An EAP-Based Mutual Authentication Protocol for WLAN connected IoT devices

Awaneesh Kumar Yadav, Student Member, IEEE, Manoj Misra, Member, IEEE, Pradumn Kumar Pandey, Member, IEEE, Madhusanka Liyanage, Senior Member, IEEE

Abstract—Several symmetric and asymmetric encryption based authentication protocols have been developed for the Wireless Local Area Networks (WLANs). However, recent findings reveal that these protocols are either vulnerable to numerous attacks or computationally expensive. Considering the demerits of these protocols and the necessity to provide enhanced security, a lightweight Extensible Authentication Protocol (EAP)-based authentication protocol for WLAN-connected IoT devices is presented. We conduct an informal and formal security analysis to ensure robustness against the attacks. Furthermore, the empirical performance analysis and comparison show that the proposed protocol outperforms its counterparts, reducing computational, communication, storage costs, and energy consumption by up to 99%, 80%, 91.8%, and 98%, respectively. Simulation results of the protocol using the NS3 and its overhead under unknown attacks demonstrate that the proposed protocol performs better in all scenarios. A prototype implementation of the protocol has also been tested to evaluate its feasibility in real-time applications.

Keywords—Authentication, Extensible Authentication Protocol, Formal Verification, Network Security, WLAN.

I. INTRODUCTION

We are rapidly moving towards a smart world where almost everything will be digital. The Internet of Things (IoT) is the next phase of technological revolution which is rapidly evolving towards a smart world where the dependence of connected things on wireless and mobile technology will be inevitable. This is due to the fact that IoT applications such as Smart city, Health monitoring, Smart homes, Smart factories, Smart grid, Hospitality and Tourism in real life are changing the way we go about every societal function. With the recent influx of low-cost WLAN-capable smart IoT gadgets, our reliance on WLAN technology has grown even more [1] [2]. WLAN is widely regarded as an insecure public network because of open-air broadcasting. Any unknown user can intercept or access WLAN communication between communicating parties. As a result, the security of the WLAN (especially authentication) is a severe concern. To resolve this concern, a robust authentication method is required to prevent illegal network access and ensure that only authorized users have access to the network [3]. Authentication is a method of confirming an entity’s identity when accessing a resource [4], [5]. The WLAN security architecture is defined by IEEE 802.11i, which outlines the flexible key hierarchy and key exchange between the IoT Device (D) and the Authentication Server (AS). IEEE 802.11i employs IEEE 802.1x, a secure and reliable authentication framework for establishing a secure connection between the D and AS with the help of AP. The IEEE 802.1x architecture uses the EAP framework for a trustworthy base and message exchange [6], [7].

Extensible Authentication Protocol (EAP) is a framework for facilitating a variety of WLAN authentication techniques known as EAP methods, and RFC-3748 [8] contains a detailed description of the EAP framework. Several authentication methods that employ the EAP architecture have been developed and are commonly used in WLANs. However, all these existing protocols fail to protect from newly identified attacks, such as privileged insider attack, traceable attack, ephemeral secret leakage. Apart from that, most of the authentication protocol does not support the fast reconnect protocol for quick re-authentication, and all the symmetric-based authentication schemes require the secure channel during the registration phase. Therefore, there is a pressing need to design an authentication mechanism that protects from newly identified attacks, supports fast reconnect, eliminates the secure channel requirement, and is suitable for ultra-low-cost IoT devices.

A. Motivation and Contributions

The increasing use of IoT devices has necessitated the design of security mechanisms for IoT applications. Generally, IoT devices that require moderate bandwidth use WLAN for communication. However, security (notably authentication) continues to be a significant impediment to WLAN adoption. Therefore, to secure the communication between the D and AS, several symmetric and asymmetric encryption-based authentication protocols have been proposed, with the majority of them relying on the EAP architecture. Asymmetric encryption-based authentication protocols offer excellent security but come at a high cost, making them unsuitable for ultra-low-cost IoT devices [9]–[13]. In order to address the cost issue, several symmetric encryption-based authentication protocols are proposed. However, some recent findings [14], [15] reveal that although these protocols are lightweight but do not ensure the prominent security features such as perfect forward secrecy, identity protection, protection from traceability attack, privileged insider attack protection, ephemeral secret leakage, and many of them do not support fast reconnect for quick re-authentication. To the best of our knowledge, all
the symmetric encryption-based authentication protocols [9]–[13] need a secure channel during the registration process. However, this is only achievable in private premises such as smart homes, smart factories, smart firm etc., and finding a secure channel is infeasible in public places such as smart hospital, smart shops, etc.

This paper proposes a symmetric key-based authentication and key agreement protocol and shows that authentication and key agreement method relying solely on symmetric key-based operations can offer the same amount of security features as provided by asymmetric key-based methods and at a much lower cost. Further, the proposed method does not need a secure channel during the registration process, which was only possible with the public key-based protocols till date.

The key contributions of this paper are as follows
1) We design a symmetric key-based authentication method that provides the same level of security as public key-based authentication and key agreement protocols.
2) The proposed method removes the necessity of a secure channel during the registration phase. To the best of our knowledge, this is the first symmetric encryption-based protocol that does not require a secure channel at the time of registration. This feature may be essential for the services such as smart hospitals, smart shops, smart transport etc., where users may have to register in the absence of a secure channel.
3) The informal and formal (i.e., BAN logic, and Scyther tool) security analysis are conducted to confirm that the proposed protocol offers all the identified security features and securely generates the secret parameters.
4) The empirical performance analysis and comparison demonstrate that the proposed protocol outperforms its counterparts in terms of computational, communication, storage costs, and energy consumption. Furthermore, we compute the overhead under unknown attacks and do simulations using the NS3 tool, which shows that the proposed protocol performs better in all parameters.
5) A prototype implementation of the proposed protocol is done to show its feasibility in real time application.

In Section II, we summarise the existing literature on authentication in WLAN, including the research gaps. Section III discusses the preliminaries and backgrounds used in the paper. Section IV presents the proposed protocol for mutual authentication. Furthermore, informal and formal security analysis of the proposed protocol is discussed in Section V and Section VI. The performance of the proposed protocol is shown in Section VII. Section VIII shows the prototype implementation followed by the conclusion in Section IX.

II. RELATED WORKS

The EAP framework has been used to create a variety of authentication methods. These protocols can be divided into two groups: a) EAP protocols based on certificates; b) EAP protocols based on strong passwords.

A. EAP protocols based on certificates

$D$ and $AS$ both utilise certificates to confirm their legitimacy in certificate-based EAP techniques. To establish a reliable authentication approach, EAP-TLS [16] method was presented. For authentication, this protocol uses certificates. However, it is computationally costly and necessitates a large number of message exchanges. As a result, resource constrained IoT devices cannot use this authentication approach.

EAP-TTLS [17] was developed in response to the constraints of EAP-TLS. Though it also uses certificates but unlike EAP-TLS, EAP-TTLS, only requires a server-side certificate rather than a client-side certificate. It, however, falls short of the cost-cutting goal. As an alternative N Cam-Winget [18] provided an authentication mechanism. When automatic PAC provisioning is enabled, it provides strong protection but fails to save cost and is unable to hide the credentials from the attacker. Shajoi et al. [19] proposed an authentication approach that improves the security of EAP-TLS while incurring a higher cost than EAP TLS. Pawan et al. [14] presented an authentication paper that establishes a connection using a combination of certificates and pre-assigned replies. Moriarty et al. [15] proposed an extended version of EAP-TLS to facilitates the identity protection.

B. EAP protocols based on strong passwords

In the strong password-based EAP approaches, $D$ and $AS$ convince each other that they know a secret without really disclosing it. Omar et al. [20] provided a user authentication strategy that also includes a mechanism for key creation, however their scheme lacks the ability to quickly reconnect. An authentication solution for IEEE 802.11 wireless LANs was presented by Younes et al. [21]. Their approach employs asymmetric public-key encryption and complies with all of the RFC-4017 specifications. Additional security needs, such as DoS attacks, perfect forward secrecy, and lightweight processing, are not met. In the WLAN context, Chan et al. [9] created a user authentication system. Though, it is lightweight but prone to replay attack. Amit et al. [11] proposed a technique that claims to relieve server’s burden while also meeting all security requirements. An authentication protocol was proposed by Pandey et al. [10]. The fast reconnect mechanism and key generation aren’t specified in their protocol. Biswanath et al. [12] presented an EAP authentication system for WLANs that uses dynamic keys. Awaneesh et al [13] proposed an authentication mechanism that ensures perfect forward secrecy and identity protection. To address these challenges, the elliptic curve cryptography (ECC) [3], [22], [23] based authentication are proposed in the literature. However, the scheme provides the protection from several type of attack excepts ephemeral secret leakage and is computationally high.

C. Research gaps in the existing authentication schemes

We observed the following flaws in the existing protocols.
• Secure channel assumption: None of the existing symmetric encryption based techniques [9]–[13], [15], [20], [21] assume insecure channel between $D$ and $AS$ during the registration phase.
• Protection from Traceable attack : All existing scheme [9]–[14], [16]–[21] fail to provide the protection from traceable attack.
III. Preliminaries and Background

The background used in the paper is discussed in this section.

A. Network model

WLAN is a wireless communication network that allows devices to access the network services in a specific range. It is commonly utilized because of its ease of installation. The user can wander throughout the region while staying connected to the WLAN [6]. Fig 1 represents the network model for IoT-WLAN that involves three entities:

![Diagram of Network Model for IoT-WLAN]

- **IoT devices (D):** require network access, such as a smartphone, smartwatch, or tablet.
- **Access Point (AP):** serves as a connection point between the device and the authentication server.
- **Authentication Server (AS):** operates as a backend server in charge of authenticating the device.

When any user or client wants to access the network using the IoT devices then first it need to establish a secure connection. To establish a secure connection, authentication is required between the device and the authentication server. During authentication, the device and the authentication server verify their authenticity; if they are confirmed to be genuine, the authentication server permits the device to connect to the network via a certain access point within a certain range.

B. Threat model

We use the widely established “Delev-Yao (DY) [24] and CK-adversary [25] threat model” to test the resilience of the developed protocol. In our threat model, the adversary has the following capabilities.

1) The adversary has complete control over the communication sent over the open wireless channels and can read, delete, or change the messages sent over the wireless channels. Adversary can also insert valid communications.

2) As it is a “computationally infeasible task” to guess multiple values at once, such as identity and password at the same time, the adversary can only guess one value in polynomial time.

3) Adversary has the ability to intercept messages from many sessions and launch a traceability attack.

4) Adversary has the ability to act as a middleman and launch a man-in-the-middle attack.

C. Design goals

The following are the security goals that the designed authentication technique must meet.

- **Mutual authentication:** It specifies that communicating parties (D and AS) must verify each other’s validity before transferring any confidential or personal information.
- **Identity protection:** To support identity protection, communicating parties’ identities should not be sent in plain text via an insecure public channel.
- **Perfect forward secrecy:** It assures that even if the attacker gains access to sensitive data of the device, it is impossible for an attacker to determine that two different authentication requests are sent by the same device.

IV. Proposed Protocol

This section presents an effective and robust authentication protocol for WLAN communication that overcomes the existing authentication protocols’ limitations and security flaws. There are the following three phases in the proposed protocol

1) **Registration phase:** During the registration phase, D and AS exchange their secrets using insecure channel.

2) **Mutual authentication phase:** With the help of AP, D and AS confirm their legitimacy and securely procure the session key for data confidentiality and integrity.

3) **Fast reconnect phase:** When D is detached from an access point due to a network fault and wishes to
reconnect with a frequently visited AP, it can quickly reconnect utilizing fast reconnect credentials without having to go through the entire authentication process.

Fig. 2: Flowchart for proposed protocol

A. Registration phase

To utilise AS’s services, D must first register by entering its identifier and password. This phase is carried out using an insecure public channel, and the steps are outlined below.

- D chooses identity UID, password PW and random number R1. Afterwards it computes \( D_1 = E_{K_m}^{'}(UID \parallel PW \parallel R_1) \) using public key \( K_m \) of AS and forwards \( D_1 \) to AS.
- Upon receiving \( D_1 \), AS decrypts \( D_{K_m}(D_1) \) using private key \( K_m \) and checks the database that UID exists. If it exits then AS notify to D to send another request with a different identifier otherwise AS selects key \( k, p \) random number \( R_2 \) and compute \( Z_{UID} = E_{K_s}(UID \parallel R_2) \). \( D_2 = E_{R_1}(k \parallel p \parallel SID \parallel Z_{UID}) \). It then sends \( D_2 \) and store the \( < PW,UID,SID,k,p > \) into his database.
- When D receives \( D_2 \) then it decrypts message \( D_{R_1}(D_2) \) and save credentials into his database in encrypted form \( J = E_{PW}(k,p,SID,Z_{UID}) \) using the PW.

B. Authentication phase

The mutual authentication procedure between D and AS is carried out during this phase which allows both parties to share a session key that will be used to encrypt future data sent across the network. We assume that the connection between D and AP is unsafe, whereas the connection between AP and AS is secure in our work. We assume that the clocks of D, AP and AS are synchronized as assumed by many other researchers [3] [4].

Table 1 summarises the symbols and abbreviations used in the paper. Fig.2 describes the flowchart of the proposed protocol, Algorithm-1 and Algorithm-2 describes the pseudo-code of the proposed authentication protocol and Fig.3 provides a detailed description of the authentication process.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Meanings</th>
</tr>
</thead>
<tbody>
<tr>
<td>D, UID</td>
<td>User device and identifier of device</td>
</tr>
<tr>
<td>AP, K_s</td>
<td>Access Point and Secret common key used by AS</td>
</tr>
<tr>
<td>AS, SID</td>
<td>Authentication server and identity of AS</td>
</tr>
<tr>
<td>k, p, L, p=SK</td>
<td>short-term key secret key</td>
</tr>
<tr>
<td>K_s, K_m</td>
<td>private key of AS and public key of AS</td>
</tr>
<tr>
<td>Z_{UID}, PW</td>
<td>Masked identity of device and Password</td>
</tr>
<tr>
<td>H, TK</td>
<td>Hash function and temporary key for fast reconnect new timestamp random numbers</td>
</tr>
</tbody>
</table>

- D \( \rightarrow \) AP: When D wants to access network services then it decrypts the \( D_{PW}(J) \) to extract the stored credentials \( (k,p,SID,Z_{UID}) \). Afterwards, it select timestamp \( T_1 \), random number \( r_1 \) and computes the \( CH = E_{k\oplus p}(UID \parallel T_1 \parallel r_1) \) and forwards \( < CH,T_1,Z_{UID} > \) to AP.
- AP \( \rightarrow \) AS: AP forwards this message \( < CH,T_1,Z_{UID} > \) to AS.
- AS \( \rightarrow \) AP: Upon receiving the message \( < CH,T_1,Z_{UID} > \), AS gets time-stamp \( T_2 \) to verify the freshness of received message by checking the freshness condition \( (T_2 - T_1 < T) \). Afterwards AS decrypts \( D_{K_s}(Z_{UID}) \) using secret key \( K_s \) to get identifier UID and based on that it extract stored credentials \( (k,p,PW,SID) \) into his database. It then decrypts \( D_{k\oplus p}(CH) \) to obtain \( (UID,T_1,r_1) \) and compare \( (UID == UID*, T_1 == T_1) \), if matches then choose new key \( p_n \), random number \( r_2,R_3 \) and compute \( Z_{new UID} = E_{K_s}(UID \parallel R_3) \). \( RCH = E_{k\oplus p}(SID \parallel T_2 \parallel p_n \parallel r_2 \parallel T_1 \parallel Z_{new UID}) \). After computing RCH, it updates \( p \) by \( p_n \) and forwards \( < RCH,T_2 > \) to D.
- AP \( \rightarrow \) D: AP forwards this message \( < RCH,T_2 > \) to D.
- D \( \rightarrow \) AP: When D receives \( < RCH,T_2 > \), get time-stamp \( T_3 \) and verify the freshness of the received message by checking \( (T_3 - T_2 < T) \). If it matches then it decrypts \( D_{k\oplus p}(RCH) \) to obtain credentials \( (SID,T_2,p_n,r_2,r_1,Z_{newUID}) \) and compare \( (SID == SID*, r_1 == r_1*) \), if matches then D believes that AS is authentic and select new key \( L \), updates \( p \) by \( p_n \) to compute \( RES_1 = E_p(L \parallel T_1 \parallel r_2) \). \( RES_2 = E_p(L \parallel T_3 \parallel r_2) \). After computing \( RES_1, RES_2, D \) updated \( k \) by \( L \) and forwards \( < RES_1,RES_2,T_3 > \) to AP.
- AP \( \rightarrow \) AS: AP forwards this message \( < RES_1,RES_2,T_3 > \) to AS.
- AS \( \rightarrow \) AP: After receiving the messages \( < RES_1,RES_2,T_3 > \) from the AP, AS gets time-stamp \( T_4 \) and verify the freshness of the received message by checking \( (T_4 - T_3 < T) \). If it matches then decrypts (i.e., \( D_p(RES_2) \)) to obtain the \( L \). After getting \( L \), it decrypts \( D_{L}(RES_1) = (PW,T_3,r_2) \) and compare \( (PW == PW*, r_2 == r_2*) \), if it matches then it believe that D is authentic and selects a temporary new identity \( ID_{new} \) for D (i.e., D will use this identity during the fast reconnect authentication process), and lease time \( (LT) \) for the session key, (i.e., defines the temporary key expiry time and also ensures that this is unique for every D) and compute \( SK = H(r_1 \parallel r_2 \parallel PW \parallel SID) \). \( CH_F = E_{L\oplus p}(SK \parallel LT \parallel T_1 \parallel ID_{new} \parallel LT) \). After computing \( CH_F, AS \) updates \( k \) by \( L \), store \( < L,p > \) and forwards \( < CH_F,LT,ID_{new},T_4 > \) to the AP.
- AP \( \rightarrow \) D: When AP receives the message \( < CH_F,LT,ID_{new} > \), it saves (\( LT, ID_{new} \)) into its database and passes the \( < CH_F,T_4 > \) to D.
- After receiving the message \( < CH_F,T_4 > \) from AP, D selects the timestamp \( T_5 \) to verify the freshness condition \( (T_5 - T_4 < T) \), if matches then decrypts the message \( D_{L\oplus p}(CH_F) \) to obtain the credentials \( (LT,ID_{new}) \) and saves the credentials \( (SK,LT,ID_{new},TK,p,L,Z_{UID}) \) for further communication.
C. Proposed protocol for fast reconnect

When $D$ is disconnected from $AP$ due to some network issue and wants to rejoin again with frequently visited $AP$, it can easily connect with the network using fast reconnect credentials without requiring the full authentication process (i.e., Mutual authentication protocol between $D$ and $AS$ will not be invoked again).

- **$D \to AP$**: $D$ selects time stamp $T_1$ and random number $r_1$ to compute the $R_{auth} = E_{SK}(T_1 \parallel r_1 \parallel ID_{new} \parallel LT)$. After computing the $R_{auth}$, it forwards $<R_{auth}, T_1, LT>$ to $AP$.

- **$AP \to D$**: Upon receiving message $<R_{auth}, T_1, LT>$ from $AP$, $D$ gets timestamp $T_2$ and a random number $r_2$. Then it verifies the freshness condition ($T >= T_2 - T_1$), if the freshness condition holds then it searches the pair $(ID_{new}, TK)$ from its database based on the received $LT$. After that it decrypts the $D_{SK}(R_{auth})$ to obtain the credentials $(T_1, r_1, LT, ID_{new})$. These credentials are compared with the stored credentials $(T_1 = T_1^*, ID_{new} = ID^*_{new} \parallel LT = LT^*)$, if they match then $AP$ believes that $D$ is authentic and computes the $R_{auth} = E_{SK}(T_2 \parallel r_2 \parallel TS_K)$, and Temporary session key $TS_K = H(ID_{new} \parallel r_1)$.

D → AP: After receiving message $<R_{auth}, T_2>$ from the access point, D selects the timestamp $T_3$ to verify the freshness of the message by checking the freshness condition $T >= T_3 - T_2$, if it holds then it decrypts the $D_{SK}(R_{auth})$ to obtain the credentials $(T_2, r_2, TS_K)$. Then it computes the temporary session key $TS_K = H(ID_{new} \parallel r_1)$, and compares the obtained credentials with its own credentials $(T_3 = T_2^*, TS_K = TS_K^*)$. If they match then it believes that $AP$ is authentic and forwards the successful message to $AP$.

**Algorithm 1** Executed by IoT device $D$

**Input:** Value stored at $D$

**Output:** $SK = r_1 \oplus r_2 \oplus PW \oplus SID$

1. **Stpe-1.** Select random number $r_1$, get the timestamp $T_1$;
2. compute $CH = E_{K_{DP}}(UID \parallel T_1 \parallel r_1)$;
3. send $(CH, Z_{UID, T_1})$ to $AP$;
4. **Stpe-2.** */Wait for the message from $AP* "/$
5. receive $(RCH, T_2)$ from $AP$;
6. */message received, go ahead*/$
7. get current timestamp $T_3$;
8. if $((T_3 - T_2 < T) \&\&(SID == SID^{*}) \&\&(r_1 == r_1^{*}))$
9. $p \leftarrow p_1$;
10. Select $L$;
11. $RES_1 = E_{L}(PW \parallel T_3 \parallel r_2)$;
12. $RES_2 = E_{P}(L \parallel T_3 \parallel r_2)$;
13. send $(RES_1, RES_2, T_3)$ to $AP$;
else
ABORT;
**Stpe-3.** */Wait for the message from AP*/$
16. receive $(CH, T_3)$ from $AP$;
17. */message received, go ahead*/$
18. get current timestamp $T_5$;
19. if $((T_5 - T_4 < T) \&\&)$
20. $SK = H(r_1 \oplus r_2 \oplus PW \oplus SID)$;
21. $Z_{UID} = Z_{UID}^{*}$;
23. return $SK$.

V. INFORMAL ANALYSIS

In this part, we examine the security of proposed Mutual authentication protocol informally, demonstrating that it has additional security characteristics.

**Proposition 1.** The proposed protocol facilitates the mutual authentication.

**Proof.** When $D$ receives message $<RCH, T_2>$ from the $AS$, $D$ decrypts message $(RCH)$ and verifies the credentials $(SID^{*} == SID \&\& r_1^{*} == r_1)$. If credentials are matched, $D$ believes that $AS$ is authentic. Otherwise, it terminates the authentication process. On the other hand, when $AS$ receives message $<RES_1, RES_2, T_3>$, it decrypts message and compares the credentials $(r_2^{*} == r_2, PW^{*} == PW)$; if they match, $AS$ believes that $D$ is authentic. Thus, our proposed protocol facilitates mutual authentication.

**Proposition 2.** The proposed protocol for mutual authentication is resilient against Identity protection and...
Proof. The $D$ and $AS$ identities are always exchanged in masked form in the proposed protocol. Identity of $D$ is exchanged in masked encrypted form as $Z_{UID}$. Server’s identity is also exchanged in encrypted form in $RCH$. As a result, collecting the exchanged messages will not provide any information regarding the identity of the communicating parties. Further $D$’s encrypted masked identity is changed after each successful authentication session to $Z'_{UID}$. So, even if attacker captures the messages from multiple sessions he can not link the messages of one session to another session.

Proposition 3. The proposed protocol is resilient against ephemeral secret leakage (ESL) attack.

Proof. If the attacker gets access to the ephemeral secrets $r_1$ and $r_2$, he will be unable to compute the session key $SK = H(r_1 \oplus r_2 \oplus PW \oplus SID)$ since the attacker can only do so if he has access to long-term credentials ($PW, SID$). The session key is calculated using both long and short term credentials, and obtaining both is computationally impossible. As a result, our proposed protocol is resilient against ephemeral secret leakage (ESL) attack.

Proposition 4. The proposed protocol ensures Perfect Forwards Secrecy.

Proof. If an attacker obtains long term credentials ($UID, PW, SID$), he will be unable to obtain session key $SK = H(r_1 \oplus r_2 \oplus PW \oplus SID)$ because an attacker can only obtain $r_1$ and $r_2$ if he has $(k, p)$, which have already been replaced by new keys $(L_p, p_n)$ after successful authentication. As a result, even if an attacker has $(UID, PW, L_p, p_n, SID)$, getting $(SK)$ is difficult. As a result, our proposed protocol ensures perfect forward secrecy.

Proposition 5. The proposed protocol is resilient against the Replay attack.

Proof. The proposed protocol utilizes timestamps to verify the freshness of the exchanged messages (i.e., $(CH, T_1, Z_{UID})$, $(RCH, T_2)$, $(RES_1, RES_2, T_3)$, $(CH_F, T_4)$) by checking the freshness condition $(T_i - T_i-1 < T)$. If it matches then it believes that message is fresh otherwise abort the process. Hence, this implies that our proposed protocol is resilient against the Replay attack.

Proposition 6. The proposed protocol is resilient against Privileged insider attack.

Proof. If an attacker gains access to $D$’s database or eavesdrops stored credentials, he will be unable to retrieve the secrets since they are encrypted with $PW$, which is kept in a secure location. So, the proposed protocol is resilient against privileged insider attack.

Proposition 7. The proposed protocol is resilient against jamming / desynchronization attack.

Proof. $D$ or $AS$ may abort and re-initiate the authentication process at any step either because the freshness condition is not met or because credentials in the received message do not match with stored credentials. This may lead to a situation where after abort keys $k$ and $p$ with $D$ and $AS$ do not match. To handle this scenario $D$ and $AS$ make a copy of $k$ and $p$ before starting the authentication process. $D$ and $AS$ discard these copies only when they are sure that the authentication process has been successfully completed. As stated earlier, the channel between $D$ and $AP$ is not secure but reliable. $D$, after successfully receiving $CH_F$, sends an ACK to $AP$. It then waits for a time $T$. If no re-transmission is received, it discards the copies of $k$ and $p$. Then informs $AS$, and $AS$ also discards copies of $k$ and $p$. In all other cases, $D$ and $AS$ revert back to copies of $k$ and $p$ saved before starting the authentication process.

VI. FORMAL ANALYSIS

In this section, we use BAN logic [26] and Scyther tool [27] to perform a formal analysis of the proposed protocol.

A. Security Verification using BAN logic

$D$ and $AS$ are the communicating agents, $V$ is the statement and $K$ is the key. The notations used to define the BAN logic and assumptions for the proposed protocol are shown in Table II.

- For the protocol’s initial state, the following assumptions apply:
  
  $R_1 : D \equiv D \rightarrow AS$, $R_2 : D \equiv D \rightarrow AS$
  
  $R_3 : D \equiv \#(T_2)$, $R_4 : D \equiv \#(T_4)$
  
  $R_5 : D \equiv D \rightarrow AS$, $R_6 : D \equiv AS \rightarrow p_n$
  
  $R_7 : AS \equiv D \rightarrow AS$
  
  $R_8 : AS \equiv D \rightarrow AS$, $R_9 : AS \equiv \#(T_1)$
TABLE II: BAN notations and formulas

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D \equiv V, D \not\Rightarrow V$</td>
<td>$D$ believes $V$, $D$ receives $V$</td>
</tr>
<tr>
<td>$D \not\Rightarrow V, D \Rightarrow V$</td>
<td>$D$ once sent $V$, $D$ has full control over $V$</td>
</tr>
<tr>
<td>$(\forall V). (\forall K)_{K'}$</td>
<td>Based on $1)$ Validation and derivation of security goals: $D$</td>
</tr>
<tr>
<td>$\not\Rightarrow (\forall V)_{K} : D \not\Rightarrow D \not\Rightarrow AS$</td>
<td>Based on $2)$ Message Meaning (MM) Rule</td>
</tr>
<tr>
<td>$D \Rightarrow (\forall V)_{K} \Rightarrow AS$</td>
<td>Based on $3)$ Timestamp Verification (TV) Rule</td>
</tr>
<tr>
<td>$D \Rightarrow (\forall V)_{K} \Rightarrow AS$</td>
<td>Based on $4)$ The Jurisdiction Rule ($JR$)</td>
</tr>
<tr>
<td>$R_{10} : AS \not\equiv #: (T_{3})_{\not\Rightarrow}$</td>
<td>$R_{11} : AS \not\equiv D \not\Rightarrow L$</td>
</tr>
<tr>
<td>$R_{12} : D \not\Rightarrow AS \Rightarrow (D \not\Rightarrow AS)$</td>
<td>$R_{13} : AS \not\equiv D \not\Rightarrow Sk_{\Rightarrow AS}$</td>
</tr>
<tr>
<td>$R_{14} : AS \not\equiv # D \not\Rightarrow AS \Rightarrow (D \not\Rightarrow Sk_{\Rightarrow AS})$</td>
<td>The proposed protocol’s security goals are: $AS \not\equiv (D \not\Rightarrow Sk_{\Rightarrow AS}), D \not\equiv (D \not\Rightarrow Sk_{\Rightarrow AS})$</td>
</tr>
<tr>
<td>Idealized form of the proposed protocol:</td>
<td></td>
</tr>
</tbody>
</table>
| $M_{1} : D \Rightarrow AS : (UID || T_{3} || r_{1})_{\not\Rightarrow}$ | $M_{2} : AS \Rightarrow D : (SID || T_{2} || r_{1} || r_{2} || D \not\Rightarrow Sk_{\Rightarrow AS})_{\not\Rightarrow 
\not\Rightarrow p_{\not\Rightarrow}}$, $M_{3,1} : D \Rightarrow AS : (PW || T_{3} || r_{2})_{\not\Rightarrow}$, $M_{3,2} : D \Rightarrow AS : (SID || T_{3} || D \not\Rightarrow Sk_{\Rightarrow AS})_{\not\Rightarrow p_{\not\Rightarrow}}$, $M_{4} : AS \Rightarrow D : (LT || ID_{\not\Rightarrow new} || TK || T_{4} || AS \not\Rightarrow Sk_{\Rightarrow AS} D)_{\not\Rightarrow 
\not\Rightarrow p_{\not\Rightarrow}}$, |

1) Validation and derivation of security goals:

- Based on $R_{7}$ and $R_{8}$, we apply $MM$ rule on $M_{1}$
  - $S_{1} : AS \not\equiv D \not\Rightarrow M_{1}$
- Based on $R_{9}$ and $S_{1}$, we apply $TV$ rule on $M_{1}$
  - $S_{2} : AS \not\equiv D \not\Rightarrow (UID, r_{1})$
- Based on $R_{3}$ and $R_{2}$, we apply $MM$ rule on $M_{2}$
  - $S_{3} : D \not\Rightarrow AS \not\Rightarrow M_{2}$
- Based on $R_{3}$ and $S_{3}$, we apply $TV$ rule on $M_{2}$
  - $S_{4} : D \not\Rightarrow AS \not\equiv (SID, p_{n}, r_{2}, r_{1})$
- Based on $R_{6}$ and $S_{4}$, we apply $JR$ rule on $M_{2}$
  - $S_{5} : D \not\Rightarrow AS \not\Rightarrow Sk_{\Rightarrow AS}$
- Based on $R_{13}$, we apply $MM$ rule on $M_{3,2}$
  - $S_{6} : AS \not\equiv D \not\Rightarrow M_{3,2}$
- Based on $R_{10}$ and $S_{6}$, we apply $TV$ rule on $M_{3,2}$
  - $S_{7} : AS \not\equiv D \not\Rightarrow r_{2}, D \not\Rightarrow Sk_{\Rightarrow AS}$
- Based on $R_{11}$ and $S_{7}$, we apply $JR$ rule
  - $S_{8} : AS \not\equiv D \not\Rightarrow Sk_{\Rightarrow AS}$
- Based on $S_{8}$, we apply $MM$ rule on $M_{3,1}$
  - $S_{9} : AS \not\equiv D \not\Rightarrow M_{3,1}$
- Based on $R_{10}$ and $S_{9}$, we apply $TV$ rule on $M_{3,1}$
  - $S_{10} : AS \not\equiv D \not\equiv (PW, r_{2})$.

- Based on $R_{15}, S_{2}, S_{10}$ and $SK = (r_{1} \oplus r_{2} \oplus PW \oplus SID)$, we can infer $S_{11}$
  - $S_{11} : AS \not\equiv D \not\Rightarrow Sk_{\Rightarrow AS}$
- Based on $D_{14}$, we apply $JR$ rule on $S_{11}$
  - $S_{12} : AS \not\equiv D \not\Rightarrow Sk_{\Rightarrow AS} \rightarrow Goal_{1}$
- Based on $R_{5}$ and $S_{9}$, we apply $MM$ rule on $M_{4}$
  - $S_{13} : D \not\equiv AS \not\equiv M_{4}$
- Based on $R_{4}$, we apply $TV$ rule on $M_{4}$
  - $S_{14} : D \not\equiv AS \not\equiv ((D \not\Rightarrow Sk_{\Rightarrow AS}), LT, ID_{\not\Rightarrow new}, TK)$
- Based on $R_{12}$ and $S_{14}$, the $JR$ rule
  - $S_{15} : D \not\equiv (D \not\Rightarrow Sk_{\Rightarrow AS}) \rightarrow Goal_{2}$

B. Security Verification using Scyther tool

Scyther is a formal verification tool used to prove or disprove the security of protocols. The security protocols are modelled with the Security Protocol Description Language (spdl). The proposed protocol’s security properties are verified using the scyther tool. As shown in Fig.4, the validation result clearly shows that our proposed protocol addresses all of the security claims such as Alive (i.e., guarantees that the communicating parties carry out all events), Weakagree (i.e., guarantees that the protocol is not vulnerable to impersonation attacks), Nisynch (i.e., guarantees that the sender sends all messages and that the recipient receives them), and Secret specified by scyther tool. As a result, we may conclude that the Scyther tool found no vulnerabilities in the proposed protocol.

Fig. 4: Scyther tool result for Mutual authentication

VII. PERFORMANCE ANALYSIS

This section outlines the comparison of security features and experimental analysis to compute the cost of proposed protocol in terms of computational, communication, storage costs and energy consumption to examine the efficacy of the proposed protocol. In addition to this, we also demonstrate the overhead under unknown attacks and simulate the proposed protocol using the NS3 tool.
A. Experimental analysis using the MIRACL

This subsection demonstrates the experimental analysis performed using the MIRACL library [4] to compute the cost of the cryptographic operations employed in the proposed protocol. MIRACL is a standard C/C++ based programming library used by cryptography researchers to compute the cost of cryptographic operations. The cryptographic symbols used in proposed protocol $T_H$, $T_{AES}$, $T_{RSA}$, $T_{DH}$ and $T_{PM}$ are defined as the time required for one way hash function (256 bit), $(AES-128$ bit) encryption/ decryption, $(RSA-2048$ bits) encryption/decryption, Diffie-Helman $(DH)$ and $ECC$ multiplication $(ECC-256$ bits) respectively.

We computed the cost of cryptographic operations on two different platforms: (1) A desktop which is used as server and (2) on Raspberry Pi used as an IoT device.

Platform-1: A desktop as the Authentication Server (AS): A desktop was used as a server having the configuration: Intel(R) Core(TM) i7-3770 with 3.40 GHz clock, 8 GB RAM running Linux Ubuntu 18.04.6 LTS.

Platform-2: A Raspberry Pi as the IoT Device (D): A Raspberry Pi was used as the IoT device (UE) having the configuration: Model: 4B, CPU: ARM Cortex-A7, Cores: 4, and RAM: 8GB.

Each cryptographic primitive was run 100 times to see how well it works. Based on the longest and shortest run times, we estimated the average run-time in milliseconds (ms), which is shown in Table III.

| Table III: Results obtained through experimental analysis using the MIRACL |
|-----------------|----------------|----------------|----------------|----------------|
| Primitives      | $T_H$         | $T_{PM}$       | $T_{RSA}$      | $T_{AES}$      | $T_{DH}$      |
| Desktop (ms)    | 0.0032        | 0.495          | 4.69           | 0.0036         | 0.0041        |
| Raspberry Pi (ms) | 0.0015        | 1.54           | 8.14           | 0.0041         | 0.0042        |

**B. Computational cost**

In this section, we calculate and compare the cost of cryptographic operations in the proposed protocol and its counterparts. We use the computing time for cryptographic operations given in Table III, to evaluate the proposed protocol’s performance. The computational cost required for proposed protocol is $(6T_{AES}+2T_H) \approx 0.16$ (ms). The proposed protocol is least costly because it uses the combination of symmetric encryption and hash function. However, the combination of symmetric encryption and the hash function is less costly as compared to the combination of asymmetric encryption [4], [9]. Therefore, it is quite clear from the output of Table IV that the proposed protocol not only takes lesser cost compared to the protocols [3], [14], [15], [19], [22], [23] that use the asymmetric encryption but also the protocols [11], [13] that use the combination of symmetric and hash. However, the proposed protocol has slightly higher cost compared to symmetric encryption based protocols [10], [12] but provides more security features such as perfect forward secrecy, traceability, protection from ephemeral secret leakage, protection from privileged insider attack and fast reconnect which is lacking in [10], [12].

**C. Communication cost**

This section calculates and compares the number of bits sent in the channel for the proposed protocol and its counterparts. Based on earlier research (see [4]), we assess the cost of communication that is, identity, random number, each requiring 160 bits. AES encryption/decryption, hashed output, public-key encryption/decryption using RSA, need 128 bits, 256 bits, 2048 bits, respectively.

The communication bits required for proposed protocol is $(CH,T_1,Z_{UID}), (RCH,T_2), (RES_1,RES_2,T_3), (CH_F,T_4) \approx 896$ bits. The proposed protocol takes lesser communication cost because the exchanged messages are encrypted with $AES$. While $AES$ with a 128-bit key has the same level of security as $RSA$ with a 2048-bit key [4]. Therefore, the output of Fig.5a clearly indicates that proposed protocol has lesser communication cost not only the protocols [3], [14], [15], [19], [23] that use the RSA or $ECC$ but also the protocols [11], [22] that use the AES for exchanged messages. However, the proposed protocol has slightly higher cost compared to [10] and equivalent to [12], [13] but provides more security features such as perfect forward secrecy, traceability, protection from ephemeral secret leakage, protection from privileged insider attack, and fast reconnect not provided by [10], [12], [13].

**D. Storage cost**

This section calculates the amount of memory required on the mobile device to hold the permanent protocol data. For the storage cost evaluation, we consider the cost of the cryptographic operations as indicated in [4]. In the proposed protocol, $J$ is stored, which requires 128 bits. The storage cost evaluation shows that our proposed protocol require less storage cost compared to [3], [10]–[15], [19], [22], [23] as shown in Fig 5b.

| Table IV: Comparison of computation cost for protocols |
|-------------|----------------|----------------|----------------|
| P           | Device side    | Server side    | Total cryptographic operations | Total time (ms) |
| 10           | $M_{H} + M_{DH}$ | $M_{H} + M_{DH}$ | $M_{H} + M_{DH}$ | 0.7 |
| 11           | $AES_{H} + M_{DH} + T_H$ | $AES_{H} + M_{DH} + T_H$ | $AES_{H} + M_{DH} + T_H$ | 1.5 |
| 12           | $AES_{H} + T_H$ | $AES_{H} + T_H$ | $AES_{H} + T_H$ | 0.9 |
| 13           | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | 0.6 |
| 14           | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | 0.6 |
| 15           | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | 0.6 |
| 16           | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | 0.6 |
| 17           | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | 0.6 |
| 18           | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | $AES_{H} + T_PM$ | 0.6 |

![Fig. 5: Comparison of (a) Communication and (b) Storage cost of authentication protocols](image-url)
E. Energy consumption

In this section, we compute the energy required for the proposed protocol and compare it with other related protocols. We compute the energy consumption as in [28]. The energy usage of a “StrongARM” CPU running at 133 MHz doing various tasks is summarised as the energy required for transmitting a bit is 0.000066 mj, energy required for AES symmetric enc/dec is 0.00217 mj, energy required for Hashed output is 0.000108 mj, energy required for public key enc/dec RSA is 15.3 mj. The energy consumption needed for the proposed protocol is (896*0.00066+6*8*0.000108)=0.56 mj). The proposed protocol uses the AES and hash function which requires lesser energy consumption as compared to RSA or ECC [28]. Therefore, we can infer from the output of energy consumption shown in Table V that proposed protocol consumes lesser energy not only the protocols [3], [14], [15], [19], [22], [23] that use the RSA and ECC but also the protocols [11]–[13] that use the AES and hash function. However, the proposed protocol has slightly higher energy consumption as compared to [10] but provides more security features than [10].

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Exchanged message Energy consumption (mj)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[10]</td>
<td>(1216 × 0.00066 + 6 × 9.1 + 8 × 0.000108) 35.4</td>
</tr>
<tr>
<td>[14]</td>
<td>(0.552 × 0.00066 × 2 + 15.1 + 1 × 0.000207 + 2 × 8.65) 43.1</td>
</tr>
<tr>
<td>[15]</td>
<td>(0.526 × 0.00066 × 2 + 15.1 + 2 × 8.65) 43.6</td>
</tr>
<tr>
<td>[11]</td>
<td>(0.526 × 0.00066 × 2 + 15.1 + 8 × 0.000108) 1.9</td>
</tr>
<tr>
<td>[12]</td>
<td>(0.68 × 0.00066 × 2 + 15.1 + 8 × 0.000108) 1.9</td>
</tr>
<tr>
<td>[23]</td>
<td>(869 × 0.00066 × 6 × 0.000108) 0.56</td>
</tr>
<tr>
<td>Others</td>
<td>(869 × 0.00066 × 6 × 0.000108) 0.57</td>
</tr>
</tbody>
</table>

F. Security analysis

This subsection compares the proposed protocol and its counterparts in terms of security features and functionality (mutual authentication, perfect forward secrecy, identity protection, traceability, ephemeral secret leakage, privileged insider attack, de-synchronization attack, replay attack, etc.). Table VI summarizes the findings. For the symmetric key-based authentication protocols [10]–[12], the security analysis in the [13] shows that [10]–[12] are prone to various attacks shown in Table VI. Apart from that, the findings of [14] reveal that existing symmetric key-based authentication protocols [10]–[13] do not offer the robust security that makes them inappropriate for practical implementation over WLAN. On the other hand, asymmetric key-based authentication protocols [3], [14], [15], [19], [22], [23] offer better security as compared to symmetric key-based authentication protocols but are expensive in terms of computational, communication cost, and energy consumption. Table IV, Fig. 5a, Fig 5b and Table V show this. Therefore, asymmetric key-based authentication protocols are unsuitable for ultra-low cost IoT devices. The outcome of Table VI indicates that as compared to [3], [10]–[15], [19], [22], [23], the proposed protocol provides robust security not only against identified attacks but also offers additional security features such as protection from the privileged insider attack, ephemeral secret leakage, and traceability attack. Reason for this is that, after each successful authentication, the secrets such as identity of the device, keys, and random numbers used in the message exchange are updated. Therefore, accessing long-term secrets or the device where all the secrets are stored will not provide the attacker any insight into the session key or earlier secret information transmitted between the communicating entities. It is worth noting that not only symmetric encryption protocols [10]–[13] are vulnerable to various attacks, but the asymmetric encryption protocols also [3], [14], [15], [19], [22], [23] fail to offer some security features as illustrated in Table VI. Therefore, we can infer that the proposed protocol has a clear edge over its counterparts in terms of security.

G. Performance under unknown attack

This section looks into how well the proposed protocol and its competitors function in the face of unforeseen attacks. In the preceding subsections, we demonstrate that the proposed protocol is resilient against all identified known attacks. We anticipate there will be certain unidentified attacks that are difficult to predict when they occur. Therefore, we assessed the performance of the proposed protocol in the face of unknown attack by calculating the likelihood impact of an unknown attack, similar to [29], [30]. The performance under the unknown attack of the proposed protocol and its counterparts is computed using the Equation (1).

\[ C_A = \frac{C_S \times (1 - P) + C_f \times P}{(1 - P)} \]  

(1)

The terms \(C_A, C_S, C_f\) and \(P\) used in Equation (1) represents average cost, total cost of the successful authentication, cost when protocol halts (i.e., Equation (2)) in the step \(k\) and probability of attack in step \(k\) (i.e., independent of steps in which attack happens). The chance of an unknown attack occurring in step \(k\) is \(1/L\), where \(L\) is the total number of signaling messages in a single protocol execution. The outcome of Fig 6a, Fig. 6b, and Fig. 6c demonstrate that the proposed protocol outperforms its counterparts when unknown attack happens. This is due to the fact that the proposed protocol has less computational, communication and energy
consumption. However, the proposed protocol has slightly greater overhead than [10], [12] since they require less communication, computational, and energy. While the security study of [13] reveals that [10], [12] lack the prominent security characteristics, making them inappropriate for use in real-time applications. As a result of the findings, we may conclude that the proposed protocol performs better not in presence of known attacks, but also when unknown attack occurs.

\[ C_f = \sum_{k=1}^{L} C_k \times \frac{1}{L} \]  

(2)

H. Practical simulation using NS3

This section demonstrates the results of the experimental analysis carried out using the Network simulator tool NS3 [4]. We measure two network performance parameters (i.e., throughput and packet delivery ratio), to demonstrate the applicability of the authentication phase. Table VII depicts the parameters used in the simulation of authentication protocols.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uploading Semester</td>
<td>Ubuntu 18.04.6 LTS</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>100ns</td>
</tr>
<tr>
<td>Network coverage area</td>
<td>100 m x 100 m</td>
</tr>
<tr>
<td>Number of Access point &amp; Authentication server</td>
<td>10, 20, and 30</td>
</tr>
<tr>
<td>Routing protocol</td>
<td>OLSR</td>
</tr>
<tr>
<td>MAC protocol</td>
<td>IEEE 802.11</td>
</tr>
<tr>
<td>Distance between the devices and Access point</td>
<td>20 m to 50 m</td>
</tr>
</tbody>
</table>

The authentication phase takes place between the D, AP and AS which contains four messages: (CH, ZUID, T1), (RCH, T2), (RES1, RES2, T3), and (CHF, T4), of size 288, 160, 288, and 160 bits long. In order to show the efficiency, we also simulate the existing authentication protocols [11] [13] [14].

1) Impact on Throughput : The throughput is computed based on the number of bits transmitted per unit using the Equation (3).

\[ Throughput = \sum P_{Mi} \times N_i \frac{1}{t} \]  

(3)

Whereas \( P_{Mi} \) denotes the number of received packet of i type, \( N_i \) denotes the length of the packet of type i and \( t \) denotes the total time. The outcome of the throughput for scenarios 1, 2, and 3 are 915, 956, and 986 Kbps, respectively. The comparison result shown in Fig. 7a indicates that the proposed protocol has the highest throughput compared to [11], [13], [14]. This is due to the fact that the message size and cryptographic operations of the proposed protocol are less than the [11], [13], [14]. Apart from that, it can be observed from the throughput results that when the number of nodes exceeds, throughput exceeds as well. The main reason behind this is that the number of authentications between the server and the devices has increased.

2) Packet delivery ratio (PDR) : The Packet delivery ratio is the parameter used to track network congestion using the Equation (4).

\[ PDR = \frac{R_p}{S_p} \]  

(4)

Where \( R_p \) represents the number of the received packet, and \( S_p \) represents the number of sent packets by the sender. The Packet delivery ratios for scenarios 1, 2, and 3 are 98.3%, 97%, and 95%, respectively. The comparison result shown in Fig. 7b shows that the proposed protocol has the highest packet delivery ratio as compared to [11], [13], [14]. The outcome of the packet delivery ratio shows that as the number of nodes increases, the PDR decreases due to congestion.

VIII. PROTOTYPE IMPLEMENTATION

To determine the feasibility of the proposed protocol, a prototype test-bed was created for real-time implementation. We set up a TCP-based communication channel between the communicating entities, as shown in Fig. 8. A Wireless Access Point (AP) was used to route the connection. The channel was built on a Java platform and used a socket-based inter-process communication (IPC) method. To complete the message flow, two primary classes were formalized, and the IPC scenario was established using the sequence shown in Fig. 1. It was assumed that pre-shared symmetric key would be distributed to all parties involved in communication. In Fig. 8, the derived parameters and procedures are listed. We tested our protocol for two different setup as explained below. The goal was to compare and contrast the performance of a resource-limited IoT device with those of a resource-rich device.

Setup-1: A laptop as IoT device and a desktop as the server: We use a laptop as the IoT device baring the configurations: Intel(R) Core(TM) i7-3770 with 3.40 GHz clock, 8 GB RAM running on Windows 10 and the desktop as the server which specifications are indicated in Fig. 8.

Setup-2: A Raspberry Pi as the IoT Device and desktop as the server: A Raspberry Pi (Model : 4B, CPU: ARM® Cortex®-A7, Cores : 4, and RAM : 8GB) was deployed as the IoT device (D). The same Wireless AP was deployed for both scenarios. This scenario reflects a real resource-constrained IoT environment.

To demonstrate the efficacy in terms of average completion time, we implement the proposed protocol and the more contemporary protocols [11], [13], [14] in a test-bed environment. These protocols are implemented for different key sizes (i.e.,Size combination (SC1)=(AES-128, Hash-160)), (SC2=(AES-128, Hash-256)), (SC3=(AES-256, Hash-160)), (SC4=(AES-256, Hash-256))). While, we used RSA encryption with a 2048-bit key size for the [14]. The comparison outcome of Fig. 9a and Fig. 9b shows that proposed protocol has the less average completion time compared to [11], [13], [14]. The rationale behind this is that proposed protocol takes less communication and computational cost compared to [11], [13], [14]. Apart from that, [14] is based on asymmetric encryption that is why it requires very high completion time while [11], [13] are based on symmetric that requires the high completion time compared to proposed protocol and are vulnerable to several types of attacks shown in Table VI. Therefore, we can conclude from the implementation results that proposed protocol performs better compared to asymmetric encryption based protocol [14] as well as symmetric encryption based protocols [11], [13].
In this paper, we present a mutual authentication protocol for the WLAN communication. To achieve a balance between security criteria and at the same time being lightweight, the proposed protocol employs a combination of symmetric encryption and hash functions. We provide an informal and formal (using BAN logic and Scyther tool) analysis of the proposed protocol, which demonstrates that it is secure against the attacks. Moreover, we evaluate the proposed protocol’s performance in terms of computational, communication, storage, and energy consumption, demonstrating that the proposed protocol is lesser expensive than the existing protocols. In addition to this, we compute the overhead under unknown attacks indicating that proposed protocol takes less overhead compared to its counterparts. Furthermore, we show the practical simulation of the proposed protocol using the NS3 tool to confirm it applicability in practical scenarios. A prototype implementation has been done to show that it can be easily implemented in real time applications. As a result, we may conclude that the proposed protocol is safe, efficient, suitable for IoT applications and provide the balance between the security and cost.

REFERENCES


Awaneesh Kumar Yadav (Member, IEEE) is doing Ph.D in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, India. His research area includes Network security, 5G security, IoT & Cloud Security. He did his M.Tech in the Department of Computer Science and Engineering, National Institute of Technology Rourkela, India, in 2019.

Manoj Misra (Member, IEEE) is a Professor in Department of Computer Science and Engineering at IIIT Roorkee. Dr. Misra got his PhD from University of New Castle upon Tyne and has past experience of working as an Engineer at CMC Limited Noida, Assistant Engineer at Hindu Aeronautic Limited at Kanpur India, Assistant Professor at HBTI Kanpur. His research interests include Distributed Computing, Performance Evaluation, Computer Networks, Network Security and Cyber frauds.

Pradumn Kumar Pandey (Member, IEEE) received the Bachelor of Technology and Ph.D. degrees in computer science and engineering from IIT Jodhpur, Jeeapansi, India, He was an Institute Post-Doctoral Fellow with the Department of Computer Science and Engineering, IIT Kharagpur, Kharagpur, India, from May to September 2018. He worked as a DST INSPIRE Faculty Member with the Department of Computer Science and Engineering, IIIT Roorkee, Roorkee, India, from October 2018 to October 2019, where he has been working as an Assistant Professor since November 2019. His research areas include modeling of complex networks, information diffusion on real networks, social security on online social networks, and network representation learning.

Madhusanka Liyanage (Senior Member, IEEE) received his the Doctor of Technology degree in communication engineering from the University of Oulu, Oulu, Finland, in 2016. From 2011 to 2012, he worked as a Research Scientist at the 13S Laboratory and Inria, Sophia Antipolis, France. He is currently an assistant professor/Ad Astra Fellow and the Director of Graduate Research at the School of Computer Science, University College Dublin, Ireland. He is also acting as an Adjunct Professor at the Center for Wireless Communications, University of Oulu, Finland and Department of Electrical and Information Engineering - University of Ruhuna, Sri Lanka. He was also a recipient of the prestigious Marie Skłodowska-Curie Actions Individual Fellowship during 2018-2020. In 2020, he received the “2020 IEEE ComSoc Outstanding Young Researcher” award by IEEE ComSoC EMEA. Dr. Liyanage is an expert consultant at European Union Agency for Cybersecurity (ENISA). In 2021, Liyanage was elevated as Funded Investigator of Science Foundation Ireland CONNECT Research Centre, Ireland. He was ranked among the World’s Top 2% Scientists (2020) in the List prepared by Elsevier BV, Stanford University, USA. Also, he was awarded a Irish Research Council (IRC) Research Ally Prize as part of the IRC Researcher of the Year 2021 awards for the positive impact he have made as a supervisor. Moreover, he is an expert reviewer at different funding agencies in France, Qatar, UAE, Sri Lanka, and Kazakhstan. Dr. Liyanage’s research interests are 5G/6G, SDN, IoT, Blockchain, MEC, mobile, and virtual network security. More info: www.madhusanka.com