Mutual Authentication in IoT Systems using Physical Unclonable Functions

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

***Abstract*—The Internet of Things (IoT) represents a great opportunity to connect people, information, and things, which will in turn cause a paradigm shift in the way we work, interact, and think. IoT devices are usually small, low cost and have limited resources, which makes them vulnerable to physical, side-channel, and cloning attacks. Therefore, any protocol designed for IoT systems should not only be secure but also efficient in terms of usage of chip area, energy, storage, and processing. To address this issue, we present light-weight mutual authentication protocols for IoT systems based on Physical Unclonable Functions (PUFs). Protocols for two scenarios are presented, one when an IoT device and server wish to communicate and the other when two IoT devices want to establish a session. A security and performance analysis of the protocols shows that they are not only robust against different types of attacks, but are also very efficient in terms of computation, memory, energy, and communication overhead. The proposed protocols are suitable for real time applications and are an attractive choice for implementing mutual authentication in IoT systems.**

I. Introduction

The number of IoT devices has been increasing rapidly over the past decade, outnumbering humans by a ratio 1.5 to 1 as of 2014 [1]. IoT is envisioned as the enabling technology for smart cities, power grids, health care, and control systems for critical installments and public infrastructure. This diversity, increased control and interaction of devices, and the fact that IoT systems use public networks to transfer large amounts of data make them a prime target for cyber attacks. IoT security and human safety are tied to each other, e.g., causing accidents by disrupting vehicular networks, putting a patient at risk by tampering with a body network, causing blackouts by interfering the smart grid, and causing accidents in nuclear reactors etc.

Network and Internet security has been an area of active research over the last two decades. However, existing security mechanisms are far from perfect and major security breaches involving personal, corporate and government data are reported regularly. Similar to the Internet, IoT systems have a long way to go in terms of achieving the desired security level. The major areas of concern in the context of IoT systems are secure booting, authentication, privacy protection, data integrity, user profiling and tracking, access control, and digital forgetting [2], [3], [4].

IoT devices can be considered as embedded systems that have the following properties. First, they are designed for a specific task and have enough memory and processing power for that task [2], [3]. Second, IoT devices are considered "headless" because these devices are not operated by a human and have to make their own decisions [3]. Finally, many IoT devices will not have any batteries or a constant power source; they will rely on energy harvesting or wireless transfer of energy [2]. These properties make the task of designing security protocols for IoT systems very challenging. Thus, any security protocol designed for IoT systems should have very low processing, communication, and memory requirements.

The existing security protocols and techniques of the Internet cannot be adopted for IoT systems due to two reasons [3], [4]. First, the implicit assumption of the nodes having unlimited power and memory is not valid in IoT systems. Second, Internet devices are considered to be physically well-protected. Unlike the Internet, IoT devices are small, simple, low cost, and are installed in locations where an adversary can capture them easily. Therefore, physical security of IoT devices is a major concern. For example, keys stored in the memory may be read off a physically captured device and used by an adversary to launch an attack. To address these issues, we consider the use of hardware security primitives based on physical unclonable functions, which work on a challenge-response mechanism. In this paper we show how PUFs can be used to eliminate the requirement of storing secrets in the nodes of a network. PUFs take advantage of the inherent variability of the manufacturing/fabrication process of integrated circuits (ICs). The output of the challenge-response mechanism of a PUF is correlated to the input and the physical (sub-)microscopic structure of the device. The random and uncontrollable effects of the IC fabrication process result in the uniqueness of PUFs i.e., it is not possible to create two identical copies of a PUF.

One of the important security requirements in IoT systems is authentication. A device/user should be able to verify that the data received from another device/user is indeed collected by the stated sensor. Therefore, authentication is the first step towards establishing a session after a secure boot of the IoT device. However, as mentioned previously, this authentication must be done in a secure and efficient way, without saving any secrets in the IoT device's memory. To address this issue, this paper presents a secure and light-weight mutual authentication protocol for IoT systems. The proposed protocol achieves the desired security and efficiency requirements using PUFs. The major contributions of this work are as follows: (i) Design of a mutual authentication protocol using PUFs, which has very low energy, memory, and communication overhead.

Moreover, the protocol is safe against physical attacks; (ii) A mechanism to establish a secure session key without any extra computational or communication overhead.

The rest of the paper is organized as follows. In Section II, we discuss the related work. Section III gives a brief introduction to PUFs. Section IV discusses the network model, threat model, requirements and assumptions for our IoT system. Section V presents the proposed PUF based mutual authentication protocols. Section VI presents a security analysis of the proposed protocols and Section VII discusses the protocol simulations. We present a formal verification of the proposed protocols in Section VIII. Finally, in Section IX we present the performance analysis of the proposed protocols and conclude the paper in Section X.

II. Literature Review

The different security challenges for designing IoT systems are outlined in [2]. The authors recommend the use of hardware security primitives such as PUFs for securing IoT systems. A one-time password (OTP) based authentication protocol for IoT devices is presented in [5]. However, multiple iterations and the requirement of computing the OTP using public/private keys increases the complexity and computational burden of the algorithm. In another work [6], the authors propose a group authentication protocol. However, collective authentication and key sharing among multiple nodes puts multiple nodes at risk even if one of the nodes is compromised. A certificate-based authentication protocol for distributed IoT systems is proposed in [7]. However, the cryptographic credentials are stored in the edge node, exposing the protocol to cloning attacks. Similarly, other works like [8], [9], [10] describe different mechanisms for authentication in IoT systems.

PUFs have been previously used in authentication protocols for wireless sensor networks (WSNs) and radio-frequency identification (RFID) systems [11], [12], [13], [14], [15]. However, these PUF based as well as non-PUF based authentication protocols mentioned previously rely on some secrets (e.g. public/private keys) being stored in the memory of an IoT device. Some of these works propose to store the secrets in a highly secure manner. However, even if the memory is highly secure the bits still need to be stored in a non-volatile digital memory which opens doors for attackers. Moreover, some of the PUF based authentication protocols require a trusted party to store a large number of challenge response pairs (CRPs) for each IoT device [11], [12]. Given the increasing number of IoT devices, these protocols are not scalable. Moreover, for PUF based authentication protocols, storing secrets locally contradicts with the philosophy of using PUFs.

The most relevant PUF based authentication protocol is presented in [16]. The authors of [16] propose an authentication protocol based on PUFs that uses the zero-knowledge proof of knowledge (ZKPK) of discrete logarithm. Their protocol does not reveal the CRP in any of its messages by using the zero-knowledge proofs. However, their protocol requires a user to input a password to the device each time it requires

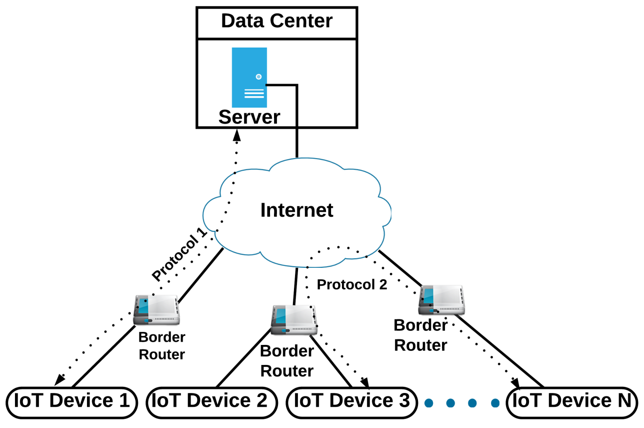


Fig. 1: Network model.

authentication, which reduces the effectiveness of this protocol in IoT systems. Moreover, their protocol can only be used to perform authentication between a user and a server and not for two devices authenticating each other. The use of the ZKPK of discrete logarithm also increases the complexity of the protocol.

III. Preliminary Background

Before presenting the proposed protocol, we first present a brief introduction to PUFs in this section.

A PUF may be regarded as a unique physical feature of a device, just like the biometric features of human beings such as fingerprints. A good definition of a PUF is given in [17] as "an expression of an inherent and unclonable instance-specific feature of a physical object". The most notable property of a PUF is that it cannot be reproduced using cryptographic primitives, rather, it requires a physical basis. Moreover, the term "physically unclonable" means that it can be shown through physical reasoning that it is extremely hard or even impossible to produce a physical clone of a PUF [18]. So the idea behind using a PUF in IoT systems is that just like human beings, every device will have a unique fingerprint in the form of a PUF and this fingerprint cannot be reproduced or cloned.

In another place, a PUF is defined as "A Physical Unclonable Function (PUF) is a function that maps a set of challenges to a set of responses based on an intractably complex physical system" [11]. Therefore, a PUF can be considered as a function, which takes a challenge in the form of a string of bits and produces a response in the form of a string of bits. We represent a PUF as a function P as follows

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

where R is the response of a PUF, while C is the challenge given to the PUF. A challenge C and its response R from a PUF is called a challenge response pair (CRP). PUFs have the following properties in terms of the response:

1) The PUF produces the same response with the same challenge with high probability even if the same challenge is used multiple times.

TABLE I: Notations

Notation	Description
ID_i	ID of the IoT device
$H(X)$	Hash of X
$\|\|$	Concatenation operator
$\{M\}_k$	Message M is encrypted using key k
C^i	Challenge for the i-th iteration
R^i	Response of the respective PUF for C^i

2) The same challenge will produce responses far apart with high probability if it is input to different PUFs.

IV. Network Model and Assumptions

A. Network Model

Figure 1 describes our network model. In this model we have IoT devices connected to the Internet through border router elements (e.g. based on 6LoWPAN). Each IoT device is equipped with a PUF. The IoT devices send the information (over the Internet) to a server in the data center.

B. Assumptions

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

a. An IoT device consists of an embedded system equipped with a PUF. Any physical tampering with the PUF such as an attempt to separate it from the embedded system will destroy the PUF.
b. The IoT device micro-controller and the PUF are considered to be a system on chip, and the communication between them is considered secure [19], [20].
c. The data center is considered the trusted party and has no limitation of resources. However, the IoT devices have limited resources.
d. Table I gives the set of notations used to describe the protocol.

C. Attack Model

IoT devices send their data to the data center server through the Internet. We assume that the IoT devices use the standard or compressed (e.g. 6LoWPAN) TCP/IP protocol suite to send data to the server [21]. The packets containing IoT data may follow different paths and pass through multiple communication links and network routers. We assume that the adversary may compromise one or more network entities and may even capture an IoT device. We assume the adversary can eavesdrop, inject packets, mimic other devices, initiate a session, and replay older messages.

The objective of the adversary is to gain authentication into the data center server or any of the IoT devices without being detected. By getting access into the data center server or an IoT device, the adversary wishes to launch various attacks with the intention of causing physical or economical damage. For example, if vehicles equipped with IoT devices are talking to the traffic light and the adversary manages to authenticate itself to the vehicles as the traffic light, it can cause large scale road accidents. The objective of this paper is to develop a mutual authentication protocol which is secure against different kinds of attacks including illegal user profiling, cloning attacks, man-in-the middle attacks, replay attacks, tampering attacks, and physical attacks.

D. Security Requirements

To achieve mutual authentication, we designed the proposed protocol according to the following security requirements:

1) Achieve mutual authentication between an IoT device and a server or between two IoT devices.
2) Establish a secure session key between the two entities before packet transmission.
3) Attacker cannot compromise the security of the scheme even if he/she captures an IoT device.

V. Proposed PUF based Mutual Authentication and Key Exchange Protocol

In this section we present the proposed mutual authentication protocol. The following two scenarios are considered for mutual authentication:

1) An IoT device wants to establish a connection with a server in the data center.
2) One IoT device wants to talk to another IoT device.

The server starts with a single CRP for each IoT device. This initial CRP can be obtained by the server using the Time-based One-time Password Algorithm (TOTP) mechanism [22]. When an IoT device is deployed in the field for the first time, an operator inputs a password into the device to exchange the initial CRP with the server using the TOTP approach. Once the initial CRP is exchanged with the server, the IoT device can function independently without the need for any operator. Thus, the server stores the identity ID_A, and the CRP (C^i, R^i) for each IoT device, while the IoT device does not store anything.

A. Protocol 1: IoT Device and Server Mutual Authentication

The proposed mutual authentication protocol for the case when an IoT device and a server want to communicate with each other is shown in Figure 2. The steps are as follows:

1) IoT device ID_A sends its ID, ID_A, and a randomly generated nonce, N_1, to the server.
2) The server tries to locate ID_A in its memory and if the search fails the authentication request is rejected. Otherwise, the server reads the CRP (C^i, R^i) stored in its memory for this device. The server then generates a random number R_{S_1} and uses R^i to form the encrypted message $M_A = \{ID_A, N_1, R_{S_1}\}_{R^i}$. The server then sends C^i, M_A, and the message authentication code (MAC) to the IoT device ID_A in message 2 in Figure 2. The MAC ensures data integrity and freshness. The first two parameters in the MAC function ensure data integrity while the last parameter, i.e., R_{S_1} serves as the freshness identifier for the source, which in this case is the server. The same approach is followed for data integrity, message freshness, and source identifier throughout the protocol.

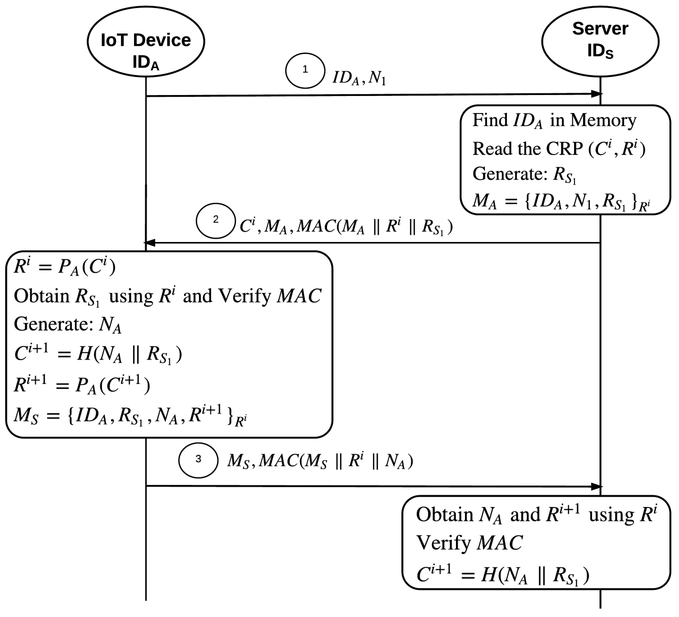

IoT Device ID_A
Server ID_S
1 ID_A, N_1
Find ID_A in Memory
Read the CRP (C^i, R^i)
Generate: R_{S_1}
$M_A = \{ID_A, N_1, R_{S_1}\}_{R^i}$
2 $C^i, M_A, MAC(M_A \| R^i \| R_{S_1})$
$R^i = P_A(C^i)$
Obtain R_{S_1} using R^i and Verify MAC
Generate: N_A
$C^{i+1} = H(N_A \| R_{S_1})$
$R^{i+1} = P_A(C^{i+1})$
$M_S = \{ID_A, R_{S_1}, N_A, R^{i+1}\}_{R^i}$
3 $M_S, MAC(M_S \| R^i \| N_A)$
Obtain N_A and R^{i+1} using R^i
Verify MAC
$C^{i+1} = H(N_A \| R_{S_1})$

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

3) IoT device ID_A generates R^i using its PUF and challenge C^i as given in (1). The device then obtains R_{S_1} using R^i and verifies the source, integrity, and freshness of the message using the MAC. If verification fails, IoT device ID_A terminates the authentication. Otherwise, it generates a random number N_A and computes the new response R^{i+1} by using the new challenge $H(N_A \| R_{S_1})$ and its PUF. This new CRP (C^{i+1}, R^{i+1}) will be used for future authentications.
   IoT device ID_A then sends an encrypted message $M_S = \{ID_A, R_{S_1}, N_A, R^{i+1}\}_{R^i}$ and the corresponding MAC to the server and then deletes all the temporary variables stored in its memory including $R_{S_1}, N_A, R^i, C^{i+1}$, and R^{i+1}.
4) The server computes N_A and R^{i+1} using R^i and verifies the MAC. If the verification fails, the server rejects the authentication. Otherwise, mutual authentication is considered complete and the two entities can now form a session.

We can use R_{S_1} and N_A to construct a session key as follows

$$H(R_{S_1}) \oplus H(N_A). \tag{2}$$

We note that the session key is constructed using the hash of the two random nonces R_{S_1} and N_A. Therefore, even if the session key is obtained by an adversary, he/she still cannot calculate R^i.

B. Protocol 2: Mutual Authentication for Two IoT Devices

In this section we describe the mutual authentication and key exchange protocol for the scenario when two IoT devices want to start a session and need to authenticate each other.

Consider a scenario where IoT device ID_A wants to establish a session with another IoT device ID_B. The mutual authentication protocol for this scenario is shown in Figure 3. The steps for the mutual authentication protocol are as follows:

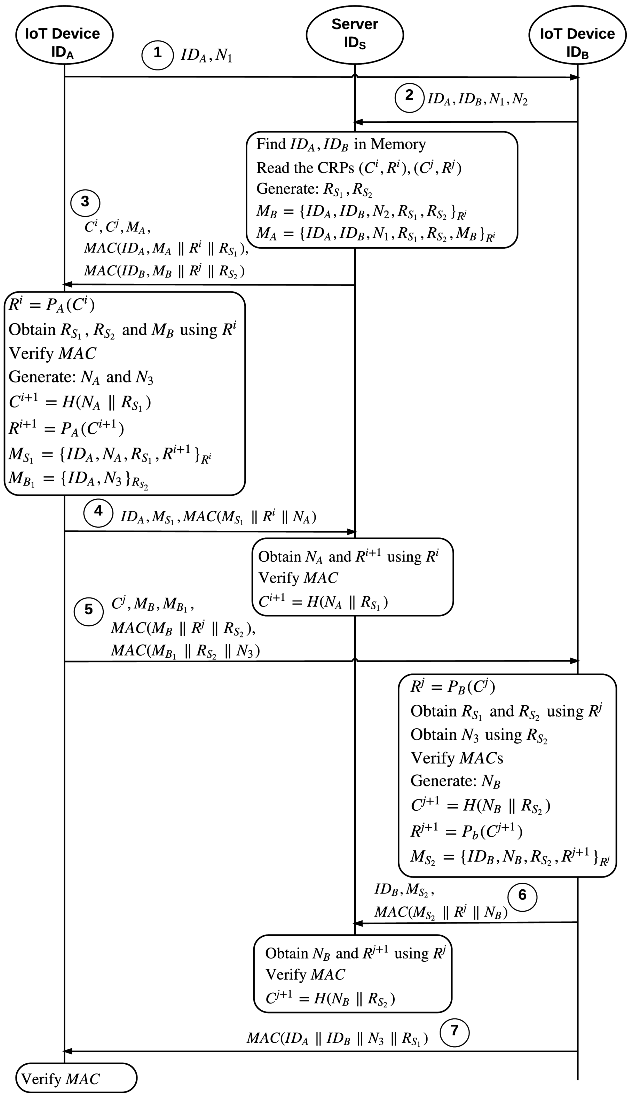

IoT Device ID_A
Server ID_S
IoT Device ID_B
1 ID_A, N_1
2 ID_A, ID_B, N_1, N_2
Find ID_A, ID_B in Memory
Read the CRPs $(C^i, R^i), (C^j, R^j)$
Generate: R_{S_1}, R_{S_2}
$M_B = \{ID_A, ID_B, N_2, R_{S_1}, R_{S_2}\}_{R^j}$
$M_A = \{ID_A, ID_B, N_1, R_{S_1}, R_{S_2}, M_B\}_{R^i}$
3 $C^i, C^j, M_A,$ $MAC(ID_A, M_A \| R^i \| R_{S_1}),$ $MAC(ID_B, M_B \| R^j \| R_{S_2})$
$R^i = P_A(C^i)$
Obtain R_{S_1}, R_{S_2} and M_B using R^i
Verify MAC
Generate: N_A and N_3
$C^{i+1} = H(N_A \| R_{S_1})$
$R^{i+1} = P_A(C^{i+1})$
$M_{S_1} = \{ID_A, N_A, R_{S_1}, R^{i+1}\}_{R^i}$
$M_{B_1} = \{ID_A, N_3\}_{R_{S_2}}$
4 $ID_A, M_{S_1}, MAC(M_{S_1} \| R^i \| N_A)$
Obtain N_A and R^{i+1} using R^i
Verify MAC
$C^{i+1} = H(N_A \| R_{S_1})$
5 $C^j, M_B, M_{B_1},$ $MAC(M_B \| R^j \| R_{S_2}),$ $MAC(M_{B_1} \| R_{S_2} \| N_3)$
$R^j = P_B(C^j)$
Obtain R_{S_1} and R_{S_2} using R^j
Obtain N_3 using R_{S_2}
Verify MACs
Generate: N_B
$C^{j+1} = H(N_B \| R_{S_2})$
$R^{j+1} = P_b(C^{j+1})$
$M_{S_2} = \{ID_B, N_B, R_{S_2}, R^{j+1}\}_{R^j}$
6 $ID_B, M_{S_2},$ $MAC(M_{S_2} \| R^j \| N_B)$
Obtain N_B and R^{j+1} using R^j
Verify MAC
$C^{j+1} = H(N_B \| R_{S_2})$
7 $MAC(ID_A \| ID_B \| N_3 \| R_{S_1})$
Verify MAC

Fig. 3: Mutual authentication of two IoT devices.

1) IoT device ID_A sends its ID, ID_A, and a random nonce, N_1, to IoT device ID_B.
2) IoT device ID_B sends the two ID's ID_A and ID_B with the corresponding nonces to the server as shown in message 2 in Figure 3.
3) The server searches its memory for ID_A and ID_B, and reads the respective CRPs (C^i, R^i), and (C^j, R^j). The server then generates two random numbers R_{S_1} and R_{S_2} and uses R^i and R^j to form the following encrypted messages

$$M_B = \{ID_A, ID_B, N_2, R_{S_1}, R_{S_2}\}_{R^j} \tag{3}$$
$$M_A = \{ID_A, ID_B, N_1, R_{S_1}, R_{S_2}, M_B\}_{R^i}. \tag{4}$$

   The server then sends the challenges C^i and C^j along with M_A and the two MACs to IoT device ID_A.

4) IoT device ID_A obtains R^i using the challenge C^i and its PUF. It then carries out the following tasks:
   (i) It decrypts the received message using R^i to obtain R_{S_1}, R_{S_2}, and M_B.
   (ii) It verifies the MAC. If the verification fails, IoT device ID_A does not respond and terminates the current authentication attempt.
   (iii) It generates a random number N_A and computes the new challenge C^{i+1} and new response R^{i+1} from its PUF.

   IoT device ID_A sends R^{i+1} in an encrypted message M_{S_1} along with a MAC to the server as shown in message 4 in Figure 3. IoT device ID_A sends another message $M_{B_1} = \{ID_A, N_3\}$ along with the corresponding MAC to IoT device ID_B. It also forwards the challenge C^j, message M_B and the corresponding MAC received from the server to IoT device ID_B. This is shown in message 5 in Figure 3.
5) The server obtains N_A and R^{i+1} using R^i. The server then verifies the MAC. If the verification fails, the server rejects the new CRP. Otherwise, the server updates the CRP for IoT device ID_A.
6) On receiving message 5, IoT device ID_B obtains R^j using C^j and its PUF. It then carries out the following tasks:
   (i) It decrypts M_B using R^j to obtain R_{S_1}, and R_{S_2}.
   (ii) It decrypts M_{B_1} using R_{S_2} to obtain the nonce N_3.
   (iii) It verifies the corresponding MACs. If the verification fails, IoT device ID_B does not respond and terminates the current authentication attempt.
   (iv) It generates a random number N_B and computes the new challenge C^{j+1} and new response R^{j+1} from its PUF.

   IoT device ID_B uses R^j to encrypt and send R^{j+1} in message M_{S_2} along with a MAC to the server as shown in message 6 in Figure 3. IoT device ID_B sends another message as an acknowledgment to IoT device ID_A, as shown in message 7 in Figure 3.
7) The server obtains N_B and R^{j+1} using R^j and verifies the corresponding MAC. If the verification fails, the server rejects the new CRP. Otherwise, the server updates the CRP for IoT device ID_B.
8) On receiving message 7, IoT device ID_A verifies the MAC. If verification fails the authentication is rejected. Otherwise, the authentication is considered complete and IoT device ID_A and IoT device ID_B have successfully authenticated each other.

Similar to Section V-A, the two IoT devices can now use R_{S_1} and R_{S_2} to establish a secret symmetric key. For example, $H(R_{S_1}) \oplus H(R_{S_2})$ can be used as the session key between IoT device ID_A and IoT device ID_B.

VI. Security Analysis

In this section we present a formal security analysis of the proposed mutual authentication protocols. The use of formal logical approaches for the security analysis of protocols is important to make sure an adversary cannot obtain or alter vital information. The work of Burrows, Abadi, and Needham [23], called the BAN logic, has been widely used for the security analysis of authentication and key distribution protocols. However, Mao and Boyd [24] showed several weaknesses in the BAN logic and put forward an improved extension of the BAN logic. We analyze our mutual authentication protocols using the Mao and Boyd logic and show that the protocols are secure against different types of attacks. For ease of notation, we represent IoT device ID_A, IoT device ID_B, and the server by A, B, and S respectively.

A. Security Analysis For Protocol 1

In this section we present a security analysis of the mutual authentication protocol for an IoT device and a server. To establish the security claims of the proposed protocol, we show that the secrets R_{S_1}, N_A, and R^{i+1} are good shared secrets between IoT device ID_A and the server (i.e. these secrets cannot be obtained or altered by an adversary). To apply the Mao and Boyd logic, the first step is to idealize the messages of a given protocol. The details on protocol message idealization are given in the Appendix. We start the formal proof by presenting the idealized versions of the message exchanged by protocol 1 as follows:

1) $A \rightarrow S : A, N_1$.
2) $S \rightarrow A : \{A, N_1 \mathbf{R} R_{S_1}\}_{R^i}$.
3) $A \rightarrow S : \{A, R_{S_1} \mathbf{R} N_A \mathbf{R} R^{i+1}\}_{R^i}$.

Note that our protocol uses MACs for effective data integrity and origin verification which is a pre-requisite for the Mao and Boyd logic. To establish the security of shared secrets, the Mao and Boyd logic uses a set of inference rules. The set of inference rules used in our analysis can be found in the Appendix. The initial beliefs/assumptions for Protocol 1 are given below:

1) $A \mid\equiv A \overset{R^i}{\leftrightarrow} S$ and $S \mid\equiv A \overset{R^i}{\leftrightarrow} S$: S saves a CRP for each IoT device in its memory while A can generate R^i by using the respective challenge.
2) $A \mid\equiv S^c \triangleleft \parallel N_A$ and $S \mid\equiv A \mid\equiv \{S\}^c \triangleleft \parallel N_A$: N_A is generated by A.
3) $A \overset{R^i}{\mid\sim} N_A$: Message 3 in the idealized protocol.
4) $A \mid\equiv \#(N_A)$ and $A \mid\equiv \#(N_1)$: A generates a new N_A and N_1 each time.
5) $S \mid\equiv \#(R_{S_1}))$: S generates a new R_{S_1} each time.
6) $A \mid\equiv sup(S)$: S is the super principal with-respect-to R_{S_1}.
7) $S \mid\equiv sup(A)$: A is the super principal with-respect-to N_A and R^{i+1}.
8) $A \overset{R^i}{\triangleleft} N_1 \mathbf{R} R_{S_1}$: Message 2 in the idealized protocol.
9) $S \overset{R^i}{\triangleleft} R_{S_1} \mathbf{R} N_A$ and $S \overset{R^i}{\triangleleft} R_{S_1} \mathbf{R} R^{i+1}$: Message 3 in the idealized protocol.
10) $A \mid\equiv S \mid\equiv \{A\}^c \triangleleft \parallel R_{S_1}$ and $S \mid\equiv A^c \triangleleft \parallel R_{S_1}$: S generates a new R_{S_1} each time.
11) $A \mid\equiv \{S\}^c \triangleleft \parallel R^{i+1}$ and $S \mid\equiv A \mid\equiv \{S\}^c \triangleleft \parallel R^{i+1}$ and $A \mid\equiv \#(R^{i+1})$: A computes a new response each time using

its PUF.

12) $S \overset{R^i}{\mid\!\sim} R_{S_1}$: Message 2 in the idealized protocol.
13) $A \overset{R^i}{\mid\!\sim} R^{i+1}$: Message 3 in the idealized protocol.

The tableau of Figure 4(b), shows the proof of the fact that A believes N_A is a good shared secret between A and S. To establish this fact, we write the statement to prove i.e., $A \models A \overset{N_A}{\leftrightarrow} S$ at the bottom. Next we apply the good-key rule from (26). This rule states that N_A is a good secret between A and S, if we can show that A is certain that no one else except A and S has seen N_A ($A \models \{A, S\}^c \triangleleft \parallel N_A$) and that N_A is a fresh nonce ($A \models \#(N_A)$). The confidentiality rule from (23) can be used to prove $A \models \{A, S\}^c \triangleleft \parallel N_A$, which requires us to show that A and S share a good secret R^i ($A \models A \overset{R^i}{\leftrightarrow} S$), and A sent N_A to S encrypting it with R^i ($A \overset{R^i}{\mid\!\sim} N_A$). We observe that these statements as well as $A \models \#(N_A)$ can be found in the set of initial beliefs. Thus, we can infer the truth of our initial statement i.e., $A \models A \overset{N_A}{\leftrightarrow} S$. Similarly, Figure 4(a) establishes the fact that R_{S_1} is a good shared secret between A and S. Similar analysis for N_A and R_{S_1} on the site of principal S is shown in the tableaux of Figures 4(c) and 4(d) respectively. We can conduct a similar analysis to prove the secrecy of R^{i+1} as given in the tableaux of Figures 4(e) and 4(f), respectively.

Without the knowledge of R^i, R^{i+1}, R_{S_1}, and N_A it is not possible for an adversary to construct valid data. It does not matter what kind of attack the adversary uses, the adversary cannot obtain these secrets as shown in the formal proofs. Moreover, even if the attacker captures the IoT device, he/she cannot obtain a valid CRP or any other secret because the device does not store any secrets and any attempt to remove the PUF from the IoT device destroys the PUF and makes it useless. Therefore, the proposed protocol is considered safe against all the major security attacks such as the spoofing attacks, cloning attacks, interleaving attacks, man-in-the-middle attacks, eavesdropping, and replay attacks.

B. Security Analysis of Protocol 2

In this section we present the security analysis for the proposed mutual authentication protocol for two IoT devices. Following the same approach as the previous section, the idealized versions of the messages exchanged during protocol 2 are given as follows:

1) $A \rightarrow B : A, N_1$.
2) $B \rightarrow S : A, B, N_1 | N_2$.
3) $S \rightarrow A : \{A, B, N_1 \mathbf{R} R_{S_1} \mathbf{R} R_{S_2}, \{A, B, N_2 \mathbf{R} R_{S_1} \mathbf{R} R_{S_2}\}_{R^j}\}_{R^i}$.
4) $A \rightarrow S : A, \{A, N_A \mathbf{R} R_{S_1} \mathbf{R} R^{i+1}\}_{R^i}$.
   $A \rightarrow B : \{A, B, N_2 \mathbf{R} R_{S_1} \mathbf{R} R_{S_2}\}_{R^j}, \{A, N_3\}_{R_{S_2}}$.
5) $B \rightarrow S : \{B, N_B \mathbf{R} R_{S_2} \mathbf{R} R^{j+1}\}_{R^j}$.

The tableaux created using the Mao and Boyd logic for the security analysis of the secrets R_{S_1} and R_{S_2} on the site of principals A and B are shown in Figure 5. Again, an adversary cannot compromise the proposed protocol without the knowledge of R^i, R^j, R^{i+1}, R^{j+1}, R_{S_1}, R_{S_2}, N_A, and N_B. This shows that the proposed protocol is resilient against various kinds of attacks including replay attacks, tampering attacks, cloning attacks, interleaving attacks, eavesdropping, and man-in-the middle attacks etc.

C. Protection Against Cloning

A large number of IoT devices are deployed out in the open, making them vulnerable to cloning attacks. However, each PUF has a unique output and it is not possible to recreate a PUF [11], [12]. The proposed protocol exploits this unique feature of a PUF to prevent it from cloning attacks. In the proposed protocol each IoT device is equipped with its own PUF and as it is not possible to clone a PUF, our proposed protocol is secure against cloning attacks.

D. Protection against Physical Attacks

One of the security requirements for IoT devices outlined in Section IV-D is that an IoT device should not expose any secrets even if it is captured by an attacker. As described in Section II, most of the authentication protocols proposed in existing literature require the device to store one or more secrets. This requirement greatly reduces the effectiveness of these protocols as it makes them susceptible to physical attacks. The proposed mutual authentication protocol has two features which safeguard it against physical attacks. First, the devices do not need to store any secrets. Second, the communication between the IoT device's micro-controller and its PUF is considered to be secure as they are on the same chip [19]. Thus, even if an attacker has physical access to an IoT device, he/she cannot obtain any secrets. This shows that our proposed mutual authentication protocol is secure against physical attacks.

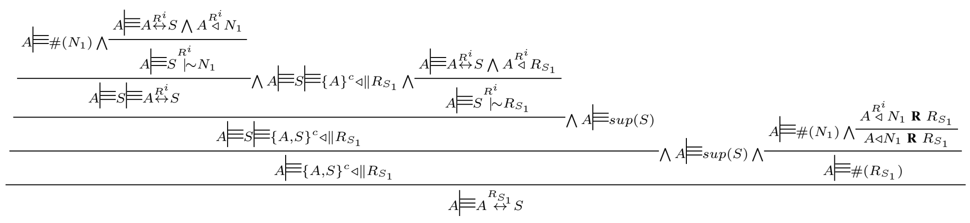
VII. Simulations

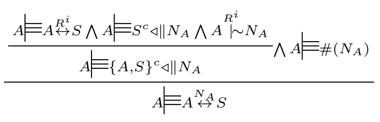
The security properties of the proposed protocols were verified using rigorous simulations and experimentation using the security verification tool ProVerif (PV) [25]. PV is considered to be sound (correct) in proving security properties [25] and can thus be used to prove reachability properties, observational equivalence, and correspondence assertions. We model the IoT devices and the server as separate processes and simulate arbitrarily many sessions of the protocol between the protocol entities. The simulation scripts for the proposed protocols can be downloaded from [26].

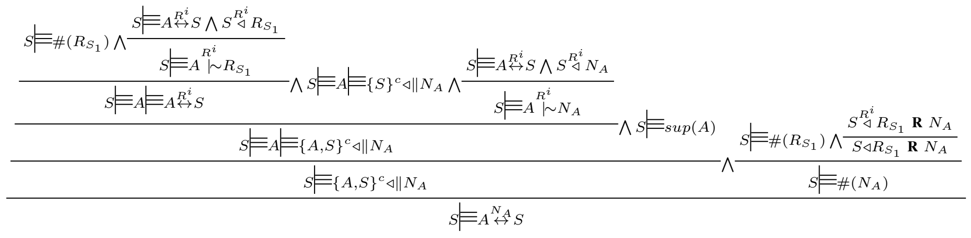
A. Protocol 1 Simulation

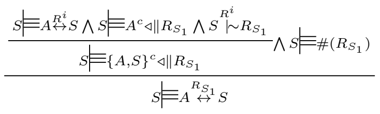
The primary objective of protocol 1 is to ensure mutual authentication of the IoT device ID_A and the server. Therefore, when the protocol reaches the end and the IoT device ID_A believes it has completed the protocol with the server, then IoT device ID_A has indeed engaged in a session with the server. Similarly, when the server reaches the end of the protocol, it has indeed engaged in a session with the IoT device ID_A. To prove the desired security properties of the proposed protocol we define the following events in PV:

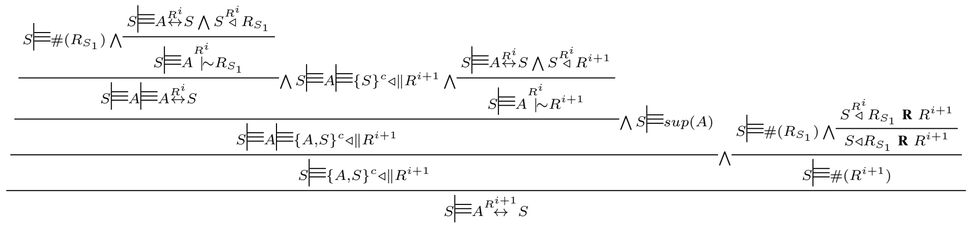
- **event** $beginAfull(ID_A, ID_S, N_A, N_B)$ is used by the server to record the belief that the IoT device ID_A with


(a) Proof of "**A** believes R_{S_1} is a good shared key of **A** and **S**".


(b) Proof of "**A** believes N_A is a good shared key of **A** and **S**".


(c) Proof of "**S** believes N_A is a good shared key of **A** and **S**".


(d) Proof of "**S** believes R_{S_1} is a good shared key of **A** and **S**".


(e) Proof of "**S** believes R^{i+1} is a good shared key of **A** and **S**".

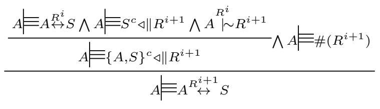

(f) Proof of "**A** believes R^{i+1} is a good shared key of **A** and **S**".

Fig. 4: Security Proofs for Protocol 1

shared secret N_A has commenced a run of the protocol with it (where N_B is the shared secret of the server).

- **event** $endAfull(ID_A, ID_S, N_A, N_B)$ which means the IoT device ID_A has successfully completed the protocol with the server, and the two parties agree to the shared secrets N_A and N_B.
- **event** $beginBfull(ID_A, ID_S, N_A, N_B)$ which denotes IoT device ID_A's intention to launch the protocol with the server with the given protocol parameters.
- **event** $endBfull(ID_A, ID_S, N_A, N_B)$ which means the server has successfully completed the protocol with IoT device ID_A with the given protocol parameters.

The protocol is expected to satisfy the following properties:

1) **Authentication of the server to IoT device ID_A:** IoT device ID_A intends to to share its data only with the server. Thus, if it believes it has executed the protocol with the server, then IoT device ID_A indeed completed the protocol with the server. Thus, authentication of the server to IoT device ID_A holds. PV uses correspondence assertions to prove authentication. We define the following correspondence assertions in PV to prove this property:

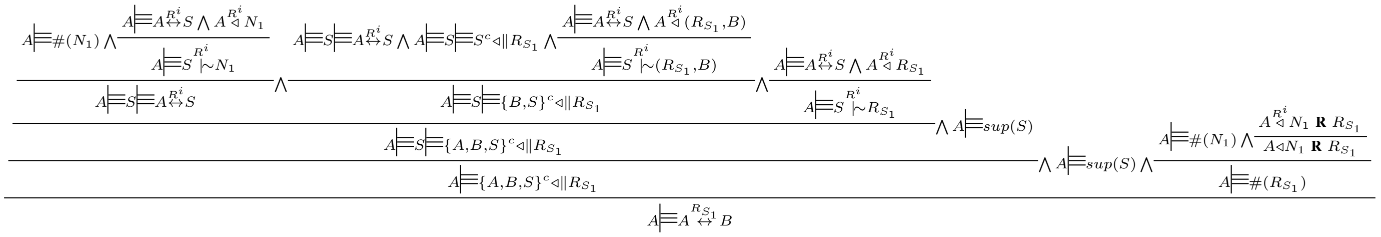
$$inj\text{–}event(endBfull(\cdots)) ==> \\ inj\text{–}event(beginBfull(\cdots)). \quad (5)$$

Note that in PV the statement $E_A ==> E_B$ is used to check the fact "*if an event E_A has been executed, then it has been preceded by the execution of event E_B previously.*"

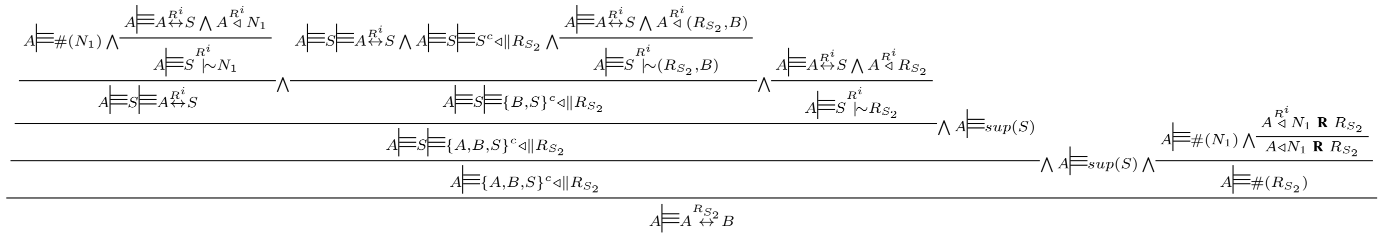
2) **Authentication of IoT device ID_A to the server:** The server can establish a session with any of its client IoT devices. Thus, if it believes it has completed the protocol with IoT device ID_A, authentication from IoT device ID_A to the server should hold. The following correspondence assertion is used to prove this property in PV:

$$inj - event(endAfull(\cdots)) ==> \\ inj - event(beginAfull(\cdots)). \quad (6)$$

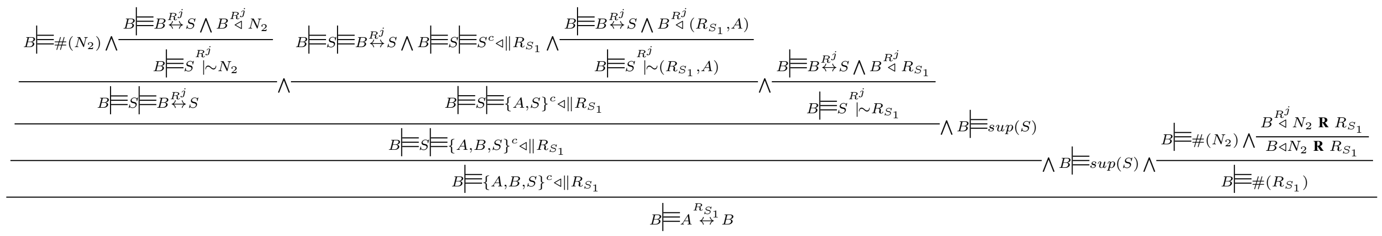
3) **Secrecy for IoT device ID_A and the server:** If IoT device ID_A completes a run of the protocol with the server, then the secrets N_A and R_{S_1} that IoT device ID_A has are good secrets. In PV we can check which terms are available to an attacker by querying an attacker. We prove the secrecy of a term by using it as a session key to encrypt an arbitrary term M and then checking whether an attacker can successfully obtain M. The syntactic secrecy of N_A, R_{S_1}, and R^{i+1} is established using this



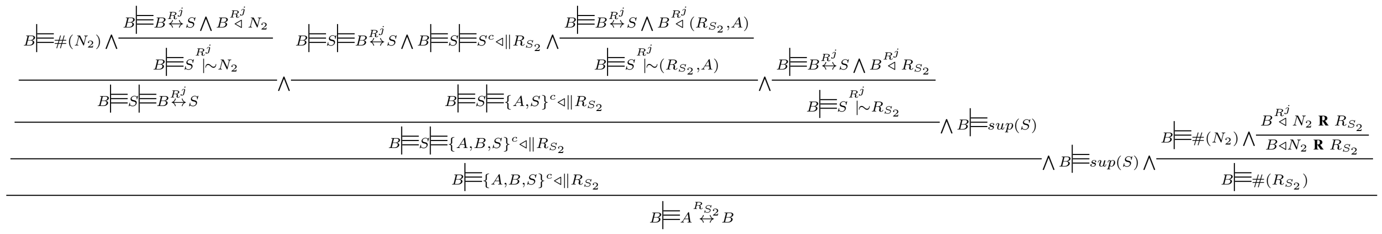
(a) Proof of "**A** believes R_{S_1} is a good shared key of **A** and **B**".



(b) Proof of "**A** believes R_{S_2} is a good shared key of **A** and **B**".



(c) Proof of "**B** believes R_{S_1} is a good shared key of **A** and **B**".



(d) Proof of "**B** believes R_{S_2} is a good shared key of **A** and **B**".

Fig. 5: Security Proofs for Protocol 2

approach in PV as follows:

$$query \quad attacker(ANa); attacker(BNa); \tag{7}$$

$$attacker(ANb); attacker(BNb); \tag{8}$$

$$attacker(AR\_new); attacker(SR\_new) \tag{9}$$

where, ANa, ANb, and AR_{new} are used to prove the secrecy of N_A, R_{S_1}, and R^{i+1}, respectively, on the site of principal IoT device ID_A, i.e., we encrypt these arbitrary terms using the respective secrets and then check whether an attacker can obtain these terms. For example, the arbitrary term ANa is encrypted using N_A and is sent by IoT device ID_A on an open channel. Later we check whether ANa is available to an attacker. Similarly, the secrecy of N_A, R_{S_1}, and R^{i+1} on the site principal of the server is proved using BNa, BNb, and BR_new, respectively. The PV simulations show that the proposed protocol is secure against any definite or possible attack.

B. Protocol 2 Simulation

The primary objective of protocol 2 is the mutual authentication of principals IoT device ID_A and IoT device ID_B using the server as the trusted party. We define the following events in PV to prove the desired security properties:

- **event** $beginAfull(ID_A, ID_B, N_A, N_B)$ which is used by IoT device ID_B to record IoT device ID_A's intention to commence the protocol with IoT device ID_B using N_A and N_B as the shared secrets.

- **event** $endAfull(ID_A, ID_B, N_A, N_B)$ records IoT device ID_A's belief that it has successfully completed the protocol with IoT device ID_B with N_A and N_B being the shared secrets .
- **event** $beginBfull(ID_A, ID_B, N_A, N_B)$, which represents the IoT device ID_A's intention to initiate the protocol with the IoT device ID_B using the given protocol parameters.
- **event** $endBfull(ID_A, ID_B, N_A, N_B)$, which means IoT device ID_B has successfully completed the protocol with the IoT device ID_A with the given protocol parameters.

The desired security properties proved for protocol 2 are as follows:

1) **Authentication of IoT device ID_B to IoT device ID_A:** IoT device ID_A intends to engage in a session with IoT device ID_B. Hence, if it completes the protocol, it has indeed done so with IoT device ID_B and the two parties agree on the set of given protocol parameters. We prove this property in PV as follows:
$$\begin{aligned} &inj-event(endBfull(\cdots)) ==> \\ &\quad inj-event(beginBfull(\cdots)). \end{aligned} \tag{10}$$
2) **Authentication of IoT device ID_A to IoT device ID_B:** If IoT device ID_B believes it has reached the end of the protocol with IoT device ID_A, it has indeed done so with IoT device ID_A. We use the following correspondence assertion using the following:
$$\begin{aligned} &inj-event(endAfull(\cdots)) ==> \\ &\quad inj-event(beginAfull(\cdots)). \end{aligned} \tag{11}$$
3) **Authentication of IoT device ID_A and IoT device ID_B to the server:** Each time an authentication protocol is run, the server needs to update its CRP for the respective IoT device. In protocol 2, the server needs to update the CRPs for both IoT device ID_A and IoT device ID_B. For this purpose it is also desirable that mutual authentication between the respective IoT devices and the server is also satisfied. The authentication properties between IoT device ID_A and the server, and IoT device ID_B and the server are proved in a similar fashion as protocol 1 and can be found in [26].
4) **Secrecy for IoT device ID_A:** If IoT device ID_A completes the protocol successfully then the secrets R_{S_1}, R_{S_2}, N_A, and R^{i+1} can be considered good secrets at the site principal of IoT device ID_A. We establish the syntactic security of these secrets using the following queries in PV:
$$query\ \ attacker(ARs1); attacker(ARs2); \tag{12}$$
$$attacker(ANa); attacker(AR\_new); \tag{13}$$
where $ARs1$, $ARs2$, ANa, and AR_new are arbitrary terms used to check the secrecy of R_{S_1}, R_{S_2}, N_A, and R^{i+1}, respectively, at the site principal IoT device ID_A.
5) **Secrecy for IoT device ID_B:** If IoT device ID_B reaches the end of the protocol then the secrets R_{S_1}, R_{S_2}, N_B, and R^{j+1} are good secrets at the site principal of IoT device ID_B. The PV queries for this property are as follows:
$$query\ \ attacker(BRs1); attacker(BRs2); \tag{14}$$
$$attacker(BNa); attacker(BR\_new); \tag{15}$$
where the secrecy of $Rs1$, $Rs2$, N_A, and R^{j+1} is checked using the arbitrary terms $BRs1$, $BRs2$, BNa, and BR_new, respectively.
6) **Secrecy for the server:** If the server believes it has successfully completed the protocol with IoT device ID_A and IoT device ID_B, then R_{S_1}, R_{S_2}, N_A, N_B, R^{i+1}, and R^{j+1} are good secrets at the site principal of the server. We use the following PV queries to prove this property:
$$query\ \ attacker(SRs1); attacker(SRs2); \tag{16}$$
$$attacker(SNa); attacker(SNb); \tag{17}$$
$$attacker(SR\_newA); attacker(SR\_newB); \tag{18}$$
where $SRs1$, $SRs2$, SNa, SNb, SR_newA, and SR_newB are used to prove the secrecy of R_{S_1}, R_{S_2}, N_A, N_B, R^{i+1}, and R^{j+1}, respectively at the site principal of the server.

Running the simulation script for protocol 2 in [26] in PV shows that the proposed protocol satisfies all the desired security properties.

VIII. Verification

In this section we provide a formal verification for the correctness of the proposed protocols. A protocol is considered correct if it possesses the following properties [27]:

1) **Completeness**: All valid inputs are accepted by the protocol.
2) **Deadlock freeness**: The protocol does not enter a state where it can stay indefinitely.
3) **Livelock or tempo-blocking freeness**: There are no infinite loops in the protocol.
4) **Termination**: The protocol always reaches a well-defined final state starting from the initial state.
5) **Absence of non-executable interactions**: The protocol only contains transmission, reception, and interaction paths realizable under normal operating conditions.

We use a finite state machine (FSM) and reachability analysis technique proposed in [27] to verify the correctness of the proposed protocols. Note that this section provides a proof that the protocol is correct with respect to its specifications, and does not provide any proof of security. The security analysis of the protocol is given in Section VI.

The first step to prove the correctness of the protocol using [27] is to represent each entity of a protocol as a directed graph. The directed graphs for the two entities in Protocol 1 are shown in Figure 6, where g_A and g_{server} represent the directed graphs for IoT device ID_A and the server, respectively. Each of these directed graphs can be considered as an FSM for that

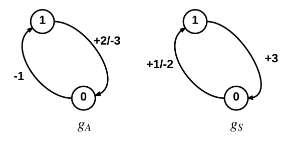


Fig. 6: Directed graphs for Protocol 1.

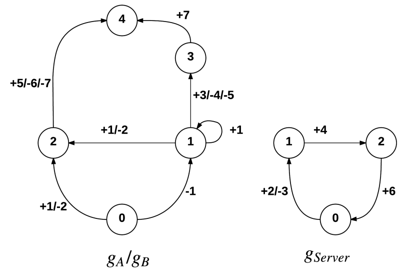


Fig. 7: Directed graphs for Protocol 2.

entity. Similarly, the directed graphs for the three entities of Protocol 2 are shown in Figure 7. The state of the protocol machine is represented by numbers in the nodes, while -n and +n represents the transmission and reception of a message, respectively. Moreover, transitions or events are represented by the integer labels on the arcs joining two states i.e., +m/-n represents the reception of a message m followed by the transmission of message n. For example, the interaction paths for g_A and g_S corresponding to one run of Protocol 1 are given below:

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

where "[]" denotes a state in Figure 6. The interaction path for IoT device ID_A represents the following sequence of activities: IoT device ID_A starts in state 0, sends message 1 to the server and enters state 1 of its FSM. IoT device ID_A then sends message 2 and receives message 3, entering state 0 again. We can similarly interpret the interaction patch for the server. Moreover, S_0 is considered the final state for both IoT device ID_A and the server in case of Protocol 1. However, S_4 is considered the final state for IoT devices ID_A and ID_B, while S_0 is the final state for the server in the case of Protocol 2.

We use the reachability analysis technique proposed in [27], [28] to prove that our protocol possesses the properties (1) - (5). Figures 8 and 9 show the result of the reachability analysis for the proposed protocols. In this technique a matrix is used to represent the state of the overall system. This matrix is constructed using the states of all the entities in the protocol. Equations (19) and (20) show the state matrices for protocol 1 and 2, respectively.

$$\begin{bmatrix} \text{A STATE} & \text{A} \rightarrow \text{Server CHANNEL} \\ \text{Server} \rightarrow \text{A CHANNEL} & \text{Server STATE} \end{bmatrix} \tag{19}$$

$$\begin{tikzcd}
 & & & & \begin{bmatrix} S_0 & E \\ E & S_0 \end{bmatrix}^{\textbf{SS0}} \arrow[d, "A^{-1}"] \\
\textbf{SS0} & \begin{bmatrix} S_0 & 3 \\ E & S_1 \end{bmatrix}^{\textbf{SS4}} \arrow[l, "S^{+3}"'] & & \begin{bmatrix} S_1 & E \\ 2 & S_1 \end{bmatrix}^{\textbf{SS2}} \arrow[ll, "A^{+2}"'] & \begin{bmatrix} S_1 & 1 \\ E & S_0 \end{bmatrix}^{\textbf{SS1}} \arrow[l, "S^{+1}"']
\end{tikzcd}$$

Fig. 8: Reachability analysis for Protocol 1.

$$\begin{bmatrix} \text{A STATE} & \text{A} \rightarrow B \text{ CHANNEL} & \text{A} \rightarrow \text{Server CHANNEL} \\ \text{B} \rightarrow \text{A CHANNEL} & \text{B STATE} & \text{B} \rightarrow \text{Server CHANNEL} \\ \text{Server} \rightarrow \text{A CHANNEL} & \text{Server} \rightarrow \text{B CHANNEL} & \text{Server STATE} \end{bmatrix} . \tag{20}$$

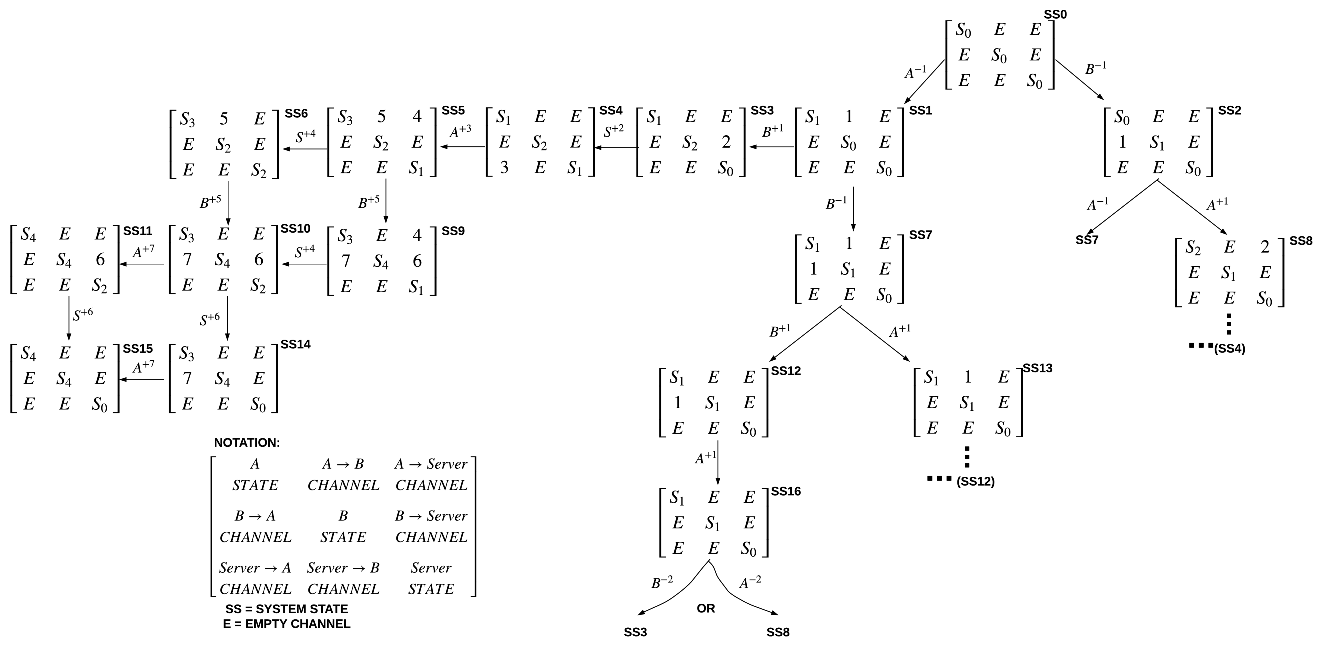
Each element in the above matrices represents either the current state of the FSM of an entity or a message sent by an entity. In (19), the element in row 1 and column 1 represents the state of entity A while the element in row 1 and column 2 represents the message sent by A to the server. Similarly, the element in row 2 and column 1 represents the message sent by the server to A while the element in row 2 and column 2 represents the current state of the server entity. For example, at the start of a protocol all the protocol entities are in their start states i.e, S_0, and the channels are empty, as shown in Figures 6 and 7. Let us consider protocol 1, in which IoT device ID_A sends message 1 to the server. This will cause a transition in the state of the FSM of IoT device ID_A from S_0 to S_1, as shown in Figure 6. The overall system state matrix is then given as follows:

$$\begin{bmatrix} S_1 & 1 \\ E & S_0 \end{bmatrix} . \tag{21}$$

The element at row 1 and column 1 of (21) shows that IoT device ID_A is in state 1, while the element at row 1 and column 2 of the matrix shows that IoT device ID_A has sent message 1 to the server. Moreover, this matrix also shows that the server is in state 0 (row 2 and column 2) and has not sent any message to IoT device ID_A (row 2 column 1). We represent the overall system state by SSi, while the state of the constituent subsystems (entities) is represented by S_i.

We always start the reachability analysis from the initial state SS_0. This is the state in which all the channels are empty, i.e., E, and all the protocol entities are in their initial states, i.e, S_0. Moreover, X^{+i} and X^{-i} represent the reception and transmission of message i by entitiy X, respectively. The reachability analysis for protocols 1 and 2 is shown in Figures 8 and 9, respectively.

Figure 8 shows a sequence of transitions for the overall system state for protocol 1. When IoT device ID_A sends message 1, the overall system state transitions from the initial state $SS0$ to $SS1$, followed by subsequent transitions. Figure 8 shows that the protocol accepts all valid messages implying the completeness property. We defined a potential deadlock



$$\left[\begin{array}{ccc} A & A \to B & A \to Server \\ STATE & CHANNEL & CHANNEL \\ B \to A & B & B \to Server \\ CHANNEL & STATE & CHANNEL \\ Server \to A & Server \to B & Server \\ CHANNEL & CHANNEL & STATE \end{array}\right]$$

Fig. 9: Reachability analysis for Protocol 2.

state as an overall system state in which all the channels are empty and it is not an initial or final state. We observe that Figure 8 does not have any potential deadlock states, implying deadlock freeness. We also observe that there are no loops among the overall system states implying livelock or tempo-blocking freeness. Moreover, Figure 9 covers all the possible transmissions, interaction paths, receptions and states and we see that following any of the interaction paths we always end up in the state $SS0$. This shows that the protocol possesses the termination property and does not possess any non-executable interactions. Thus, the proposed protocol is considered to be correct.

The reachability analysis for Protocol 2 shows that $SS18$ is the final state for Protocol 2, i.e., the channels are empty and all the subsystems are in their respective final states. We observe that Protocol 2 always starts from the initial state $SS0$ and terminates at the final state $SS18$, implying the termination property. The protocol has one potential deadlock state i.e., $SS16$. However, $SS16$ has outgoing transitions involving a decision based on the random numbers N_1 and N_2 in the protocol. Note that $SS16$ is observed by the system when both IoT devices ID_A and ID_B attempt to initiate authentication concurrently. We assume that the protocol uses the random nonces N_1 and N_2 to break the tie. For example, if $N_1 > N_2$ then IoT device ID_A gets to act as the initiator of the authentication and IoT device ID_B sends message 2 to the server and vice versa. This leads to IoT device ID_B transitioning to state 2 as shown by the overall system state $SS3$. This shows that the proposed protocol possesses the deadlock freeness property. We observe that there are no loops in Figure 9. Therefore, the proposed protocol can be considered free from livelocks or tempo-blocking. We observe that the reachability analysis of Protocol 2 covers all the interaction paths, states, transmissions, and receptions. This shows that the proposed protocol does not contain any non-executable interactions. This proves that the proposed protocol is correct.

TABLE II: Computational Complexity

Task	IoT Device	Server
Protocol 1	$1N_H + 2N_{MAC} + 2N_{ENC}$	$1N_H + 2N_{MAC} + 2N_{ENC}$
[16]	$2N_H + 2N_{exp} + N_\times$	$1N_H + 3N_{exp}$
Protocol 2	$1N_H + 4N_{MAC} + 3N_{ENC}$	$2N_H + 4N_{MAC} + 4N_{ENC}$

IX. Performance Analysis

In this section we present a performance analysis of the proposed protocols and compare it with the most relevant protocol in literature, proposed by Frikken et al. [16].

A. Computational Complexity

Table II shows the number of hash (N_H), MAC (N_{MAC}), encryption/decryption (N_{ENC}), modular exponentiation (N_{exp}), and modular multiplication ($N_\times$) operations required by the proposed mutual authentication protocols and [16]. We can directly obtain these values by counting the occurrence of the respective operations in Figures 2 and 3.

Let us assume the use of message authentication codes based on universal hashing (UMACs) [29], that have a worst case running time of $O(n)$ [30], [31], where n is the message size. Moreover, let us assume the use of block ciphers. Then the encryption and decryption operations can be considered $O(n)$. Thus, the proposed protocols have a complexity of $O(n)$ for the IoT devices as well as the server. However, the technique in [16] requires hash operations and modular

TABLE III: Parameter Lengths

Parameter	Size (bits)
ID	8 [35]
N_1, N_2	48 [36]
N_A, R_{S_1}, R_{S_2}	128 [33]
C, R	128 [33]
MAC	32/64/96/128 [29]

exponentiation. Therefore, the complexity of their protocol is $O(n + M(l)k)$ for both the IoT devices as well as the server, where $M(l)$ represents the complexity of a general modular multiplication with l bit operands, and k is the the exponent. Moreover, $M(l)$ is generally quadratic in l [32]. Thus the computational complexity of the proposed mutual authentication protocols is lower.

B. Communication Overhead

We observe that message 2 is the longest message in Protocol 1, while message 3 is the longest message in Protocol 2 as shown in Figures 2 and 3. To calculate the length of these messages we use the parameter sizes given in Table III. We assume the use of block ciphers such as CLEFIA with 128-bit keys [33]. Note that the parameters in Table III which are used to encrypt data in the proposed protocols are 128-bit long. Moreover, we assume the use of UMAC as the MAC, which provides the flexibility in meeting security as well as performance needs [29]. UMAC offers MACs of varying lengths as given in Table III.

For the purposes of calculating the communication overhead, we use a MAC length of 32 bits. Thus, the length of message 2 in Protocol 1 is 42 bytes, while the length of message 3 in Protocol 2 is 120 bytes[1]. Moreover, the longest message in [16] is approximately 68 bytes which is much larger compared to Protocol 1.

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 authentication and deleted afterward. Moreover, only one CRP pair (C^i, R^i), and the respective ID_i are stored for each IoT device in the server. In contrast, most of the protocols in existing literature impose one the following requirements:

1) The IoT devices store secret information.
2) The server stores a large number of CRPs for each IoT device. One of the CRPs is used each time the server authenticates an IoT device. Given the large number of IoT devices, this approach is not scalable.

The proposed mutual authentication protocols do not impose these requirements and have very low storage requirements.

[1]Note that 6LoWPAN has a maximum transmission unit (MTU) size of 127 bytes [34]. Although the maximum length of messages in the proposed protocols is less than 127 bytes, the addition of IP and other headers may result in fragmentation of the packets at the 6LoWPAN adaptation layer [34].

X. Conclusions

This paper presented mutual authentication protocols for two different scenarios in IoT systems: (i) for cases when an IoT device and a server want to communicate with each other, and (ii) for the case when two IoT devices want to communicate with each other. The proposed protocols are based on a challenge-response mechanism using PUFs and have the unique security feature of not saving any secrets in the IoT devices, while keeping the storage requirements at the server to the minimum. Moreover, a session key can also be established using the proposed protocols. Performance analysis of the proposed protocols shows that they have very low computation, storage, and communication overhead. However, to use these protocols for applications with strict timing requirements such as vehicular networks, it is desirable to further reduce the latency of authentication by reducing the number of messages exchanged between the entities.

References

[1] *The Internet of Things Reference Model,* CISCO, 2014, http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf.

[2] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of IoT Systems: Design Challenges and Opportunities,"*Proceedings of IEEE/ACM ICCAD,* pp. 417-423, San Jose, CA, November 2014.

[3] *Security in the Internet of Things*, Wind River, January 2015, http://www.windriver.com/whitepapers/security-in-the-internet-of-things/wr_security-in-the-internet-of-things.pdf.

[4] G. Woo, P. Kheradpour, D. Shen and D. Katabi, "Gartner Says the Internet of Things will Transform the Data Center," Gartner, March 2014.

[5] Shivraj, V.L., Rajan, M.A., Singh, M., and Balamuralidhar, P., "One time password authentication scheme based on elliptic curves for Internet of Things (IoT)," *Proceedings of National Symposium on Information Technology: Towards New Smart World (NSITNSW),* pp.1-6, Riyadh, KSA, February 2015.

[6] P. N. Mahalle, N. R. Prasad, and R. Prasad, "Threshold Cryptography-based Group Authentication (TCGA) Scheme for the Internet of Things (IoT)," *Proceedings of IEEE VITAE,* pp. 1-5, Aalborg, Denmark, May 2014.

[7] P. Porambage, C. Schmitt, P. Kumar, A. Gurtov, and M. Ylianttila, "Two-phase Authentication Protocol for Wireless Sensor Networks in Distributed IoT Applications," *Proceedings of IEEE WCNC,* pp. 2728-2733, Istanbul, Turkey, April 2014.

[8] X. Yao, X. Han, and X. Du,"A Lightweight Multicast Authentication Mechanism for Small Scale IoT Applications," *IEEE Sensors Journal,* vol.13, no.10, pp.3693-3701, Oct 2013.

[9] V. Petrov, S. Edelev, M. Komar, and Y. Koucheryavy,"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.

[10] Y. Kim, S. Yoo, and C. Yoo,"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.

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

[12] 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.

[13] P. Cotese, F. Gemmiti, B. Palazzi 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.

[14] H. Ghaith, O. Erdinc, and S. Berk, "A Tamper-Proof and Lightweight Authentication Scheme", *Pervasive Mobile Computing,* Vol.4, no.6, pp. 807-818, 2008.

[15] Y. S. Lee, H. J. Lee, and E. Alasaarela, "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.

[16] 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.

[17] R. Maes, "Physically Unclonable Functions: Constructions, Properties and Applications," Katholieke Universiteit Leuven Belgium DEngg Thesis, 2013.

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

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

[20] 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.

[21] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols,* Addison-Welsey Publishing Company, 1994.

[22] "TOTP: Time-Based One-Time Password Algorithm", IETF RFC 6238, 2011.

[23] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication", *ACM Transactions on Computer Systems,* 8, February 1990.

[24] W. Mao and C. Boyd, "Towards formal analysis of security protocols", *Proc. Computer Security Foundations Workshop VI,* pp. 147-158, June 1993.

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

[26] https://www.ece.nus.edu.sg/stfpage/bsikdar/scripts/IoTJ.

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

[28] G. V. Bochman, "Finite state description of communication protocols", *Computer Networks,* Vol. 2, pp. 361-372, 1978.

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

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

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

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

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

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

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

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

APPENDIX

This section serves as a brief introduction to the formal analysis of security protocols using the Mao and Boyd logic [24]. In this techniqe we start by idealizing the protocol messages. Three types of information is used to construct logical formulas: M: messages, P: principals, and F: formula. We use capital letters A, B, P, $Q, \cdots$ to represent principals, letters K, M, $N, \cdots$ to denote messages, while X, Y, $Z, \cdots$ are used for formulas. We use the following predicate constructs in our analysis:

- $P \models X$: P believes X is true and may act accordingly.
- $P \overset{K}{\mid\!\sim} X$: P encrypted X using key K.
- $P \overset{K}{\triangleleft} X$: P sees X using decipherment key K. In the absence of encryption we use $P \triangleleft X$.
- $P \overset{K}{\leftrightarrow} Q$: K is a good shared key for P and Q.
- $\#(M)$: M is fresh (not used before).
- $sup(S)$: Principal S is the trusted party.
- $P \triangleleft \| M$: Message M is not available to principal P.

Next, we present some definitions to understand the rules for protocol message idealisation.

- **Atomic Message:** A piece of data in a message constructed without using any of the symbols ",", "|", "**R**", or "$\{\}$" is called an atomic message. We use "," to separate fields in a message and "$\{\}$" for encryption. The purpose of the symbols "|" and "**R**" is defined below.
- **Challenge:** An atomic message sent and received in separate lines by the same principal (the originator). A time stamp is not considered an atomic message.
- **Replied Challenge:** A challenge that appears in a message sent to the originator.
- **Response:** An atomic message (except timestamps) and a replied challenge sent together by the sender of the response.
- **Nonsense:** An atomic message is a nonsense if it is not a challenge, response, or a timestamp.

We idealize the messages of a protocol using the following rules:

1) Any nonsense is removed.
2) An atomic message is considered a response if it acts as a challenge as well as a response in a line.
3) Combine challenges separated by commas using operator "|".
4) Combine responses separated by commas using operator "|" to form a combined response.
5) Combine a challenge and its response using "**R**" into "*response* **R** *replied challenge*".
6) Combine a message and its corresponding timestamp using "**R**" into "*message* **R** *timestamp*".

Finally we use the following inference rules:

1) **Authentication Rule**: If K is a shared secret key between P and Q, and P used K to decrypt a received message M, then P can believe that Q sent M. The rule is given as
$$\frac{P \models P \overset{K}{\leftrightarrow} Q \wedge P \overset{K}{\triangleleft} M}{P \models Q \overset{K}{\mid\!\sim} M}. \tag{22}$$
2) **Confidentiality Rule**: If K is a shared secret key between P and Q and P encrypted M with K and sent it without sharing it with anyone else, then P can believe that M is only available to P and Q. The rule can be represented as
$$\frac{P \models P \overset{K}{\leftrightarrow} Q \wedge P \models S^c \triangleleft \| M \wedge P \overset{K}{\mid\!\sim} M}{P \models (S \cup \{Q\})^c \triangleleft \| M}. \tag{23}$$
3) **Super-Principal Rule**: P believes what Q believes if P believes Q is a trusted server. The rule is given as
$$\frac{P \models Q \models X \wedge P \models sup(Q)}{P \models X}. \tag{24}$$

This rule can be interpreted as P can trust Q about X. This means that an IoT device can be the super-principal for a nonce it generated. For example, in Protocol 1 IoT device ID_A generates N_A. Therefore, we can consider ID_A as the super principal in terms of N_A. This fact has been used in the tableaux of Figures 4(c) and 4(e).

4) **The Fresh Rule**: P can believe N is fresh if P believes M is fresh and P has recieved N and M together in a message. The rule is given as

$$\frac{P \mid\equiv \#(M) \wedge P \triangleleft N\mathbf{R}M}{P \mid\equiv \#(N)}. \tag{25}$$

5) **The Good-Key Rule**: There are two variations to this rule: (i) if P believes K is only available to P and Q, and P knows that K is fresh, then P can believe K is a good key between P and Q

$$\frac{P \mid\equiv \{P,Q\}^c \triangleleft\| K \wedge P \mid\equiv \#(K)}{P \mid\equiv P \overset{K}{\leftrightarrow} Q} \tag{26}$$

and (ii) if P believes K is only available to P, Q and R and no one else, and P trusts R, and P knows K is fresh, then P can believe K is a good key between P and Q. The rule is given as

$$\frac{P \mid\equiv \{P,Q,R\}^c \triangleleft\| K \wedge P \mid\equiv sup(R) \wedge P \mid\equiv \#(K)}{P \mid\equiv P \overset{K}{\leftrightarrow} Q}. \tag{27}$$

6) **Intuitive Rule**: P has seen message M if it can decrypt message M using K. The rule is given as

$$\frac{P \overset{K}{\triangleleft} M}{P \triangleleft M}. \tag{28}$$

7) **Derived Rule**: This rule is obtained by combining the belief axiom

$$P \mid\equiv (X \bigwedge Y) \text{ if and only if } P \mid\equiv X \bigwedge P \mid\equiv Y \tag{29}$$

with the confidentiality rule. The rule can be represented as

$$\frac{P \mid\equiv Q \mid\equiv P \overset{K}{\leftrightarrow} Q \wedge P \mid\equiv Q \mid\equiv S^c \triangleleft\| M \wedge P \mid\equiv Q \overset{K}{\mid\sim} M}{P \mid\equiv Q \mid\equiv (S \cup \{P\})^c \triangleleft\| M}. \tag{30}$$