Data Provenance for IoT with Light Weight Authentication and Privacy Preservation

Muhammad Naveed Aman, Mohammed Haroon Basheer, *Student Member, IEEE*
Biplab Sikdar, *Senior Member, IEEE*

***Abstract*—The Internet of Things (IoT) engulfs a large number of interconnected heterogeneous devices from a wide range of pervasive application areas including health-care systems, energy management, environmental monitoring, and home and commercial automation. Although IoT is considered an enabling technology for a variety of services, it also raises many security and privacy concerns. This paper focuses on developing secure protocols for data provenance with authentication and privacy preservation in IoT systems. Protocols for two scenarios are presented, one when an IoT device is directly connected to a wireless gateway and the other when an IoT device is indirectly connected to the wireless gateway through multiple hops of other IoT devices. The proposed protocols use Physically Unclonable Functions along with wireless link fingerprints derived from the wireless channel characteristics between two communicating entities. This results in protocols which are not only efficient in terms of computational complexity and energy requirements but are also safe against various types of attacks including physical and cloning attacks. Experimental results show that in comparison to existing protocols, the proposed protocols are upto 100% more accurate in detecting attacks on data provenance and can save upto 83.8% and 73.5% energy consumption for the IoT devices in terms of CPU and radio energy, respectively.**

***Index Terms*—Internet of Things, Physically Unclonable Functions, RSSI, Data Provenance, Authentication.**

I. Introduction

Internet of Things (IoT) represents a network of connected heterogeneous devices to enable intelligent services in a wide range of domains including industrial automation, home/building automation (heating, air conditioning, ventilation, lighting, fire, access control), smart health care systems, smart agriculture, global supply chain, and smart things (connected homes, cars, RFID, and cities). The number of IoT devices is growing exponentially, and according to Cisco, over 50 million devices are expected to be connected to the Internet by 2020 [1]. Moreover, the heterogeneity of devices and the sensitivity and volumes of data generated by IoT devices raises serious security concerns. Authentication, privacy and data provenance are among the top security challenges.

The lack of trust in the digital world spurs from the use of online credentials with low levels of authentication assurance. Secure authentication is crucial for IoT systems given the fact that IoT devices may be deployed out in the open and remote locations. This exposes them to physical attacks which was not a concern in the traditional Internet where personal computers are considered physically protected. Furthermore, the resource constrained nature of IoT devices makes the task of designing secure protocols even more challenging. In this paper we use Physically Unclonable Functions (PUFs) to establish the root of trust in IoT systems for authentication. PUFs can provide a challenge-response mechanism by exploiting the (sub-)microscoping structure of integrated circuits. The use of PUFs can provide security against physical and cloning attacks [2].

Data provenance institutes trust in the origin and creation process of data. Through data provenance, a user can warrant confidence in the fidelity of data, i.e., that the data is indeed collected by the specific IoT device at the stated location and time. Trustworthiness of the the data generated by IoT devices is of utmost importance for the correct operation of IoT based systems [3]. For example, consider the case of a nuclear power plant where the temperature and pressure need to be monitored and maintained within a strict range by IoT devices. An adversary may try to invalidate this data by moving an IoT device to a different location or even cloning it.

Most of the existing techniques on data provenance are related to databases. Work on data provenance in IoT networks is limited and most of the existing techniques are vulnerable to physical, cloning, impersonation, and denial of service (DoS) attacks. Moreover, these techniques solely depend upon computationally intensive cryptographic operations. To solve these issues, this paper exploits the wireless channel characteristics (such as received signal strength indicator (RSSI) measurements) between two entities to generate "wireless fingerprints", that are unique to the two communicating entities. Also, in this paper we use pseudonym identities in place of the real identities of the IoT devices to provide privacy preservation. The sensitive and personal nature of the data generated by IoT devices calls for security protocols with privacy preservation. In particular, an adversary should not be able to link a communication session between an IoT device and server to the identity of that particular device. Privacy

This work was supported in part by the National Research Foundation, Prime Ministers Office, Singapore under its Corporate Laboratory@University Scheme, National University of Singapore, and in part by the Singapore Telecommunications Ltd. (Corresponding author: Muhammad Naveed Aman)

M. N. Aman and M. H. Basheer are with the Department of Computer Science, National University of Singapore, 13 Computing Drive, Singapore 117417, e-mail: naveed@comp.nus.edu.sg, haroon.basheer@nus.edu.sg.

B. Sikdar is with the Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117576, e-mail: bsikdar@nus.edu.sg.

leakage can result in serious security threats. For example, if an adversary is able to analyze the electricity usage patterns of a particular house, he/she may be able to identify opportunities for burglary, i.e., periods when the house is empty. Similarly, if an adversary can link a piece of data with a wearable IoT device such as the one transmitting critical health data to a server, he/she may be able to tamper, drop or replay (older) data.

This paper focuses on secure data provenance for IoT with light weight mutual authentication and privacy preservation. The major contributions of this paper are as follows:

1) A technique based on RSSI measurements of the wireless channel between two entities to distinguish legitimate channels from adversarial channels, thereby establishing data provenance in terms of the location of data.
2) PUF based authentication protocols to establish the data provenance in terms of the source of data.
3) Data provenance protocols for single hop and multi-hop data communication scenarios.
4) Experimental results confirming that the proposed technique can generate unique and close to perfect matching wireless fingerprints for typical data exchanges between an IoT device and a base station.

The rest of the paper is organized as follows. Section II discusses the related work. Section III provides the background for this work and Section IV discusses our network model, assumptions, and security requirements. Section V describes the technique to derive wireless fingerprints from wireless channel characteristics. Section VI presents the proposed single hop and multi-hop data provenance protocols. The security analysis is presented in Section VII, and the formal security analysis using BAN logic is presented in Section VIII. The experimental results are discussed in Section IX. Performance analysis of the proposed protocol is presented in Section X and we conclude the paper in Section XI.

II. Literature Review

Recent works on data provenance for IoT include the following. The authors of [4] discuss the challenges of implementing data provenance in IoT networks. Similarly, [5] discusses the integration of provenance within the IoT. The authors propose a trace-based provenance collection framework for IoT devices called Provenance Aware Internet of Things System (PAIoTS). However, the authors focus on the aspect of collecting provenance information rather than securing it. The authors of [6] present a provenance based trust management system for the IoT. However, this technique requires dedicated trusted hardware in the form of data provenance modules to be added to IoT devices. In another work [7], a hash chain based scheme is proposed to transmit provenance information for IoT sensor data across multiple hops (IoT devices). However, the proposed technique depends on a hash of the identity of each IoT device, which makes it vulnerable to impersonation attacks. The authors of [8] propose the use of non-interactive zero-knowledge proofs (NI-ZKP) to establish data provenance. However, this technique depends on complex computations.

The above discussion shows that most of the existing techniques for authentication and data provenance in IoT suffer from one or more of the following problems:

1) Require **specialized hardware** not suitable for simple IoT devices.
2) Depend on **complex computations** not suitable for resource constrained IoT devices.
3) Secret keys are stored in the device's memory making them vulnerable to **physical and cloning attacks**.
4) Do not provide **privacy preservation**.
5) Protocols proposed are only for the simple single hop scenarios, i.e., when the IoT device is directly connected to the Internet through a border gateway.

PUFs have mainly been used in security for the purpose of key generation and authentication. Some of the recent works on using PUFs for authentication and secret key establishment include [9]–[11]. However, these techniques do not provide any mechanism to establish data provenance. Combining a PUF with sensor readings has been proposed in [12]. However, while the proposed technique successfully establishes the authenticity of the identity of the IoT device generating the data, it can not provide any guarantees as to the location from which the data has been gathered. For example, if the sensor is moved from its original location with malicious intent, then the scheme breaks down, i.e., the receiver of the data will continue accepting invalid sensor readings without knowing that the location of the data's origin has changed.

To solve the problems identified above, this paper uses the following techniques:

1) The proposed protocols use symmetric key cryptography which is light weight and suitable for resource constrained IoT devices.
2) PUFs are used to provide protection against physical and cloning attacks. PUFs support ultra high throughput with ultra low energy and silicon area footprints which makes them suitable for simple and low cost IoT devices [13]. The use of PUFs eliminates the need to store secret keys in an IoT device's memory.
3) This paper uses PUFs to establish the data provenance in terms of the source of the data and wireless fingerprints to establish the data provenance in terms of the location and path that the data traverses. Note that the use of PUFs for data provenance has not been proposed earlier. Similarly, to the best of our knowledge, PUFs and wireless fingerprints have not been used together in the existing literature.
4) The proposed protocols use pseudonym identities for the IoT devices, hiding the actual identities and providing privacy preservation.

III. Background

A. Introduction to Physically Unclonable Functions

This section gives a brief introduction to PUFs which will help in understanding the proposed protocols.

A PUF is characterized by the intractably complex physical system embedded into an integrated circuit (due to random variations in the manufacturing process) giving rise to a unique

challenge response mechanism. A PUF can be represented as $R = P(C)$, i.e., a PUF P maps a challenge C to a unique response R. A challenge and its corresponding response from a PUF is termed a challenge-response pair (CRP). If a PUF is excited using the same challenge multiple times, it will always produce the same response. However, if the same challenge is used to excite a different PUF, the response will be significantly different. This implies that each PUF is unique in terms of its CRPs.

PUFs are considered sensitive to environmental factors. Therefore, the PUF output for a given challenge may vary slightly due to environmental factors such as temperature. However, the use of fuzzy extractors has been proposed to avoid this problem and obtain stable PUF outputs suitable for security applications [14], [15]. Thus, this paper assumes the use of ideal PUFs. PUFs have shown resilience against physical and invasive attacks and can be employed to generate secret keys without actually storing them, making them attractive for hardware authentication and secure key generation [13], [16]–[18]. Different types of PUFs include delay-based PUFs (exploiting the variation among circuit delays), and memory based PUFs (exploiting the random process variability in memory cells), among others [19].

B. Security from Wireless Channel Characteristics

The use of wireless channel characteristics to derive security primitives has gained interest over the last decade. The theory behind these techniques is as follows: consider two communicating parties, Alice and Bob, that encounter an intrinsically symmetric wireless channel. Then, if Alice and Bob transmit identical signals, they will receive identical signals, given they use identical transceivers and antennas. Moreover, according to the reciprocity property of electromagnetic wave propagation, the multiple path propagation of radio signals is identical in both directions, resulting in identical phase shifts, delays, and gains. Thus, if Alice and Bob measure parameters such as radio signal strength, delay, and angle of arrival, etc., the measurements will agree to a high degree.

Jake's fading model [20] states that if either of two communicating parties moves a distance larger than half a wavelength, then the wireless channel decorrelates rapidly until it becomes independent for a distance greater than one wavelength. This shows that wireless channels are highly sensitive to spatio-temporal changes and location. Therefore, if an adversary is located at a distance of at least one wavelength, he/she will effectively experience an independent wireless channel and is unable to obtain Alice and Bob's channel measurements. Thus, Alice and Bob may exploit the unique channel measurements that they share for security purposes.

The use of wireless channel characteristics as security primitives has been studied widely for a broad range of platforms as well as a variety of applications. The use of wireless channel characteristic for secret key generation has been studied and proposed for software defined radios [21], UWB Communications [22], Bluetooth [23], and WiFi networks [24]–[26]. The use of wireless channel measurements for security in wireless body area networks has also been extensively studied in [27]–[29]. Similarly, the authors of [30] quantify the effects of small-scale fading on secret-key agreement in an office space and an anechoic chamber. They propose a technique to generate a secret key only when the wireless channel has high variation. Other works that exploit the entropy of the shared wireless channel state include secure pairing [31], proximity based authentication [32], intrusion detection [33], and detecting spoofing and Sybil attacks [34], [35].

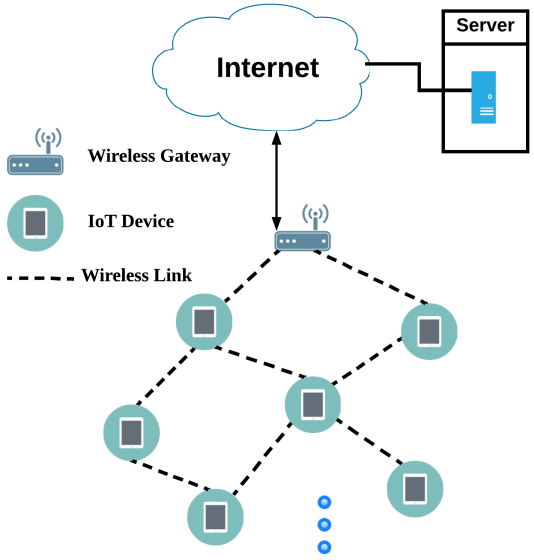


Fig. 1: Network model.

Deriving data provenance from wireless channel characteristics has been proposed in [36]. The authors propose the extraction of wireless fingerprints from the received signal strength indicator (RSSI) for body area networks. The proposed technique computes a Pearson correlation coefficient of the wireless fingerprints derived individually at the transmitter and the receiver. However, this technique depends on long wireless fingerprints which increase the communication overhead. Furthermore, the authors do not present a complete protocol to use with the wireless fingerprints and the proposed protocol does not support privacy preservation. Similarly, in [37], the authors use the same technique as [36] to generate the wireless fingerprints and present a protocol for establishing data provenance in multi-hop IoT networks. However, this technique suffers from a serious key leakage problem.

IV. Network Model, Assumptions, Threat Model, and Security Objectives

A. Network Model

The network model consists of a set of IoT devices, 6LoWPAN wireless gateway, and the server. The IoT devices are wirelessly connected to each other. Some of the IoT devices close to the wireless gateway are connected to it through a wireless link. The network model is shown in Figure 1.

B. Notations

ID_A, $\{M\}_k$, C^i, and R^i denote the ID of IoT device A, message M encrypted using key k, challenge to a PUF, and the response of a PUF for C^i, respectively. Similarly, $\|$ represents the concatenation operator.

C. Assumptions

The following assumptions are made in this paper:

a. Every IoT device has a PUF embedded with the device's microcontroller, forming a system-on-chip (SoC). Any attempt to tamper/separate the PUF from the device will make the PUF un-usable [39], [40]. This in turn implies that the device will not be able to use the PUF for any security processes.

b. The communication between the PUF and microcontroller is secure and cannot be accessed from outside given the SoC assumption [39], [40].

c. The wireless gateway is considered to be secure but not all IoT devices are assumed to be honest.

d. The server is assumed to be trusted and secure.

e. IoT devices have limited resources such as energy, memory, and processing capabilities. However, the server does not have such limitations.

f. We make the standard assumption about a PUF: every PUF is unique and it is not possible to predict its behavior, i.e., it is un-clonable [2]. We can model a PUF as PUF: $\{0,1\}^{l_1} \rightarrow \{0,1\}^{l_2}$, i.e., if an input of length l_1 is given to a PUF it will produce an output of length l_2. The security of a PUF can be modeled by the following security game (denoted by $\text{Exp}^{\text{Sec}}_{PUF,\mathcal{A}}$ between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$:
   (i) $\mathcal{A}$ randomly chooses a challenge C^i and sends it to $\mathcal{C}$.
   (ii) $\mathcal{C}$ uses the PUF to obtain the response R^i and reveals R^i to $\mathcal{A}$.
   (iii) $\mathcal{C}$ selects a random challenge C^x which has not been used before and obtains the response R^x using the PUF, i.e., $R^x = PUC(C^x)$.
   (iv) $\mathcal{A}$ can query the PUF a polynomial number of times for challenges other than C^x.
   (v) $\mathcal{A}$ outputs its guess $R^{x'}$ for the challenge C^x.
   (vi) $\mathcal{A}$ wins the game if $R^{x'} = R^x$.
   We can represent the advantage of adversary $\mathcal{A}$ in this game by $\text{Adv}^{PUF}_{\mathcal{A}} = \texttt{Pr}[R^{x'} = R^x]$.

g. According to Jake's fading model, a channel is symmetric between a transmitter (Tx) and receiver (Rx). However, if an entity is moved away from the Tx the wireless channel decorrelates rapidly and eventually may be assumed to be independent for a distance greater than one wavelength. We assume that the adversary is located atleast one wavelength away from a legitimate IoT device and thus, cannot infer the wireless fingerprints between legitimate IoT devices. The security of the wireless fingerprints can be modeled by the following security game denoted by $\text{Exp}^{\text{Sec}}_{FP,\mathcal{A}}$:
   (i) $\mathcal{C}$ selects two IoT devices ID_1 and ID_2.
   (ii) An adversary $\mathcal{A}$ randomly chooses a location atleast one wavelength away from ID_1 and ID_2.
   (iii) $\mathcal{A}$ is allowed to gather the wireless fingerprints $F_{\mathcal{A}1}$ and $F_{\mathcal{A}2}$ for its communication with ID_1 and ID_2.
   (iv) $\mathcal{C}$ initiates a communication session between ID_1 and ID_2 and obtains the wireless fingerprint F_{12} for the channel between ID_1 and ID_2.
   (v) $\mathcal{A}$ can communicate with ID_1 and ID_2 (for the purpose of acquiring wireless fingerprints) a polynomial number of times from any location atleast one wavelength away from ID_1 and ID_2.
   (vi) $\mathcal{A}$ outputs its guess F^*_{12} for the wireless fingerprint between ID_1 and ID_2.
   (vii) $\mathcal{A}$ wins the game if $F^*_{12} = F_{12}$.
   We can model the advantage of the adversary in this game by $\text{Adv}^{FP}_{\mathcal{A}} = \texttt{Pr}[F^*_{12} = F_{12}]$.

D. Threat Model

A set of IoT devices $\mathcal{ID} = ID_1, ID_2, \cdots, ID_n$ generate data and forward it to the trusted server S. However, before communicating data to the server, the IoT devices need to authenticate themselves with the server over an insecure network. At the completion of the authentication protocol, the entities either accept the authentication or reject it. If authentication is accepted, the IoT devices start transferring data to the server over an insecure channel. It is assumed that the attacker $\mathcal{A}$ has full control over the communication channel between the IoT devices and the server. This includes eavesdropping, replaying, tampering, and injecting packets in the network. This set of attacks can be modeled with the following set of queries:

- `Send`S(S, `m0`, `r0`, `m1`) is used to model the query where the attacker $\mathcal{A}$ acts like a legitimate IoT device and sends a message `m0` to S and receives `r0`. The IoT device then replies to S with `m1`.
- `Send`ID(ID, `m0`, `r0`) is used to model the query where the attacker $\mathcal{A}$ acts like a server and sends a message `m0` to an IoT device and receives `r0`.
- `Monitor`(ID,S) is used to model the attacker's ability to continuously eavesdrop and monitor the radio channel between IoT device ID and S.
- `Drop`($\mathcal{A}$) is used to model the attacker's ability to drop packets between ID and S. This may be done in two ways; First, $\mathcal{A}$ may directly drop a packet passing through the network. Secondly, $\mathcal{A}$ may cause a packet to be dropped by using a malicious node to deliberately falsify one or more authentication parameters in the proposed protocols. This query may be used by the attacker to launch a DOS attack, i.e., dropping selective packets to interrupt the synchronization between the two entities. Note that, the attacker may also use this query to drop (or cause to drop) all or some randomly selected packets from the source to the destination to degrade the network performance. However, in this paper we do not consider such type of attacks termed as grey hole/black hole attacks.
- `Reveal`(ID) is used to model the attacker's ability to launch a physical attack on an IoT device and obtain the secrets stored in its memory.

The attacker $\mathcal{A}$ can call Send$S$, SendID, Monitor, and Drop any polynomial number of times. However, as mentioned in the assumptions, any physical tampering with an IoT device renders it useless, therefore, $\mathcal{A}$ may call Reveal only once.

E. Security Requirements

The protocols proposed in this paper intend to achieve the following security requirements:

1) Mutual authentication of the IoT device and server.
2) Establish data provenance including the identity and location of an IoT device.
3) Ensure privacy by providing forward secrecy and backward untraceability.
4) Protection against DoS attacks.
5) Protection against physical and cloning attacks by ensuring no secrets are stored in an IoT device's memory.

V. Proposed Technique for Data Provenance using Wireless Fingerprints

In this paper we propose the use of wireless fingerprints derived from the RSSI values at the receiver and the transmitter and comparing them on the basis of the mean squared error, *MSE*. To establish the legitimacy of the wireless channel between a legitimate IoT device and base station, we calculate the mean squared error *MSE* as follows:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2 \tag{1}$$

where X_i and Y_i represent the RSSI values for the *i*-th packet at each entity and n is the wireless fingerprint size.

The procedure for validating the wireless fingerprints between two entities Alice and Bob is as follows:

1) Alice and Bob construct their respective wireless fingerprints individually by concatenating the RSSI values for the wireless link between them for a specified period of time depending on the wireless fingerprint size used.
2) Alice and Bob send their wireless fingerprints to a server for verification.
3) The server compares the wireless fingerprints of two communicating entities by calculating the *MSE* of the two wireless fingerprints. The server then compares the resulting *MSE* to a threshold value.
4) If the *MSE* of the wireless fingerprints is above the threshold value then the wireless fingerprints are considered invalid, pointing to a possible attack by an adversary.

The threshold value for *MSE* is determined using experiments described in Section IX. The experimental results show that we can effectively distinguish between a legitimate channel and an adversarial channel using the *MSE* of the wireless fingerprints. The experimental evidence also shows that the proposed technique not only outperforms [36] in terms of miss-classification rates but also requires shorter fingerprints.

VI. Proposed Data Provenance Protocols

In this section we describe the proposed data provenance protocols for two scenarios: single hop and multi-hop. In the single hop scenario, the source IoT device is directly connected to the wireless gateway. On the other hand, in the multi-hop scenario the source IoT device is indirectly connected to the wireless gateway through intermediate IoT devices.

A. Device Registration

The server needs an initial CRP (C^i, R^i), pseodonym identity (PID^i) and a list of emergency CRPs (C_{em}) and emergency identities (EID) for each IoT device which can be done using a time-based one-time password algorithm (TOTP) [38]. The exchange of the initial parameters is carried out with the help of an operator using a password and TOTP at the time of first deployment of an IoT device. The server stores (C^i, R^i), PID^i, C_{em}, and EID. The IoT device stores C^i, PID^i, C_{em}, and EID. We also assume that the wireless gateway and the server have a pre-shared secret symmetric key k_{GS}.

B. Single Hop Data Provenance

This section describes the proposed protocol for the case when the IoT device is directly connected to the wireless gateway with no intermediate IoT devices. Let us consider an IoT device ID_A directly connected to the wireless gateway, who wants to send some data to the server. The proposed protocol for this scenario has two phases, i.e., authentication phase and data transfer phase as shown in Figures 2 and 3. The steps of the proposed protocol are as follows:

1) IoT device ID_A generates the response R^i using the stored challenge C^i. It then sends message M_0 containing the device's pseudonym identity and a random nonce encrypted with R^i, i.e., $\{N_a\}_{R^i}$ along with an authentication parameter $I_0 = H(M_0 \parallel R^i)$ to the server through a wireless gateway ID_G. This is shown in message 1 in Figure 2. Note that the authentication parameter is used to establish the data integrity of messages in this paper.
2) The wireless gateway ID_G generates the wireless fingerprint F_{AG} for the wireless link between IoT device ID_A and the wireless gateway ID_G and forwards Message 1 to the server.
3) The server locates PID^i_A it its memory and reads the corresponding CRP (C^i, R^i). If the server cannot locate PID^i_A, the authentication request is rejected. The server then verifies the authentication parameter I_0. If verification fails, the authentication request is rejected. Otherwise, the server decrypts M_0 to obtain N_a and generates a random nonce N_b. It then creates a message $M_1 = PID^i_A, \{N_a, N_b\}_{R^i}$ and sends it along with the authentication parameter I_1 to the IoT device in Message 2 in Figure 2.
4) When the IoT device receives Message 2 from the wireless gateway ID_G, it generates the wireless fingerprint F_{GA} using the RSSI values of the received packet. It then decrypts M_1 to obtain N_b and verifies

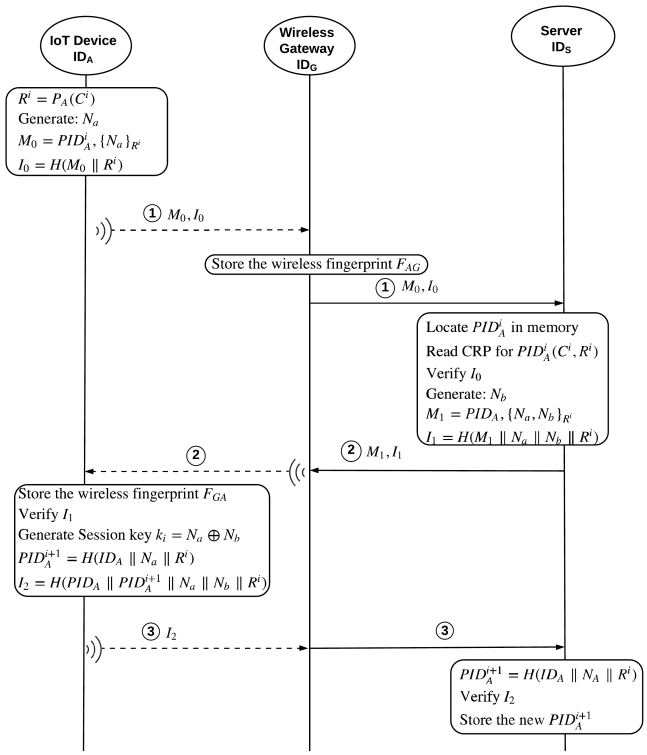


Fig. 2: Single hop data provenance: Authentication Phase.

the integrity of the message using I_1. If verification fails, then the authentication request is terminated. Otherwise, the IoT device generates the session key as follows: $k_i = N_a \oplus N_b$. It then updates its pseudonym identity as $PID_A^{i+1} = H(ID_A \parallel N_a \parallel R^i)$. The IoT device then sends an acknowledgment to the server in the form of the authentication parameter I_2.

5) The server updates the pseudonym identity of IoT device ID_A and verifies I_2. If the verification fails, then the authentication request is terminated. Otherwise, the server accepts the authentication and stores PID_A^{i+1} for any future authentications.

After the successful completion of the authentication phase, the IoT device ID_A can now transfer data to the server as shown in Figure 3. The steps of the data transfer phase are as follows:

1) The IoT device creates the message D_A carrying the data as follows: $D_A = PID_A^i, n_1, \{Data, F_{GA}\}_{k_i}$, where n_1 is a random nonce acting as the freshness identifier for this message and F_{GA} is the wireless fingerprint generated by IoT device ID_A during the authentication phase. The IoT device ID_A then forwards D_A along with the authentication parameter V_A to the wireless gateway in Message 4 in Figure 3.
2) The wireless gateway generates a message D_G by encrypting the wireless fingerprints F_{AG} generated during the authentication phase. It then forwards Message 4 along with D_A and the corresponding authentication parameter V_G to the server.
3) The server decrypts D_A and D_G using k_i and k_{GS}, respectively, to obtain the data sent by the IoT device ID_A and the wireless fingerprints F_{AG} and F_{GA}. It then verifies the authentication parameters V_A and V_G and using the technique described in Section III-B, validates the provenance of the data using the wireless fingerprints. If the validation fails, the data is discarded. Otherwise, the data is accepted and the server sends an acknowledgment in the form of the authentication parameter V_S to the IoT device ID_A.
4) After receiving the acknowledgment from the server, the IoT device ID_A verifies V_S. If verification fails, then the IoT device ID_A may retry to send the data again. Otherwise, the IoT device ID_A may continue to send more data using the same steps as above or may conclude the session.

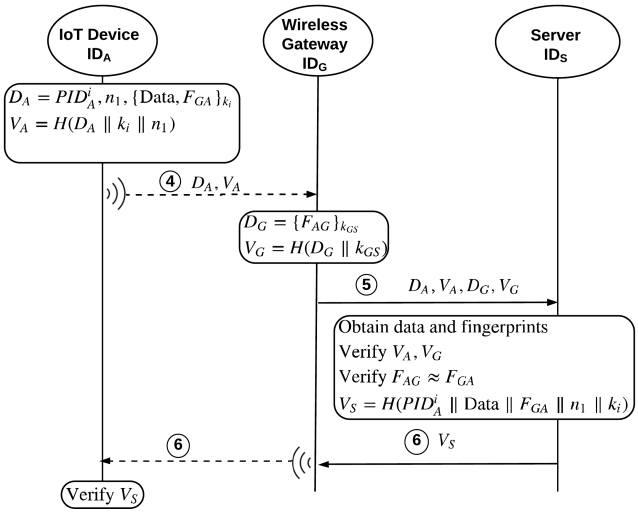


Fig. 3: Single hop data provenance: Data Transfer Phase.

C. Multi-Hop Data Provenance

The multi-hop data provenance protocol is proposed for the case when an IoT device is not directly connected to a wireless gateway and needs to relay its messages/data through intermediate IoT devices. Let us consider an IoT device ID_1, who wants to send some data to the server. However, the IoT device ID_1 is indirectly connected to the wireless gateway through intermediate IoT devices ID_2 and ID_3. The proposed protocols for the authentication phase and data transfer phase are shown in Figures 4 and 5, respectively. The steps for the authentication phase are similar to the single hop data provenance protocol except that now each intermediate IoT device stores two wireless fingerprints, i.e., one for each direction of the multi-hop link. For example, IoT device ID_2 stores the wireless fingerprints F_{12} and F_{32} for the wireless links between IoT devices ID_1 and ID_2; and ID_2 and ID_3, respectively.

After successfully completing the authentication phase, the data transfer phase for the multi-hop scenario is shown in Figure 5. The steps of the data transfer phase are as follows:

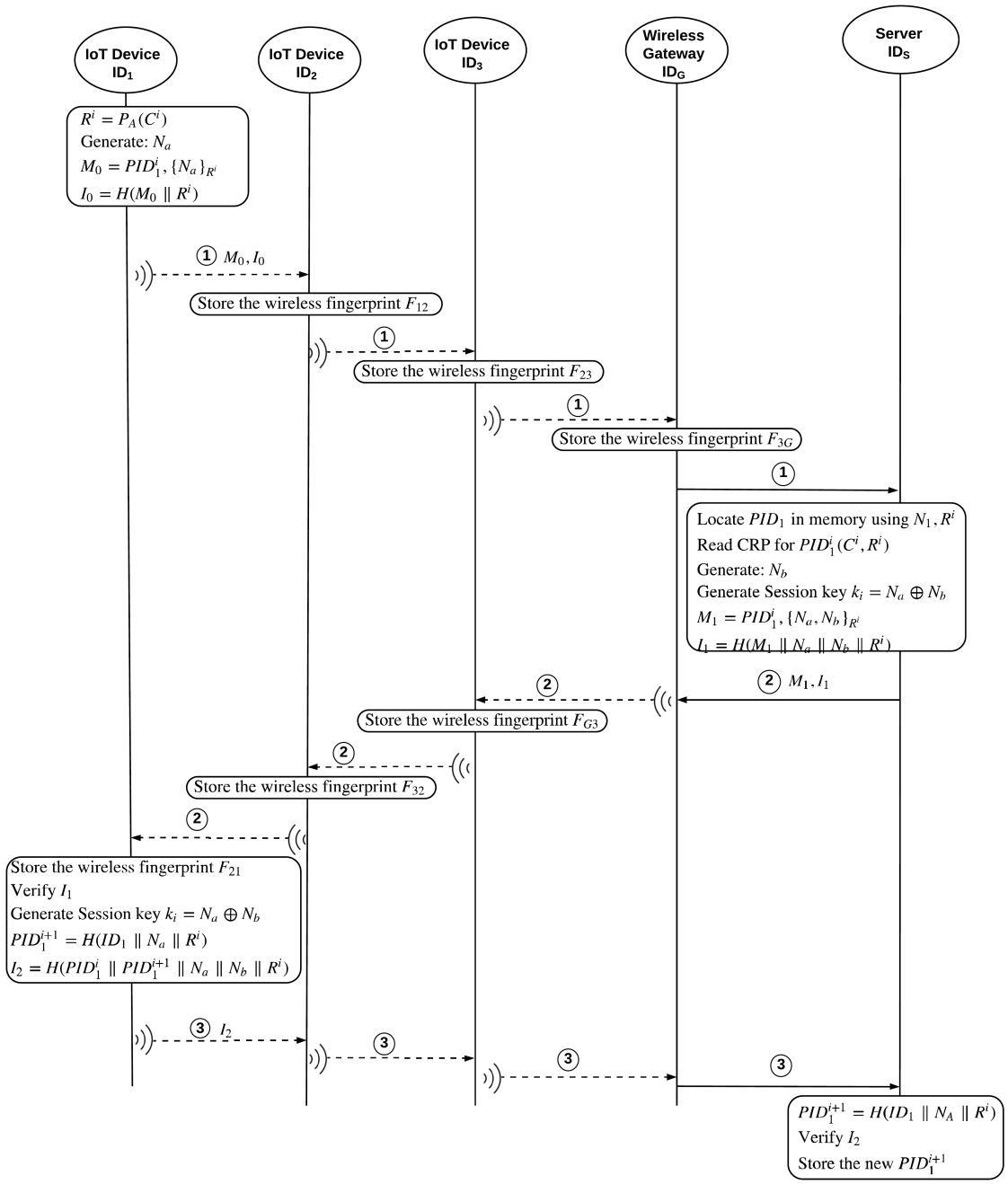


Fig. 4: Multiple hop data provenance: Authentication Phase.

1) IoT device ID_1 sends Message $D_1 = PID_1^i, n_1, \{Data, F_{21}\}_{k_i}$ and the corresponding authentication parameter V_1 to intermediate IoT device ID_2 in Message 4 in Figure 5.
2) IoT device ID_2 carries out the following steps. Note that each intermediate IoT device (in this case IoT devices ID_2 and ID_3) carry out the same steps.
   (i) Generate the response R^i using the stored challenge and the device's PUF.
   (ii) Generate a random nonce N_{a_2} and update the device's pseudonym identity as follows: $PID_2^i = H(ID_2 \parallel N_{a_2} \parallel R^i)$.
   (iii) Create a message D_2 by encrypting N_{a_2}, and the wireless fingerprints F_{12} and F_{32}.
   (iv) Forward Message 4 along with D_2 and corresponding authentication parameter V_2 to the next hop, i.e., IoT device ID_3 as shown in message 5 in Figure 5.
3) IoT device ID_3 carries out the same steps as IoT device ID_2.
4) The wireless gateway encrypts the stored wireless fingerprint F_{3G} using k_{GS} to form Message D_G and forwards

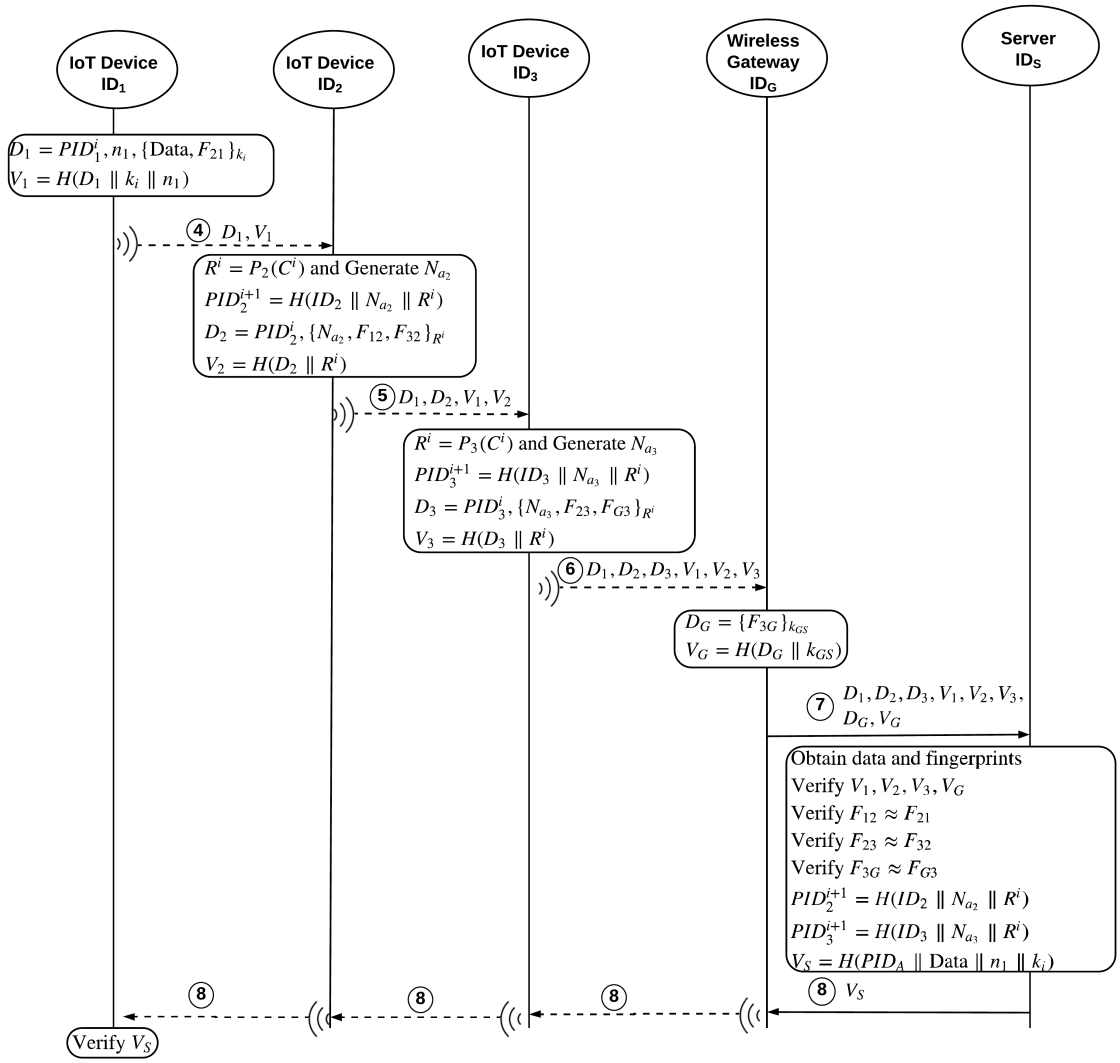


Fig. 5: Multiple hop data provenance: Data Transfer Phase.

all the previous messages along with D_G and the corresponding authentication parameter V_G to the server.

5) The server obtains the data and the corresponding fingerprints using the stored secret keys and CRPs for each IoT device. It then verifies all the authentication parameters. If verification fails, the packet is discarded. Otherwise, the server validates the data provenance of the received data using the technique described in Section III-B and the wireless fingerprints. If the validation fails, then the server discards the packet. Otherwise, the server updates the pseudonym identities of the intermediate nodes and generates an acknowledgment in the form of the authentication parameter V_S. The server then sends V_S to the IoT device ID_1.

D. CRP Update

The proposed data provenance protocols store a list of CRPs at the server, with one CRP for each IoT device. However, to maintain freshness, the server needs to update the CRP and obtain a new CRP. This is done using the CRP update protocol shown in Figure 6. The steps for the CRP update protocols are as follows:

1) The server initiates the CRP update by sending Message 1 to the IoT device ID_A in Figure 6. The server requests the IoT device ID_A to send the response for the new challenge C^{i+1} through this message.
2) The IoT device ID_A generates the response R^i using its PUF and decrypts Message M_1 to obtain C^{i+1} and N_1. It then verifies the MAC. If verification fails, then the CRP update request is rejected. Otherwise, the IoT device ID_A generates a random nonce N_2 and the new response as follows: $R^{i+1} = P_A(C^{i+1})$. The IoT device ID_A then updates its pseudonym identity and sends R^{i+1} in Message $M_2 = \{R^{i+1}, N_1, N_2\}_{R^i}$ along with the corresponding MAC to the server. Note that N_1 and N_2 are the random nonce used by the IoT device ID_A and the server as freshness identifiers.
3) After receiving Message 2 from the IoT device ID_A in

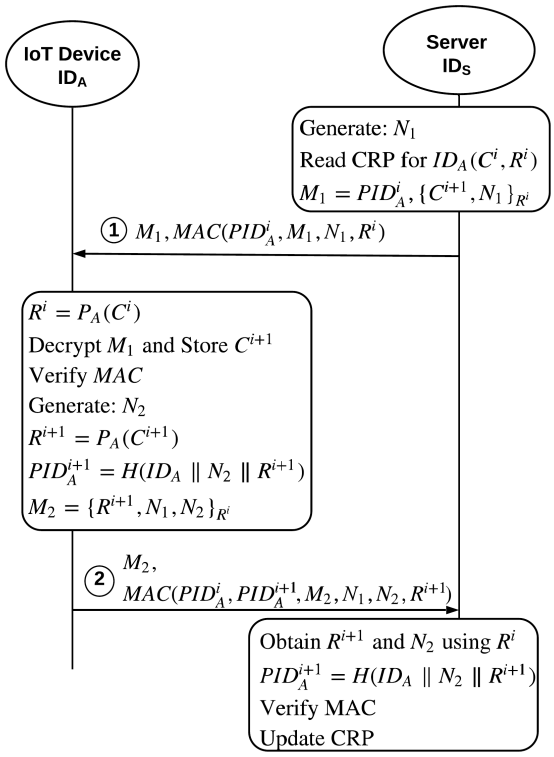


Fig. 6: Protocol for CRP update.

Figure 6, the server first decrypts M_2 to obtain R^{i+1} and N_2, and it then generates the new pseudonym identity PID_A^{i+1} for IoT device ID_A. The server then verifies the MAC and if the verification fails, the CRP update is terminated and the server may restart the update process at a subsequent time. Otherwise, the current CRP at the server for IoT device ID_A is replaced with the new CRP (C^{i+1}, R^{i+1}) and CRP update is considered complete.

VII. Security Analysis

In this section, we present a formal security analysis on the security requirements outlined in Section IV-E.

Lemma 1. *The secrets used in the proposed protocol cannot be revealed even after calling the* `Reveal` *oracle.*

Proof. In the proposed protocols the IoT devices do not store any secret in their memory except for the current challenge C^i, pseudonym identity PID_{ID}^i and a list of emergency identities EID and challenges C_{em}. Therefore, if the attacker calls a `Reveal` oracle to extract secrets from an IoT device's memory, then he/she may only be able to obtain C^i, PID_{ID}^i, EID, and C_{em}. However, the attacker cannot reveal the secret response R^i from C^i or the other stored parameters. Note that $\mathcal{A}$ cannot use C^i or C_{em} to obtain R^i because of the SoC assumption. □

Lemma 2. *In the proposed protocols the pseudonym identities of an IoT device cannot be correlated even after calling the* `Reveal` *oracle.*

Proof. Each pseudonym identity PID_{ID}^i is valid for one round and is updated using a random nonce, the PUF response, and a one-way hash function H, i.e., the pseudonym identity for the next round is created as $H(ID_A, N_a, R^i)$. Thus, without knowledge of R^i the attacker cannot correlate the pseudonym identity for the current round PID_{ID}^i with that of the next or previous round. Note that even if the attacker $\mathcal{A}$ calls the `Reveal` oracle, then according to lemma 1, he/she cannot obtain any secrets including R^i. □

Lemma 3. *In the proposed protocols the attacker cannot obtain a valid wireless fingerprint even after calling the* `Reveal` *oracle.*

Proof. In the data transfer phase of the proposed protocols, the IoT device sends its wireless fingerprint FP along with the data, i.e., the datagram sent is $D_{ID} = PID_{ID}^i, n_1, \{Data, FP\}_{k_i}$ with the authentication parameter $H(D_{ID} \parallel k_i \parallel n_1)$. Similarly, in the multi-hop protocol, each IoT device relaying the data also attaches its wireless fingerprints to the datagram, i.e., $D_{ID} = PID_{ID}, \{FP\}$ along with the authentication parameter $H(D_{ID} \parallel R^i)$. In these messages the two secret parameters are k_i and R^i. Note that k_i is dynamically generated during the authentication phase and according to Lemma 1 the adversary $\mathcal{A}$ cannot obtain R^i by calling the `Reveal` oracle. Moreover, the adversary is at a distance of atleast one wavelength from the legitimate IoT device. Therefore, $\mathcal{A}$ cannot generate the correct wireless fingerprints. Thus, by calling the `Reveal` oracle the adversary can only obtain C^i, PID_{ID}^i, EID, and C_{em}. However, with these he/she cannot pass the server's verification process. □

Theorem 4. *Mutual Authentication: If an IoT device successfully completes a run of the protocol, it has indeed done so with the legitimate server. Similarly, if the server has successfully completed a run of the protocol with an IoT device, it has indeed done so with the legitimate device.*

Proof. The adversary may try to authenticate itself as a legitimate IoT device or server. There are two parts to this proof. First we consider the case when the adversary tries to authenticate itself as a legitimate IoT device. This can be modeled by the following game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

1) $\mathcal{C}$ selects a legitimate IoT device ID_1 and initiates the proposed protocol between IoT device ID_1 and the server.
2) $\mathcal{A}$ calls `Send`S, `Send`ID, `Monitor`, and `Drop` a polynomial number of times on the server and the IoT device ID_1.
3) $\mathcal{A}$ invokes the `Send`S oracle to authenticate itself as a legitimate IoT device to the server.
4) $\mathcal{A}$ wins the game if he/she can successfully complete the authentication phase of the proposed protocol.

When $\mathcal{A}$ requests authentication, he/she needs to respond to the server by sending an authentication parameter $I_2 = H(PID_1 \parallel PID_1^{i+1} \parallel k_i)$. However, to generate k_i, he/she needs R^i to decrypt $\{N_a, N_b\}_{R^i}$ sent by the server. Similarly, to generate $PID_1^{i+1} = H(ID_1 \parallel N_a \parallel R^i)$, $\mathcal{A}$ needs R^i.

However, By Lemma 1, $\mathcal{A}$ cannot obtain the secret response R^i and is left with only one option, i.e., randomly guess R'. The advantage of the adversary for successfully authenticating itself and winning the game can be modeled as $Adv_{\mathcal{A}}^{Auth1} = 2 \times (\Pr[R' = R^i] - \frac{1}{2})$. An adversary has no advantage if he/she makes a random guess, i.e., $\Pr[R' = R^i] = \frac{1}{2}$. Thus, in the proposed protocols, an adversary has zero advantage ($Adv_{\mathcal{A}}^{Auth1} = 0$) and he/she cannot successfully authenticate to the server.

The second case is when the adversary $\mathcal{A}$ tries to authenticate itself to an IoT device as a server. This can be modeled by a game which is similar to the above game except that in step 3, instead of calling the Send$S$ oracle, $\mathcal{A}$ calls SendID to impersonate as the legitimate server. To act as the legitimate server, $\mathcal{A}$ needs to respond to IoT device ID_1's authentication request with the parameter $I_1 = H(M_2 \parallel N_a \parallel N_b \parallel R^i)$. However, to obtain N_a, $\mathcal{A}$ needs R^i to decrypt $\{N_a\}_{R^i}$ sent by the IoT device ID_1. The adversary's advantage can be modeled as $Adv_{\mathcal{A}}^{Auth2} = 2 \times (\Pr[R' = R^i] - \frac{1}{2})$. As $\mathcal{A}$ cannot expose R^i by Lemma 1, therefore, $\Pr[R' = R^i] = \frac{1}{2}$ and he/she will fail to to produce the valid response, i.e., $Adv_{\mathcal{A}}^{Auth2} = 0$. Thus, $\mathcal{A}$ cannot impersonate as the server.

This shows that the proposed protocols can successfully achieve mutual authentication. □

Theorem 5. *Data Provenance: The proposed protocols successfully establish data provenance, i.e., the origin of the data is authentic.*

Proof. The adversary $\mathcal{A}$ may try to impersonate as a legitimate IoT device and send tampered data to the server. This can be modeled by the following security game between a challenger $\mathcal{C}$ and adversary $\mathcal{A}$.

1) $\mathcal{C}$ selects an IoT device ID_1 and runs the proposed protocol with IoT device ID_1 and the server.
2) $\mathcal{A}$ calls Send$S$, SendID, Monitor, and Drop a polynomial number of times on the server and the IoT device ID_1.
3) $\mathcal{A}$ calls the Send$S$ oracle to impersonate an IoT device.
4) If $\mathcal{A}$ can successfully authenticate the tampered data sent by it to the server during the data transfer phase of the proposed protocol, then $\mathcal{A}$ wins the game.

To prove his/her legitimacy and tamper with the data transferred in the data transfer phase of the proposed protocol, $\mathcal{A}$ has to produce a valid pseudonym identity PID_1^i along with a valid authentication parameter $V_1 = H(D_1, R^i)$. For that, it should be able to reveal the PUF response R^i and the valid wireless fingerprint (as D_1 is the wireless fingerprint encrypted with R^i) for the IoT device ID_1. However, by Lemmas 1 and 3, the adversary $\mathcal{A}$ cannot achieve this. This shows that the proposed protocol can successfully ensure the provenance of the data transferred by an IoT device. □

Theorem 6. *Privacy: In the proposed protocols the IoT devices are untraceable.*

Proof. In the proposed protocols, the IoT devices are said to be untraceable if an adversary $\mathcal{A}$ cannot correlate two successful runs of a protocol with the server by the same IoT device. We consider the following security game between a challenger $\mathcal{C}$ and the adversary $\mathcal{A}$. We assume that both $\mathcal{A}$ and $\mathcal{C}$ can only run polynomial time algorithms.

1) $\mathcal{C}$ selects two IoT devices ID_1 and ID_2 and uses each IoT device to run the proposed protocol with the server.
2) $\mathcal{A}$ calls Send$S$, SendID, Monitor, and Drop a polynomial number of times on the server and the IoT devices. $\mathcal{A}$ notifies $\mathcal{C}$ after it is done.
3) $\mathcal{C}$ randomly picks one of the IoT devices $ID*$.
4) $\mathcal{A}$ calls Send$S$, SendID, Monitor, and Drop a polynomial number of times on the server and IoT device $ID*$.
5) $\mathcal{A}$ reveals her/his guess ID'.
6) $\mathcal{A}$ wins the game if $ID' = ID*$.

The advantage of the adversary for successfully guessing ID' and winning the game can be modeled as $Adv_{\mathcal{A}}^{Pri} = 2 \times (\Pr[ID' = ID*] - \frac{1}{2})$. An adversary has no advantage if he/she makes a random guess, i.e., $\Pr[ID' = ID*] = \frac{1}{2}$.

By Lemma 1, the adversary cannot obtain R^i and eventually cannot generate a valid pseudonym identity PID_{ID}^{i+1} for the next round. Similarly, he/she will fail to correlate the pseudonym identities to each other (by Lemma 2). Therefore, the adversary is left with only one option, i.e., random guess and accordingly the adversary as zero advantage ($Adv_{\mathcal{A}}^{Pri} = 0$). This shows that the proposed protocols can guarantee privacy and untraceability. □

Theorem 7. *The proposed protocols provide forward secrecy with backward untraceability support.*

Proof. A protocol is said to have forward secrecy with backward untraceablility if an adversary $\mathcal{A}$ cannot attach a protocol session with an IoT device even after calling the Reveal oracle on the IoT device. We can model this with a security game as follows:

1) $\mathcal{C}$ selects two IoT devices ID_1 and ID_2.
2) $\mathcal{A}$ calls Send$S$, SendID, Monitor, and Drop a polynomial number of times on the server and the IoT devices. $\mathcal{A}$ notifies $\mathcal{C}$ after it is done.
3) $\mathcal{C}$ uses each IoT device to successfully run the proposed protocol with the server. $\mathcal{C}$ then randomly picks one of the IoT devices $ID*$ and hands it over to the adversary $\mathcal{A}$.
4) $\mathcal{A}$ invokes Reveal oracle on IoT device $ID*$.
5) $\mathcal{A}$ reveals her/his guess ID'.
6) $\mathcal{A}$ wins the game if $ID' = ID*$.

The advantage of the adversary for winning the game can be modeled as $Adv_{\mathcal{A}}^{Sec} = 2 \times (\Pr[ID' = ID*] - \frac{1}{2})$. An adversary has no advantage if he/she makes a random guess, i.e., $\Pr[ID' = ID*] = \frac{1}{2}$.

When $\mathcal{A}$ invokes the Reveal oracle on the IoT device $ID*$, it obtains the pseudonym identity and challenge stored in the device's memory. However, note that the current pseudonym identity is generated using the IoT device identity, a random secret nonce N_a, and the secret PUF response R^i. By Lemma 1, $\mathcal{A}$ cannot extract R^i even after calling the Reveal oracle. Since N_a is encrypted using R^i during the previous session, therefore, $\mathcal{A}$ cannot obtain N_a. Moreover, a new N_a is

generated in each session which is independent of the previous session. This shows that the adversary $\mathcal{A}$ cannot trace a session back to an IoT device even if he/she has access to the stored contents of a device's memory. $\mathcal{A}$ can only use a random guess, i.e., $Adv_{\mathcal{A}}^{Sec} = 0$. Thus, the proposed protocols are successful in maintaining forward secrecy with backward untraceability. □

Lemma 8. *The proposed protocols are secure against DoS attacks.*

Proof. In a DoS attack, an adversary $\mathcal{A}$ attempts to break the synchronization between an IoT device and the server by dropping/blocking packets carrying important parameters. In the proposed protocols, $\mathcal{A}$ can attempt a DoS attack in two ways; First, he may call the `Drop` oracle on Message 3 sent by the IoT device, i.e., the server cannot receive the acknowledgment of Message 3 which will result in the server not updating the pseudonym identity of the IoT device. However, to tackle such a situation, the IoT device stores a set of emergency identities. Thus, in the next round when the IoT device tries to authenticate itself using the current pseudonym identity, the server will reject the request. In turn, the IoT device can use one of its emergency identities to authenticate itself and establish a new pseudonym identity.

The second way in which $\mathcal{A}$ can launch a DoS attack is by calling the `Drop` oracle on Message 2 in the CRP update protocol. However, to deal with such a situation the proposed protocol maintains a list of emergency challenges at the IoT device. Therefore, if a request by the IoT device is rejected by the server after the CRP update protocol, then the IoT device will use one of its emergency identities along with an emergency challenge to authenticate itself. Thus, the proposed protocols can successfully block DoS attacks. □

Lemma 9. *The proposed protocols are safe against physical and cloning attacks.*

Proof. Using physical attacks, the objective of an adversary is to extract secret keys from an IoT device's memory. However, as shown in Lemma 1, the proposed protocols do not store any secret keys in the IoT device's memory. Moreover, with the PUF and the micro-controller being a SoC, an adversary can neither separate the PUF from the micro-controller nor can it eavesdrop on the communication between the PUF and the micro-controller [39], [40]. Thus, we can infer that the proposed protocols are resilient against physical attacks. In contrast, most of the existing security protocols for IoT use some kind of stored secret key at the IoT device and are vulnerable to physical attacks.

IoT devices are usually installed in remote locations which exposes them to cloning attacks. According to the unclonability assumption for PUFs, it is extremely difficulty and even impossible to clone a PUF or predict its behavior [2], [41]. The proposed protocols are based on PUFs, i.e., each IoT device is equipped with its own unique PUF. Therefore, we can infer that the proposed protocols are safe against cloning. □

VIII. Formal Proof of Security Using BAN Logic

In this section we present a formal security analysis of the proposed protocols for data provenance using the Mao and Boyd logic [42]. The Mao and Boyd logic is an extension to improve the BAN logic [43]. The security analysis of protocols using formal logical approaches is critical to ensure an adversary cannot obtain or alter vital information. For ease of notation, we use A to represent the IoT device requesting authentication in the proposed protocols, i.e., ID_A in the single hop protocol and ID_1 in the multi-hop protocol. Similarly, S is used to denote the secure server. We prove the authentication properties as well as secrecy of N_a, N_b, and R^{i+1}.

The idealized messages for the proposed protocols in the authentication phase are given below. Note that the single hop as well as multi-hop protocols have the same set of idealized messages for authentication.

1) $A \rightarrow S : A, \{N_a\}_{R^i}$.
2) $S \rightarrow A : A, \{N_a, N_b\}_{R^i}$.
3) $A \rightarrow S : \{A, N_a, N_b\}_{R^i}$.

The initial beliefs/assumptions for the authentication phase are as follows:

1) $A \models A \overset{R^i}{\leftrightarrow} S$ and $S \models A \overset{R^i}{\leftrightarrow} S$.
2) $A \models S^c \triangleleft \parallel N_a$ and $S \models A \models \{S\}^c \triangleleft \parallel N_a$.
3) $A \models S \models \{A\}^c \triangleleft \parallel N_b$ and $S \models A^c \triangleleft \parallel N_b$.
4) $A \models \#(N_a)$ and $S \models \#(N_b)$.
5) $A \models sup(S)$ and $S \models sup(A)$.
6) $A \overset{R^i}{\triangleleft} N_a$ **R** N_b.
7) $S \overset{R^i}{\triangleleft} N_a$ **R** N_b.
8) $S \overset{R^i}{\sim} N_b$ and $A \overset{R^i}{\sim} N_a$.

The authentication properties for the proposed protocol are established using the tableau in Figure 7. The set of inference rules for the Mao and Boyd logic can be found in the Appendix. To prove the authentication of A to S, we need to infer $S \models A \overset{R^i}{\sim} N_a$, i.e., S believes that A sent N_a using R^i as the encryption key. To establish this, the authentication rule from [42] is applied and we need to prove $A \models A \overset{R^i}{\leftrightarrow} S$, i.e., R^i is a good shared secret between A and S and that $S \overset{R^i}{\triangleleft} N_a$, i.e., S decrypted N_a using R^i as the key. Since both of these statements are part of the initial assumptions/beliefs, the authentication of the IoT device to the server is established from Figure 7(a). Similarly, to prove that A and S successfully establish a symmetric key during the authentication phase, we need to prove the secrecy of N_a and N_b. This is shown in the tableaux in Figure 8. Thus, these proofs show that an adversary cannot obtain the secrets N_a and N_b used to create the session key k_i. Therefore, we can infer that k_i is a good secret key between the IoT device and the server.

To prove the syntactic secrecy of R^{i+1} during the CRP update phase of the proposed protocol, we idealize the messages of the protocol shown in Figure 6 as follows:

1) $S \rightarrow A : \{A, N_1\}_{R^i}$.
2) $A \rightarrow S : \{A, R^{i+1}\mathbf{R}N_1\mathbf{R}N_2\}_{R^i}$.

$$\frac{A \models A \overset{R^i}{\leftrightarrow} S \wedge S \overset{R^i}{\triangleleft} N_a}{S \models A \overset{R^i}{\sim} N_a}$$

(a) "**S** believes **A** sent N_a using R^i as the encryption key". This proves authentication of A to S.

$$\frac{A \models A \overset{R^i}{\leftrightarrow} S \wedge A \overset{R^i}{\triangleleft} N_b}{A \models S \overset{R^i}{\sim} N_b}$$

(b) "**A** believes **S** sent N_b using R^i as the encryption key". This proves authentication of S to A.

Fig. 7: Security proofs for authentication.

The set of initial beliefs/assumptions for this protocol are given as follows:

1) $A \models A \overset{R^i}{\leftrightarrow} S$ and $S \models A \overset{R^i}{\leftrightarrow} S$.
2) $A \models S^c \triangleleft \parallel R^{i+1}$ and $S \models A \models \{S\}^c \triangleleft \| R^{i+1}$.
3) $A \models \#(N_2)$ and $S \models \#(N_1)$.
4) $S \models sup(A)$.
5) $S \overset{R^i}{\triangleleft} N_1$ **R** R^{i+1}.
6) $A \overset{R^i}{\sim} R^{i+1}$.

The secrecy of R^{i+1} is established using the tableaux in Figure 9.

IX. Experimental Validation

In this section we present the experimental results and compare our technique's accuracy with the state of the art existing work in [36]. The authors of [36] proposed the use of correlation of the wireless fingerprints between two entities to establish data provenance.

MICA-Z motes, running TinyOS, and operating in the 2.4 GHz band were used for these experiments. MICA-Z motes use the CC2420 transceiver with the IEEE 802.15/Zigbee wireless communication protocol. The MICA-Z motes can output an 8-bit signed 2's complement RSSI value which can be converted to the received power (in dBm) using the following equation [44]

$$P_r = RSSI - 45. \tag{2}$$

The experiments were conducted in an indoor laboratory environment with typical furniture and WiFi equipment. The layout of the experiment is depicted in Figure 10. In this setup we have a legitimate IoT device, a base station and two adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. Note that the adversaries are located atlease one wavelength away from the legitimate IoT device.

Our experiments included two scenarios: *High Mobility* where the IoT device is moved around within the lab space to different locations, and *Low Mobility* where the IoT device is occasionally moved smaller distances within a small area. For each scenario, we gather the RSSI values between the legitimate IoT device and base station; and the adversaries and the base station for a period of one hour at a packet rate of 1 packet per 2 seconds. The resulting traces were analyzed offline with Matlab.

An empirical study of the *MSE* values shows a clear separation between the *MSE* values of a legitimate channel and adversarial channels, as shown in Figure 11 for 16-byte wireless fingerprints. The *MSE* values for adversarial channels are orders of magnitude larger than those for the legitimate channel. Therefore, to fit the large *MSE* values into Figure 11, we show the results for *MSE* on the `log` scale. We observe that the *MSE* value for adversarial channels is greater than 15 ($\texttt{log}_{10}15 = 1.1761$). Therefore, an *MSE* value of 15 can be used as the threshold to detect attacks. To compare [36] with our technique, we can compute the Pearson correlation coefficient r for channel measurements as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2} \cdot \sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{3}$$

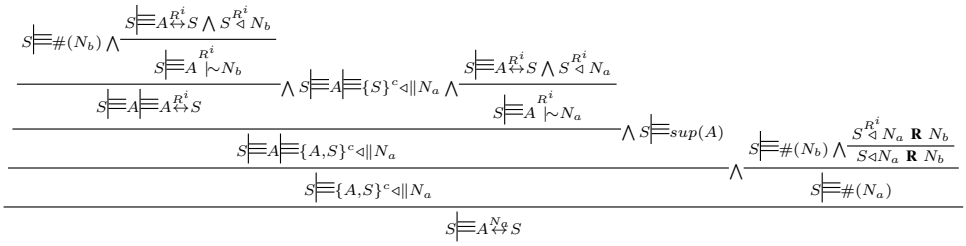
where X_i and Y_i represent the RSSI values for the *i*-th packet at each entity and $\bar{X}$ and $\bar{Y}$ are the corresponding mean RSSI values of a sequence of n packets. Furthermore, the threshold value for r is taken as 0.9 [36]. The proposed technique is compared to [36] using the probability of false alarm (i.e., the probability that a technique classifies a legitimate channel as an adversarial channel), and the probability of missed detection (i.e., the probability of a technique classifying an adversarial channel as a legitimate channel).

We compare the two techniques using different sizes for the wireless fingerprints, i.e., 16, 32, and 64 bytes. Note that to get comparable accuracy in [36], the link fingerprints using raw RSSI values must be 2392 bytes long. The authors of [36] propose the use of quantization to reduce the fingerprint size. However, to accurately detect attacks, the link fingerprint sizes for the low mobility and high mobility scenarios must still be atleast 24 and 54 bytes, respectively, in [36]. Using our technique, we show that it is possible to use smaller wireless fingerprints obtained from raw RSSI values to effectively detect the attacks. This results in a simpler technique without the need for quantization.

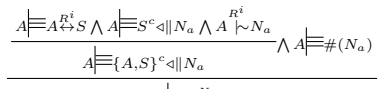
The results for the low mobility and high mobility scenarios are shown in Tables I and II, respectively. In these tables P_{FA} represents the probability of false alarm for the channel between the legitimate IoT device and base station. Similarly, P_{MD_1} and P_{MD_2} represent the probability of missed detection for the channel between $\mathcal{A}_1$ and base station, and $\mathcal{A}_2$ and base station, respectively. These results show that the proposed technique clearly outperforms the technique in [36].

X. Performance Analysis

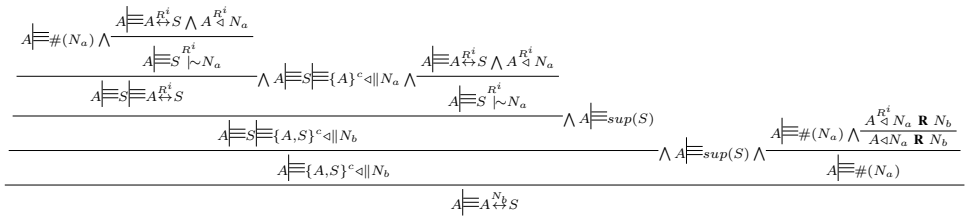
In this section we present the performance analysis of the proposed protocols. Although the existing work on protocols for data provenance in IoT is not extensive, we compare the performance of the proposed protocols with one of the most recent privacy preserving data provenance protocols for IoT by Sanchez et al. [8]. The protocol in [8] employs the state of the art non-interactive zero-knowledge proofs (NI-ZKP) technique to establish data provenance. Note that the authors of [36] do not provide a full protocol with authentication,



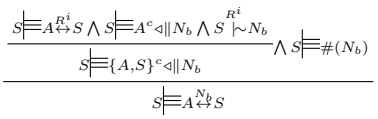
(a) Proof of "**S** believes N_a is a good shared key of **A** and **S**".



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

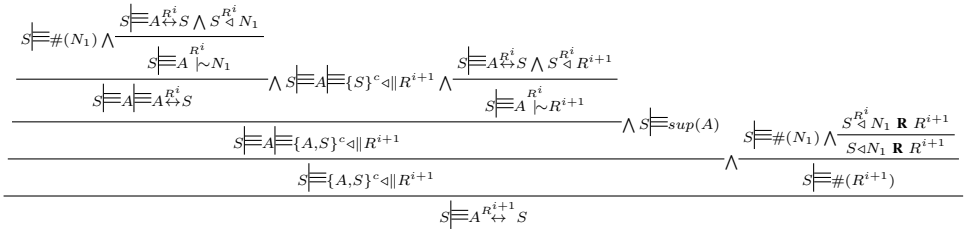


(c) Proof of "**A** believes N_b is a good shared key of **A** and **S**".

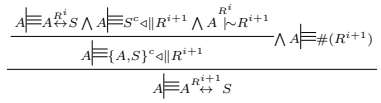


(d) Proof of "**S** believes N_b is a good shared key of **A** and **S**".

Fig. 8: Security proofs for secrecy.



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



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

Fig. 9: Security proofs for CRP update protocol.

TABLE I: Miss-classification rates for proposed protocol and reference [36] - low mobility.

Fingerprint Size	P_{FA} (%)			P_{MD_1} (%)			P_{MD_2} (%)		
	Proposed	[36]	% Improvement	Proposed	[36]	% Improvement	Proposed	[36]	% Improvement
16	16.7	78.6	79	0	11.1	100	0	12.5	100
32	5	55	91	0	5.3	100	0	11.8	100
64	0	11.1	100	0	2.6	100	0	0	0

privacy preservation, and data integrity. Therefore, [36] is not considered in this section.

A. Computational Complexity

We use N_H, N_{ENC}, N_{MAC}, $N_\times$, and N_{exp} to denote the number of hash, encryption/decryption, modular multiplication, and modular exponentiation operations, respectively. Moreover, we denote the number of intermediate devices in case of the multi-hop protocol by n_{Int}. The number of each of these operations for the proposed protocol and the protocol in [8] are shown in Table III.

We use universal hashing and block ciphers for encryption/decryption. This results in a worst case running time of $O(n)$ for hash, MAC, and encryption/decryption operations [45], [46]. Thus, the worst case running time at the IoT device as well as the server for the proposed single hop, multi-hop, and CRP update protocols is $O(n)$ (for a message size of n). On the other hand, the protocol proposed in [8] depends on modular multiplication and exponentiation which are considered computationally expensive operations. The complexity of modular multiplication is given by $O(M(l))$ which is a quadratic function of l-bit operands. Thus, for an exponent k, this results in a worst case running time of $O(n + M(l)k)$ at the IoT device as well as the server for the protocol in [8]. This shows that the proposed protocols have significantly lower computational complexity.

TABLE II: Miss-classification rates for proposed protocol and reference [36] - high mobility.

Fingerprint Size	P_{FA} (%)			P_{MD_1} (%)			P_{MD_2} (%)		
	Proposed	[36]	% Improvement	Proposed	[36]	% Improvement	Proposed	[36]	% Improvement
16	3.8	34.6	89	3.7	7.7	52	0	33.3	100
32	0	20	100	0	0	0	0	25	100
64	0	8.3	100	0	0	0	0	0	0

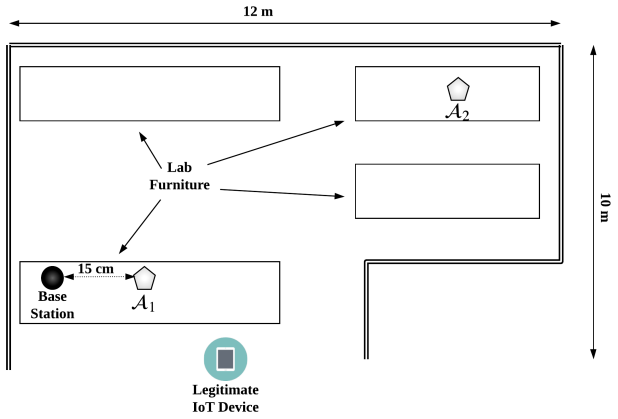
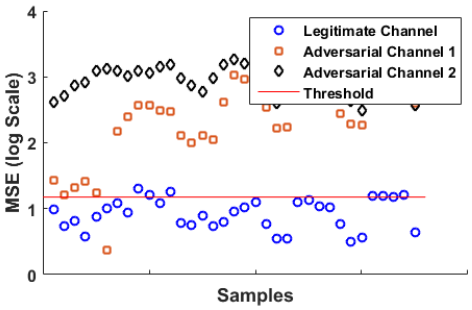
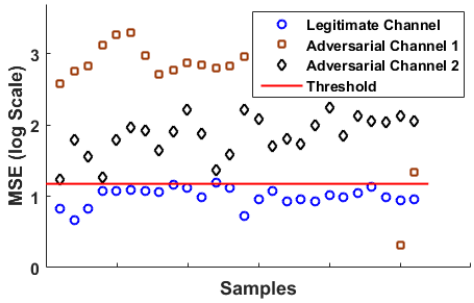


Fig. 10: Experiment Layout.



(a) Low Mobility



(b) High Mobility

Fig. 11: Comparison of the mean squared error for legitimate channel and adversarial channels.

B. Energy Requirements

In this section we analyze the energy requirements of the proposed protocols. We also compare our results with the protocol proposed in [8]. The protocols were emulated on the MICA 2 mote platform. The energy requirements were evaluated using the AVRORA energy analysis tool. The results presented here depict the average energy requirements for the CPU and radio subsystems after running the protocols a 100 times. Furthermore, the energy requirements for the protocols are reported for three different key sizes: 128, 192, and 256 bits. The wireless fingerprint size for the proposed protocols is taken as 64 bytes, i.e., the smallest size at which the probability of miss-classification is null. The energy associated with the CPU and radio subsystems includes the security sub-system, and other tasks such as boot, idle state, etc.

TABLE III: Computational Complexity.

Task	Source IoT Device	Intermediate IoT Device	Server
Single hop Protocol	$6N_H + 3N_{ENC}$	–	$6N_H + 4N_{ENC}$
Multi-hop Protocol	$6N_H + 3N_{ENC}$	$2N_H + 1N_{ENC}$	$(6 + n_{Int})N_H + (4 + n_{Int})N_{ENC}$
CRP Update	$2N_{MAC} + 2N_{ENC}$	–	$2N_{MAC} + 2N_{ENC}$
[8]	$3N_H + 2N_{exp} + N_\times$	–	$1N_H + 3N_{exp}$

The energy consumed by the CPU and the radio subsystems for the proposed protocols and the protocol in [8]. for the three key sizes is shown in Table IV. It is observed that for a key size of 256 bits, the CPU consumes 83.8% less energy while the radio consumes 73.5% less energy in the proposed protocol. The reduction in the CPU energy consumption comes from the lower computational complexity of the proposed protocols as discussed in Section X-A. Moreover, the reduction in the radio subsystem energy can be attributed to the fact that due to the higher computational complexity of [8], the radio needs to stay active for a longer period of time. This results in higher energy consumption, as obvious from the higher energy numbers shown in the other tasks column for [8]. Similarly, an increase of 64 bits in the key size results in an increase of 45,347 μJ and 57,676 μJ of energy by the CPU and radio, respectively, for [8]. However, an increase of 64 bits in key size results in a mere increase of approximately 110 μJ and 85 μJ in the CPU and radio energy consumption, respectively, for the proposed protocol. The total energy consumed by the CPU and radio subsystems for the proposed protocols and the protocol in [8] are shown in Figure 12. We observe that the energy consumption for the proposed protocols is significantly lower.

TABLE IV: Energy Consumption

Key Size	Protocol Proposed by [8]			Proposed Single Hop /Multi-Hop Protocol			Total Improvement %
	Protocol μJ	Other tasks μJ	Total μJ	Protocol μJ	Other tasks μJ	Total μJ	
CPU							
128-bits	47,653.26	30,131.76	77,785.03	508.70	26,490.87	26,999.56	12.1
192-bits	87,656.83	33,694.50	121,351.34	618.70	26,491.56	27,110.26	77.6
256-bits	131,223.14	37,257.24	168,480.38	728.71	26,492.25	27,220.95	83.8
Radio							
128-bits	1,924.85	82,036.25	83,961.10	2,993.59	49,634.36	52,627.95	59.5
192-bits	2,488.31	139,148.71	141,637.01	3,560.59	49,152.59	52,713.19	62.8
256-bits	305,176.40	196,261.16	199,312.92	4,127.59	48,670.83	52,798.42	73.5

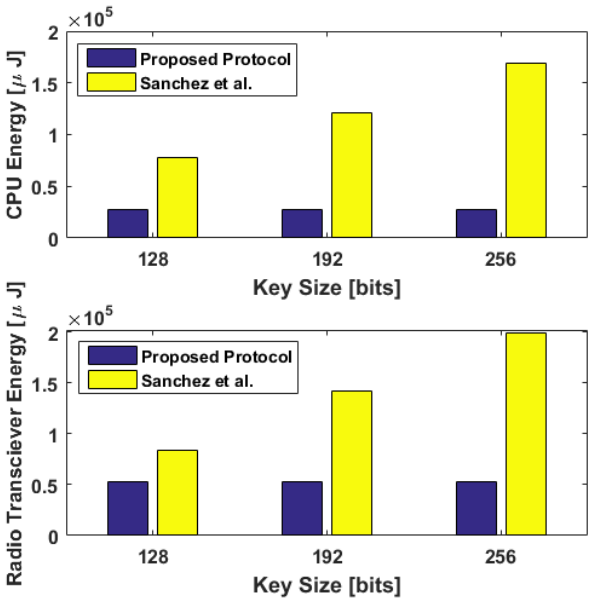


Fig. 12: Energy Consumed

XI. Conclusions

In this paper, security protocols providing mutual authentication, privacy, and data provenance in IoT systems have been introduced for two different scenarios: (i) when an IoT device is directly connected to a wireless gateway, and (ii) for the case when an IoT device is indirectly connected to the gateway through other IoT devices. The proposed protocols are based on PUFs to provide protection against physical and cloning attacks. The proposed protocols use wireless link fingerprints with a threshold mechanism based on mean squared error to establish data provenance. Experimental validation of this technique and comparison with state-of-the-art using real deployment on MICA-Z motes has shown that the proposed technique can significantly reduce the miss-classification rates of detecting attacks on data provenance. A formal analysis of the proposed protocols has been used to assess their security. Finally, emulation of the proposed protocols on the popular MICA 2 mote shows a significant reduction in the energy requirements for the proposed protocols as compared to existing state-of-the-art protocols.

References

[1] D. Evans, "The internet of things how the next evolution of the internet is changing everything," https://www.cisco.com/c/dam/en us/about/ac79/docs/innov/IoT IBSG 0411FINAL.pdf, Apr 2011, accessed: 2018-10-01.

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

[3] V. Varadharajan, S. Bansal, "Data Security and Privacy in the Internet of Things (IoT) Environment," in: Z. Mahmood (eds) *Connectivity Frameworks for Smart Devices, Computer Communications and Networks*. Springer, Cham, 2016.

[4] S. Suhail et al., "Introducing Secure Provenance in IoT: Requirements and Challenges," in *Proc. International Workshop on Secure Internet of Things (SIoT)*, Heraklion, 2016, pp. 39-46.

[5] E. Nwafor et al., "Towards a provenance collection framework for Internet of Things devices," in *Proc. IEEE SmartWorld,* San Francisco, CA, 2017, pp. 1-6.

[6] M. Elkhodr, B. Alsinglawi and M. Alshehri, "Data Provenance in the Internet of Things," in *Proc. WAINA,* Krakow, 2018, pp. 727-731.

[7] S. Suhail et al., "Data trustworthiness in IoT," in *Proc. ICOIN,* Chiang Mai, 2018, pp. 414-419.

[8] J. L. C. Sanchez, J. B. Bernabe and A. F. Skarmeta, "Towards privacy preserving data provenance for the Internet of Things," in *IEEE WF-IoT,* Singapore, 2018, pp. 41-46.

[9] U. Chatterjee et al., "A PUF-Based Secure Communication Protocol for IoT," in *ACM Trans. Embedded Comput. Sys.*, vol. 16, no. 3, pp. 67-91 ,Jul. 2017.

[10] U. Chatterjee et al., "Building PUF based Authentication and Key Exchange Protocol for IoT without Explicit CRPs in Verifier Database," in *IEEE Trans. Dependable and Secure Comput.*, preprint, May 2018.

[11] M. N. Aman, K. C. Chua and B. Sikdar, "Mutual Authentication in IoT Systems Using Physical Unclonable Functions," in *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1327-1340, Oct. 2017.

[12] A. Kanuparthi et. al., "Hardware and Embedded Security in the Context of Internet of Things," *Proc. ACM CyCAR,* November 2013.

[13] M. N. Aman, et. al. "Physical Unclonable Functions for IoT Security." *In Proc. ACM IoTPTS*, New York, NY, USA, 10-13, 2016.

[14] P. Tuyls and L. Batina, "RFID-Tags for Anti-Counterfeiting," *In Proc. CT-RSA),* vol. 3860 of LNCS, pp. 115-131, Springer Verlag, February 2005.

[15] Y. Dodis et al., " Fuzzy extractors: How to generate strong kesy from biometrics and other noisy data." *In Advances in Cryptology - EUROCRYPT '2004,* Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, 2004.

[16] J. W. Lee et. al., "A technique to build a secret key in integrated circuits for identification and authentication applications," *2004 Symp. on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525),* 2004, pp. 176-179.

[17] R. Maes, *Physically Unclonable Functions: Construction, Properties and Applications* (Springer, London, 2013).

[18] M. Aman, K. C. Chua and B. Sikdar, "Mutual Authentication in IoT Systems using Physical Unclonable Functions," *IEEE Internet of Things Journal,* vol. 4, no. 5, pp. 1327-1340, October 2017.

[19] B. Gassend et. al., "Silicon physical random functions." *In Proc. ACM CCS*, Vijay Atluri (Ed.), New York, NY, USA, 148-160, 2002.

[20] W. C. Jakes. "Microwave Mobile Communications". Wiley, 1974.

[21] Z. Li et al., "Securing wireless systems via lower layer enforcements," in *Proc. ACM WiSe,* Los Angeles, CA, 2006, pp. 33-42

[22] R. Wilson, D. Tse and R. A. Scholtz, "Channel Identification: Secret Sharing Using Reciprocity in Ultrawideband Channels," in *IEEE Trans. Inform. Forensics Sec.*, vol. 2, no. 3, pp. 364-375, Sept. 2007.

[23] S. N. Premnath et al., "Secret key extraction using Bluetooth wireless signal strength measurements," in Proc. IEEE SECON, Singapore, 2014, pp. 293-301.

[24] S. Mathur et al., "Radiotelepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel." In *Proc. ACM MobiCom,* San Fancisco, CA, 2008, pp. 128139.

[25] N. Patwari et al., "High-Rate Uncorrelated Bit Extraction for Shared Secret Key Generation from Channel Measurements," in *IEEE Trans. Mobile Comput.,* vol. 9, no. 1, pp. 17-30, Jan. 2010.

[26] S. N. Premnath et al., "Secret Key Extraction from Wireless Signal Strength in Real Environments," in *IEEE Trans. Mobile Comput.,* vol. 12, no. 5, pp. 917-930, May 2013.

[27] S. L. Cotton and W. G. Scanlon, "An experimental investigation into the influence of user state and environment on fading characteristics in wireless body area networks at 2.45 GHz," in *IEEE Trans. on Wireless Commun.,* vol. 8, no. 1, pp. 6-12, Jan. 2009.

[28] L. Shi et al., "BANA: Body Area Network Authentication Exploiting Channel Characteristics," in *IEEE J. Selected Areas in Commun.,* vol. 31, no. 9, pp. 1803-1816, September 2013.

[29] L. Shi et al., "ASK-BAN: Authenticated Secret Key Extraction Utilizing Channel Characteristics for Body Area Networks," in *Proc. ACM WiSec,* Hungary, 2013, pp. 155-166.

[30] S. T. Ali, V. Sivaraman, and D. Ostry, "Zero Reconciliation Secret Key Generation for Body-worn Health Monitoring Devices," in *Proc. ACM WiSec,* Tucson, AZ, 2012, pp. 3950.

[31] S. Mathur et al., "ProxiMate: Proximity-based Secure Pairing using Ambient Wireless Signals," in *Proc. ACM MobiSys*, Bethesda, MS, 2011.

[32] A. Kalamandeen et al., "Ensemble: Cooperative Proximity-based Authentication," in *Proc. ACM MobiSys*, San Francisco, CA, 2010, pp. 331344.

[33] J. Tang, P. Fan and X. Tang, "A RSSI-Based Cooperative Anomaly Detection Scheme for Wireless Sensor Networks," in *Proc. WiCom*, Shanghai, 2007, pp. 2783-2786.

[34] Y. Chen et al., "Detecting and Localizing Identity-Based Attacks in Wireless and Sensor Networks," in *IEEE Trans. Veh. Tech.*, vol. 59, no. 5, pp. 2418-2434, Jun 2010.

[35] J. Yang et al., "Detection and Localization of Multiple Spoofing Attackers in Wireless Networks," in *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 44-58, Jan. 2013.

[36] S. T. Ali et al., "Securing First-Hop Data Provenance for Bodyworn Devices Using Wireless Link Fingerprints," in *IEEE Trans. Inform. Forensics Sec.*, vol. 9, no. 12, pp. 2193-2204, Dec. 2014.

[37] M. Kamal and S. Tariq, "Light-Weight Security and Data Provenance for Multi-Hop Internet of Things," in *IEEE Access,* vol. 6, pp. 34439-34448, 2018.

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

[39] S. Guilley, and R. Pacalet, "SoCs security: a war against side-channels", *Annals of Telecommunications,* 59(7), pp. 998-1009, 2004.

[40] 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.

[41] M. Alioto (Ed.), *Enabling the Internet of Things from Integrated Circuits to Integrated Systems*, Springer, 2017.

[42] W. Mao and C. Boyd, "Towards formal analysis of security protocols", *Proc. Comp. Sec. Foundations Workshop VI,* pp. 147-158, 1993.

[43] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication", *ACM Trans. Comp. Sys.,* 8, February 1990.

[44] The MEMSIC solution: MICAz mote platform datasheet, http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf.

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

[46] Y. Mansour et. al., "The Computational Complexity of Universal Hashing," *Theoretical Computer Science,* vol. 107, no. 1, pp. 121-133, 1993.

APPENDIX

This section presents a brief introduction to the Mao and Boyd logic for the formal security proof of the proposed protocols [42]. Logical formulas are constructed uing messages M, principals P, and formulas F. Capital letters A, B, P, $Q, \cdots$ are used for principals, messages are denoted by K, M, $N, \cdots$, and formulas are represented by X, Y, $Z, \cdots$. The main constructs used in our analysis are as follows:

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

Following are the definitions used:

- **Atomic Message:** A message constructed without using any of the symbols ",", "|", "**R**", or "{}". Note that fields in a message are separated using "," while encryption is represented by"{}". The symbols "|" and "**R**" are defined below.
- **Challenge:** An atomic message excluding time stamps sent and received in separate messages by the same principal (the originator).
- **Replied Challenge:** A challenge sent back in a message to the originator.
- **Response:** A challenge and a replied challenge sent together in reply.
- **Nonsense:** An atomic message excluding timestamps, challenges, or responses.

Protocol message idealization is done using the following rules:

1) Discard any nonsense.
2) An atomic message acting as a challenge as well as a response in a line is considered as a response.
3) Merge challenges separated by commas using operator "|".
4) Merge responses separated by commas using operator "|".
5) Merge a challenge and its response using operator "**R**" i.e., "*response* **R** *replied challenge*".
6) Affix a timestamp to its message using operator "**R**", i.e., "*message* **R** *timestamp*".

The set of inference rules of Mao and Boyd logic used in the analysis of the proposed protocols are as follows:

1) **Authentication Rule**:

$$\frac{P \overset{K}{\models} P \overset{K}{\leftrightarrow} Q \bigwedge P \overset{K}{\triangleleft} M}{P \overset{K}{\models} Q \overset{K}{\sim} M}. \tag{4}$$

2) **Confidentiality Rule**:

$$\frac{P \overset{K}{\models} P \overset{K}{\leftrightarrow} Q \bigwedge P \overset{K}{\models} S^c \triangleleft \| M \bigwedge P \overset{K}{\sim} M}{P \overset{K}{\models} (S \cup \{Q\})^c \triangleleft \| M}. \tag{5}$$

3) **Super-Principal Rule**:

$$\frac{P \models Q \models X \wedge P \models sup(Q)}{P \models X}. \tag{6}$$

4) **The Fresh Rule**:

$$\frac{P \models \#(M) \wedge P \triangleleft N \mathbf{R} M}{P \models \#(N)}. \tag{7}$$

5) **The Good-Key Rule**:

$$\frac{P \models \{P,Q\}^{c} \triangleleft \| K \wedge P \models \#(K)}{P \models P \overset{K}{\leftrightarrow} Q} \tag{8}$$

6) **Intuitive Rule**:

$$\frac{P \overset{K}{\triangleleft} M}{P \triangleleft M}. \tag{9}$$

Muhammad Naveed Aman received the B.Sc. degree in Computer Systems Engineering from KPK UET, Peshawar, Pakistan, M.Sc. degree in Computer Engineering from the Center for Advanced Studies in Engineering, Islamabad, Pakistan, M.Engg. degree in Industrial and Management Engineering and Ph.D. in Electrical Engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA in 2006, 2008, and 2012 respectively. He is a recipient of the prestigious Fulbright Scholarship for Ph.D. studies in the USA.

He is currently working as a Senior Research Fellow with the Department of Computer Science at the National University of Singapore, Singapore. Dr. Aman previously served on the faculty of National University of Computer and Emerging Sciences Pakistan as an Assistant Professor. His research interests include IoT and network security, wireless and mobile networks, and secure embedded systems.

Mohammad Haroon Basheer (S18) is a Research Assistant with NUS-Singtel Cybersecurity Research & Development Laboratory since Nov 2017. He received his Bachelor of Technology in Electronic Engineering from National University of Singapore, where he is also pursuing his Master of Computing degree in Computer Science.

Biplab Sikdar (S98-M02-SM09) received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University,Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001. He was on the faculty of Rensselaer Polytechnic Institute from 2001 to 2013, first as an Assistant and then as an Associate Professor.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include wireless network, and security for IoT and cyber physical systems. Dr. Sikdar is a member of Eta Kappa Nu and Tau Beta Pi. He served as an Associate Editor for the *IEEE Transactions on Communications* from 2007 to 2012. He currently serves as an Associate Editor for the *IEEE Transactions on Mobile Computing*.