol 2 are S_6, S_6, and S_0, respectively.

The next step to prove the correctness of the proposed protocols is to perform a reachability analysis technique proposed in [19], [17]. In this analysis we represent the overall state of the system (consisting of all entities in the protocol) as a matrix. The state matrices for protocol 1 and 2 are given in (3) and (4), respectively, as

$$\begin{bmatrix} A & A \rightarrow Server \\ STATE & CHANNEL \\ Server \rightarrow A & Server \\ CHANNEL & STATE \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} A & A \rightarrow B & A \rightarrow Server \\ STATE & CHANNEL & CHANNEL \\ B \rightarrow A & B & B \rightarrow Server \\ CHANNEL & STATE & CHANNEL \\ Server \rightarrow A & Server \rightarrow B & Server \\ CHANNEL & CHANNEL & STATE \end{bmatrix}. \tag{4}$$

The elements of the above matrices represent the current state of the FSM of an entity and the messages sent by the entity. For example, when the protocol starts all the entities will be in their initial state, i.e., state 0 in Figures 4 and 5. Now let us assume ID_A sends message 1 to the server in protocol 1. From the FSM of ID_A in Figure 4 we can see that ID_A transitions into state 1 after sending message 1. The matrix for the overall state of the system can then be constructed as follows:

$$\begin{bmatrix} S_1 & 1 \\ E & S_0 \end{bmatrix} \tag{5}$$

The matrix in (5) shows that ID_A is in state 1 (element at row 1 and column 1), while the element at row 1 and column 2 shows the contents of the channel from ID_A to the server, i.e., the message sent from ID_A to the server, which in this case is message 1. Similarly, in this example the server is currently in state 0 denoted by S_0 and has not sent any message to ID_A, represented by the E at row 2 and column 1. We denote the overall state of the system by SSi, while the constituent states

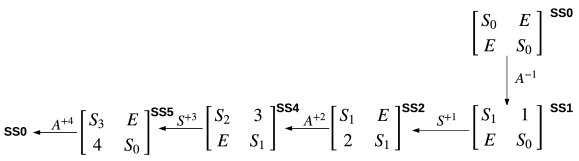

Fig. 6: Reachability Analysis for Protocol 1

of the subsystems (entities) are denoted by S_i.

Figures 6 and 7 show the results of the reachability analysis for protocols 1 and 2, respectively. The reachability analysis always starts in the initial state $SS0$. All the protocol entities are in their respective initial states, i.e., S_0 and all the channels are empty, i.e., E. Moreover, a transition from one system state to another caused by the transmission (reception) of message i by entity X is denoted by X^{-i} (X^{+i}) on the respective directed arcs.

Figure 6 shows that protocol 1 starts with a transition from $SS0$ to $SS1$ when ID_A sends message 1, followed by subsequent transitions. This figure shows the transitions of the overall system state caused by all the possible inputs, implying completeness of the protocol. A potential deadlock state is defined as an overall system state which is not an initial or final state and does not have any messages in any channel. Figure 6 shows that the protocol does not have any potential deadlock states, implying deadlock freeness. Moreover, the analysis covers all the possible interaction paths, transmissions, and receptions, and shows that the protocol does not contain any infinite loops and always ends up in state $SS0$. This implies the remaining three properties, i.e., livelock freeness, termination, and absence of non-executable interactions. This shows the correctness of protocol 1.

Similarly, the reachability analysis for protocol 2 shows that we always start from the initial state $SS0$ and terminate at the final state $SS18$. $SS18$ is the final state because all the entities are in their respective final states and the channels are empty. Note that the overall system state $SS19$ involves a decision based on the message contents of the protocol. The system reaches $SS19$ when both IoT devices ID_A and ID_B try to initiate authentication at the same time. We assume that the protocol uses the nonces provided by ID_A and ID_B as the tie breaker, e.g., if $nonce_A > nonce_B$ then ID_A gets to initiate the authentication and ID_B sends message 2 to the server. Accordingly, the state machine for ID_B transitions to state 2 as shown by the overall system state $SS3$.

Figure 7 shows that protocol 2 accepts all valid messages implying the completeness property. We also observe that the protocol has one potential deadlock state i.e., $SS19$. However, $SS19$ has outgoing transitions depending on a decision based on the nonces of ID_A and ID_B as explained above. Therefore the deadlock is resolved and the protocol can be considered deadlock free. There are no loops among the overall systems states implying livelock or tempo-blocking freeness. Figure 7 covers all the transmissions, interaction paths, receptions, and states, and we see that following any of the interaction paths we always end up in the final state. This shows that the protocol possesses the termination property and does not have any non-executable interactions. Thus, the proposed protocol is proved to be correct.

VI. Security Analysis

In this section we analyze the security of the proposed mutual authentication protocols. The automated security verification tool ProVerif (PV) [20] was used to thoroughly simulate and verify the security properties of the proposed protocols. PV has the ability to prove reachability properties, correspondence assertions, and observational equivalence. Although PV may not be able to prove a property, however, when PV says a property holds, the model does guarantee that property. The IoT devices and the server are modeled as separate processes. To simulate arbitrarly many sessions of the protocol between the entities, we instantiate an unbounded number of instances of these processes. This simulates arbitrarly many sessions of the protocol between the two parties. The simulation scripts for protocol 1 and protocol 2 can be found at [21].

A. Security Analysis For Protocol 1

Mutual authentication of the principals ID_A and ID_S is the primary objective of the protocol. Thus, when ID_A reaches the end of the protocol, she should be assured that she has indeed completed the protocol with ID_S. Similarly, when ID_S completes a run of the protocol, he should be assured that he has indeed done so with IoT device ID_A. The security of mutual authentication in the proposed protocols was evaluated using correspondence assertions. For this purpose we declare the following events in PV:

- **event** $beginAfull(ID_A, ID_S, N_A, N_B)$, is used by ID_S to represent the start of a protocol run by the IoT device ID_A with N_A and N_B as the shared secrets.
- **event** $endAfull(ID_A, ID_S, N_A, N_B)$, is used by ID_A to record the belief that she has successfully completed the protocol with ID_S and agrees to the shared secrets N_A and N_B.
- **event** $beginBfull(ID_A, ID_S, N_A, N_B)$, is used to record ID_A's intention to launch the protocol with ID_S with the given protocol parameters.
- **event** $endBfull(ID_A, ID_S, N_A, N_B)$, is used to represent ID_S's belief that he has successfully completed the protocol with ID_A with the given protocol parameters.

These events are used to prove authentication properties for protocol 1. The authentication properties we intend to prove for protocol 1 are as follows:

1) **Authentication of ID_S to ID_A:** ID_A is only willing to share her data with the server ID_S. Thus, if she completes the protocol, she has indeed executed the protocol with ID_S. This implies authentication of ID_S to ID_A holds. The correspondence assertion used to prove this property in PV is as follows:

$$inj\text{–}event(endBfull(\cdots)) ==> inj\text{–}event(beginBfull(\cdots)). \quad (6)$$

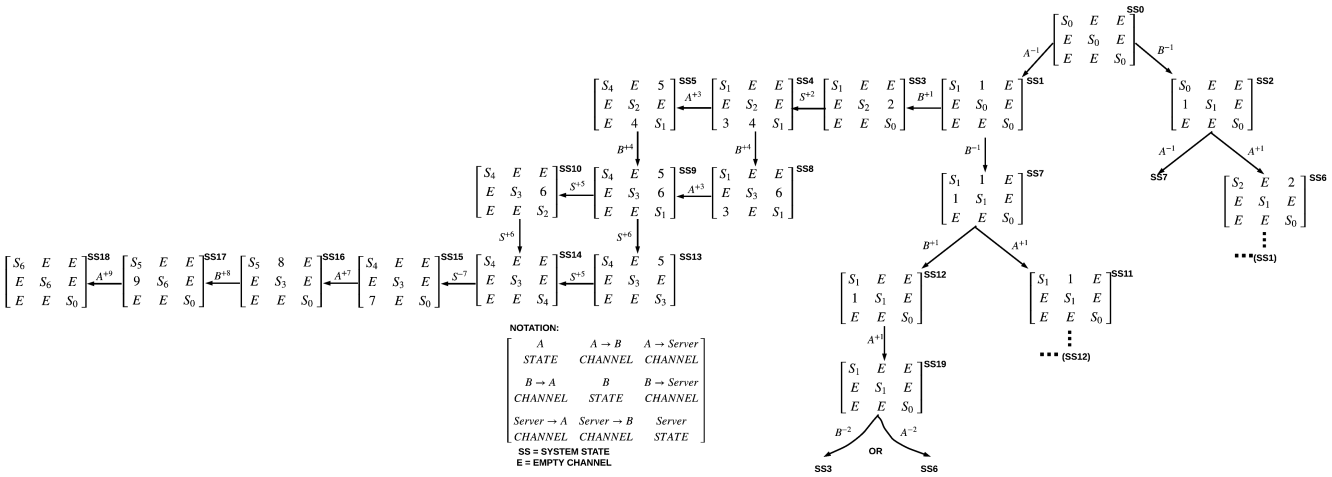

Fig. 7: Reachability Analysis for Protocol 2

Note that the statement $event_A ==> event_B$ is used to check the fact that whenever there is an occurrence of the event $event_A$, it must always be preceded by the event $event_B$.

2) **Authentication of ID_A to ID_S:** Server ID_S is willing to establish a session with any of the IoT devices in its clientele. Thus, if he runs the protocol with ID_A, he only requires authentication from ID_A to ID_S to hold. The correspondence assertion used to prove this property in PV is as follows:

$$inj - event(endAfull(\cdots)) ==> inj - event(beginAfull(\cdots)). \quad (7)$$

Moreover, PV can also be used to establish the (syntactic) secrecy of N_A, N_B, and R^{i+1} after the protocol is successfully executed. The following queries in PV are used to prove the secrecy of the secrets in the proposed protocol.

$$query \quad attacker(ANa); attacker(BNa); \quad (8)$$
$$attacker(ANb); attacker(BNb); \quad (9)$$
$$attacker(AR\_new); attacker(SR\_new) \quad (10)$$

where, ANa, ANb, and AR_new are used to prove that N_A, N_B, and R^{i+1} can be considered good secrets on the site of principal ID_A. Similarly, BNa, BNb, and BR_new are used to prove the secrecy of N_A, N_B, and R^{i+1} on the site of principal ID_S. PV has the ability to identify any definite/possible attack, and therefore, we conclude that the proposed protocol is secure against different types of attacks.

B. Security Analysis for Protocol 2

In this section we discuss the security of the proposed mutual authentication protocol for two IoT devices. The events declared to prove mutual authentication between ID_A and ID_B in PV for protocol 2 are as follows:

- **event** $beginAfull(ID_A, ID_B, N_A, N_B)$, is used by ID_B to record the initiation of the protocol by the IoT device ID_A with N_A and N_B as the shared secrets.
- **event** $endAfull(ID_A, ID_B, N_A, N_B)$, meaning ID_A believes that she has successfully completed the protocol with ID_B and both agree on the shared secrets N_A and N_B.
- **event** $beginBfull(ID_A, ID_B, N_A, N_B)$, meaning ID_A intends to initiate the protocol with ID_B using the given protocol parameters.
- **event** $endBfull(ID_A, ID_B, N_A, N_B)$, denotes ID_B's belief that she has successfully completed the protocol with ID_A with the given protocol parameters.

The authentication properties proved for protocol 2 are as follows:

1) **Authentication of ID_B to ID_A:** ID_A wants to establish a session with ID_B. Thus, if she completes the protocol, she has indeed executed the protocol with ID_B. This also implies that ID_A and ID_B agree on the set of given protocol parameters. The correspondence assertion used to prove this property in PV is as follows:

$$inj - event(endBfull(\cdots)) ==> inj - event(beginBfull(\cdots)). \quad (11)$$

2) **Authentication of ID_A to ID_B:** If ID_B thinks she has completed the protocol with ID_A, she indeed did so with ID_A. The correspondence assertion used to prove this property in PV is as follows:

$$inj - event(endAfull(\cdots)) ==> inj - event(beginAfull(\cdots)). \quad (12)$$

Similarly, we also define separate events and correspondence assertions in PV to prove the mutual authentication between ID_A and ID_S, and ID_B and ID_S. The details can be found in the simulation scripts [21]. These authentication properties are important because the IoT devices update their CRPs with the server.

To establish the secrecy of the secrets in Protocol 2, we use

TABLE I: Computational Burden

Task	IoT Device	Server
Protocol 1	$1N_H + 3N_{MAC} + 3N_{\oplus}$	$3N_H + 3N_{MAC} + 3N_{\oplus}$
[12]	$2N_H + 2N_{exp} + N_{\times}$	$1N_H + 3N_{exp}$
Protocol 2	$5N_H + 5N_{MAC} + 7N_{\oplus}$	$7N_H + 5N_{MAC} + 10N_{\oplus}$

the following queries in PV:

$$query \quad attacker(ANa); attacker(ANb); \tag{13}$$
$$attacker(BNa); attacker(BNb); \tag{14}$$
$$attacker(SNa); attacker(SNb); \tag{15}$$
$$attacker(ARs1); attacker(SRs1); \tag{16}$$
$$attacker(BRs2); attacker(SRs2); \tag{17}$$
$$attacker(AR\_new); attacker(SR\_newA); \tag{18}$$
$$attacker(BR\_new); attacker(SR\_newB) \tag{19}$$

where ANa, ANb, $ARs1$, and AR_new are used to check the secrets N_A, N_B, R_{S_1}, and R^{i+1} on the site of principal ID_A. The rest of the queries are used in a similar fashion to prove the secrecy of the secrets on the site of principal ID_B and ID_S. Using these queries and correspondence assertions we have established the security of protocol 2.

C. Protection against Physical Attacks and Cloning

An adversary may get easy access to IoT devices. Therefore, it is desirable that IoT devices should be safe against cloning and physical attacks. and do not store any secrets within the device. Therefore, The proposed protocols require each IoT device to be equipped with a PUF which makes them safe against cloning [6], [18]. Moreover, the PUFs are used to generate the secrets whenever needed and do not rely on any stored secrets. Stored secrets can lead to leakage of keys using physical attacks. Most of the existing authentication protocols proposed for the IoT rely on one or more secrets (in the form of keys) to be stored in a device's memory. However, this approach can lead to leakage of keys using physical attacks. The proposed mutual authentication protocols do not use any stored secrets. Moreover, the PUF and micro-controller of an IoT device are considered inseparable [15]. Therefore, we can conclude that even if an adversary has access to an IoT device, he/she can not compromise the security of the proposed protocols.

VII. Performance Analysis

IoT devices have limited resources. Therefore, any protocol designed for IoT systems should be efficient in terms of memory, processing, energy, storage, and communication overhead. In this section we compare the performance of our protocol with the most relevant PUF based authentication protocol, proposed by Frikken et al. [12].

A. Computational Efficiency

Table I shows the number of hash (N_H), MAC (N_{MAC}), exclusive-or (N_{oplus}), modular exponentiation (N_{exp}), and modular multiplication ($N_{\times}$) operations required by the proposed mutual authentication protocols, and the protocol proposed by Frikken et al. [12]. Note that these values can be directly obtained by counting the occurrence of the respective operation in Figures 2 and 3.

If we assume the use of message authentication codes using universal hashing (UMACs) the complexity of calculating the hash function and MAC is be $O(n)$ [22], [23]. Therefore, the complexity of both proposed protocols is $O(n)$ on the IoT device side as well as the server side, where n is the number bits in the response of the PUF (128 bits in our case). However, it can be shown that the complexity of [12] is $O(n + M(l)k)$ on the user as well as the server side, where $M(l)$ denotes the complexity of a general modular multiplication with l bit operands, and k is the exponent. $M(l)$ is generally quadratic in l [24]. This shows that the computational complexity of the proposed mutual authentication protocols is lower.

TABLE II: Parameter Lengths

Parameter	Size (bits)
ID	8 [28]
N_A, N_B,R_{S_1}, R_{S_2}	128 [25], [29]
C, R	128 [25]
MAC	32/64/96/128 [26]

B. Communication Overhead

Let us assume the parameter sizes given in Table II. Looking at Figures 2 and 3, we observe that message 3 is the longest message in Protocol 1 as well as Protocol 2. We assume the use of UMAC as the MAC for data integrity. UMAC provides the flexibility of different levels of security by offering MACs of varying lengths as given in Table II [26]. Using the maximum MAC size of 128 bits, the length of message 3 in Protocol 1 is 48 bytes while the length of message 3 in protocol 2 is 50 bytes. This shows that the messages of the proposed protocols can fit in a single protocol data unit (PDU) of 6LoWPAN that has a maximum transmission unit (MTU) size of 127 bytes [27]. Moreover, the length of the longest message in [12] is approximately 68 bytes which is much larger than the proposed protocols.

C. Storage Requirement

The proposed protocols are very efficient in terms of storage requirements. Variables such as N_A, N_B, R_{S_1}, and R_{S_2} are only stored temporarily during the authentication process and deleted afterward. Moreover, only one CRP pair (C^i, R^i), and the ID are stored for each IoT device in the server. In contrast, most of the protocols in existing literature either require the IoT devices to store secret information or the server needs to store a large number of CRPs for each IoT device. However, these approaches are vulnerable to physical attacks and are not scalable. The proposed mutual authentication protocols do not impose these kind of requirements and do not require any stored secrets.

VIII. Conclusions

In this paper we presented two mutual authentication protocols: for communication between an IoT device and a server,

and between two IoT devices. We showed that the system remains safe even if an adversary has physical access to an IoT device. The proposed protocols provide the desired security characteristics efficiently by exploiting the inherent security features of PUFs. It is shown that the IoT devices do not need to store any secrets (such as keys). Moreover, the storage requirements at the server are also very low. The proposed protocols can also be used to establish session keys.

REFERENCES

[1] *The Internet of Things Reference Model,* CISCO, 2014.

[2] V. Shivraj et. al., "One time password authentication scheme based on elliptic curves for Internet of Things (IoT)," *Proceedings of NSITNSW,* pp. 1-6, Riyadh, KSA, Feb 2015.

[3] P. Porambage et. al., "Two-phase Authentication Protocol for Wireless Sensor Networks in Distributed IoT Applications," *Proceedings of IEEE WCNC,* pp. 2728-2733, Istanbul, Turkey, April 2014.

[4] Y. Kim et. al.,"DAoT: Dynamic and Energy-aware Authentication for Smart Home Appliances in Internet of Things," *Proceedings of IEEE ICCE,* pp.196-197, Las Vegas, NV, Jan 2015.

[5] V. Petrov et. al.,"Towards the Era of Wireless Keys: How the IoT Can Change Authentication Paradigm," *Proceedings of IEEE WF-IoT,* pp.51-56, Seoul, South Korea, Mar 2014.

[6] G. E. Suh, and S. Devadas "Physical Unclonable Functions for Device Authentication and Secret Key Generation," *Proceedings of IEEE/ACM DAC,* pp. 9-14, San Diego, CA, June 2007.

[7] E. Ozturk et. al., "Towards Robust low cost authentication for pervasive devices", *Proceeding of IEEE PerCom,* pp. 170-178, 2008.

[8] E. Simpson, and P. Schaumont, "Offline hardware/software authentication for reconfigurable platforms", *In: L. Goubin, M. Matsui, (eds.) CHES 2006.* LNCS, vol. 4249, pp. 311-323, Springer, Heidelberg 2006.

[9] J. Guajardo et. al., "Physical unclonable functions and public key crypto for FPGA IP protection", *International Conference on Field Programmable Logic and Applications,* pp. 189-195, 2007.

[10] P. Cotese et. al., "Bernardo, Efficient and Practical Authentication of PUF-Based RFID Tags in Supply Chains," *Proceedings of IEEE RFIDTA,* pp. 182-188, Guangzhou, China, June 2010.

[11] Y. S. Lee et. al., "Mutual Authentication in Wireless Body Sensor Networks (WBSN) based on Physical Unclonable Function (PUF)," *International Wireless Communications and Mobile Computing Conference (IWCMC),* pp.1314-1318, Sardinia, July 2013.

[12] K. Frikken et. al., "Robust Authentication Using Physically Unclonable Functions", *In: P. Samarati et al. (eds.): ISC 2009,* LNCS 5735, pp. 262-277, Springer, Heidelberg 2009.

[13] M. Aman, K. C. Chua and B. Sikdar, "Physical Unclonable Functions for IoT Security," *Proceedings of ACM IoTPTS,* pp. 10-13, Xian, China, June 2016.

[14] C. Bohm, and M. Hofer, "Physical Unclonable Functions in Theory and Practice," Springer, 2012.

[15] S. Guilley, and R. Pacalet, "SoCs security: a war against side-channels", *Annals of Telecommunications,* Vol. 59, no. 7, pp 998-1009, 2004.

[16] M. Kirkpatrick et. al., "System on Chip and Method for Cryptography using a Physically Unclonable Function," U.S. Patent 8750502 B2, issued March 22, 2012.

[17] D. P. Sidhu, "Authentication protocols for computer networks: I", *Computer Networks and ISDN systems,* Vol. 11, pp. 287-310, 1986.

[18] P. Tuyls, and L. Batina, "RFID-tags for Anti-Counterfeiting, Topics in Cryptology CT-RSA", *Lecture Notes in Computer Science*, Vol. 3860, pp.115-131, San Jose, CA,2006.

[19] V. Varadharajan, "Verification of network security protocols", *Computers and Security,* Vol. 8, no. 8, pp. 693-708, 1989.

[20] B. Blanchet and B. Smyth, *ProVerif: Automatic Cryptographic Protocol Verier, User Manual and Tutorial*.

[21] https://www.ece.nus.edu.sg/stfpage/bsikdar/scripts/DSC17.

[22] M. Babka, "Properties of Universal Hashing," Charles University in Prague, Master Thesis, 2010.

[23] Y. Mansour, N. Nissan and P. Tiwari, "The Computational Complexity of Universal Hashing," *Theoretical Computer Science,* vol. 107, no. 1, pp. 121-133, 1993.

[24] T. Kivinen and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)," IETF RFC 3526, May 2003.

[25] M. Katagi and S. Moriai, "The 128-bit blockcipher CLEFIA," IETF RFC 6114, March 2011.

[26] T. Krovetz, "UMAC: Message Authentication Code using Universal Hashing", IETF RFC 4418, March 2006.

[27] G. Montenegro et. al., "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," IETF RFC 4944, September 2007.

[28] P. Kim, " IoT Specific IPv6 Stateless Address Autoconfiguration with Modified EUI-64," IETF Internet-Draft, July 2015.

[29] D. Whiting et. al., "Counter with CBC-MAC (CCM)," IETF RFC 3610, September 2003.