A Novel Secure Authentication Protocol for IoT and Cloud Servers

Ummer Iqbal, Aditya Tandon, Sonali Gupta, Arvind R. Yadav, Rahul Neware, and Fraol Waldamichael Gelana

1National Institute of Technology Srinagar, Jammu and Kashmir, India
2Krishna Engineering College, Ghaziabad, Uttar Pradesh, India
3Department of Computer Engineering, J C Bose University of Science and Technology, Faridabad, Haryana, India
4Parul Institute of Engineering & Technology, Parul University, Vadodara, India
5Department of Computing, Mathematics and Physics, Høgskulen På Vestlandet, Bergen, Norway
6Artificial Intelligence Centre, Addis Abada, Ethiopia

Correspondence should be addressed to Ummer Iqbal; drkhaniqbalummer@gmail.com and Fraol Waldamichael Gelana; fraol.gelana@aic.et

Received 29 December 2021; Accepted 1 February 2022; Published 3 March 2022

Copyright © 2022 Ummer Iqbal et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The integration of IoT with the cloud infrastructure is essential for designing smart applications. However, such integration may lead to security issues. Authentication and key establishment is an essential security requirement for secure communication between IoT devices and cloud servers. For evaluating authentication key agreement schemes, the extended Canetti–Krawczyk (eCK) adversary model is regarded to be a more strict and relevant adversary model. Many schemes for authenticated key exchange between IoT devices and cloud servers have been proposed in the literature but have been assessed under Dolev and Yoa (DY) adversary model. Recently, Rostampour et al. introduced an ECC-based approach for enabling authentication between IoT devices and cloud servers that is secure and robust to various attacks under the Dolev and Yoa adversary model. In this paper, a detailed review and the automated security verification of the Rostampour et al. scheme are carried out under the eCK adversary model using Scyther-Compromise. The validation indicates that the scheme is not secure and is susceptible to various attacks under the eCK adversary model. To overcome the limitation of the Rostampour et al. scheme, a design of an ECC-based scheme for authentication between IoT devices and cloud servers under the eCK adversary model is proposed. The Scyther verification indicates that the scheme is safe under the eCK adversary model. The soundness of the correctness of the proposed scheme has been analyzed using BAN logic. Comparative analysis indicates that the scheme is resilient under the eCK adversary model with an energy overhead of 278.16 mJ for a resource constraint IoT device and a communication overhead of 1,408 bits.

1. Introduction

The internet of things (IoT) is a new network that provides numerous embedded devices to the Internet to share data. It is based on information and communication technologies. Embedded devices are becoming increasingly complicated as a result of the fast growth of technology, and they are employed in a broad variety of applications. The ability to relate such gadgets to huge resource pools, such as the cloud, is a significant advancement in modern technology. The integration of embedded devices and cloud servers enables the use of IoT in a wide range of commercial and government applications. However addressing the security issues, such as authentication and data privacy is important for the efficient integration of these two systems [1–3]. IoT has grown significantly over the years because of its relevance in smart applications. The internet of things (IoT) market was valued at USD 250.72 billion in 2019 and is expected to grow to USD 1,463.19 billion by 2027 [4]. This exponential advancement of IoT is governed by its role in developing applications that include smart parking, smart building, smart health, smart environment, smart agriculture, and smart homes [5, 6]. Smart parking can be employed for effective monitoring of parking areas in the city, and
smart building applications can be used for monitoring the structural health of a building in terms of vibrations and material conditions. IoT can also be used for real-time monitoring the health and the hydraulic parameters of the water bodies and effective monitoring of road conditions in terms of climate conditions [5]. In terms of security and emergency applications, IoT nodes can be deployed for unauthorized perimeter access, detection of harmful chemical and radiation leaks, and detection of gas leaks in industrial surroundings [6]. For smart agriculture, the nodes can be employed for monitoring soil quality, climate control in the greenhouse, and smart irrigation. The various applications in smart homes include surveillance, remote monitoring of appliances, energy, and water conservation. In the medical field, IoT can be employed for developing applications that include wireless body area networks, geriatric assistance, and automatic monitoring of medical freezers [5].

A typical IoT-based smart application comprises a three-tier architecture that includes perception tier, network and server tier, and application tier [5, 7] as shown in Figure 1. The physical parameters are perceived in the perception layer using IoT devices, and the data perceived are further relayed to the server tier. The server tier serves as a communication backbone, employing different communication technologies [7]. As IoT applications generate a large volume of data, the server tier includes features for storing and processing it. A variety of servers, including mobile, web, and real-time communication servers, provide middleware support at the server tier. As the perception layer generates a very large volume of data, cloud servers can also be used at this layer for data storage and management. Various end-users connect to the server tier for online and offline data analysis in the user tier [8, 9]. The security of IoT-based smart applications is of paramount importance [10]. The communication between the server tier and client tier can be made secure using traditional security protocols. However, as IoT devices are resource-constrained, the communication between IoT devices in the perception tier and servers in the server tier becomes an emerging and active area of research [5].

Authentication and session key establishment are bedrock for all other security services between IoT devices in the perception layer and the various servers in the server tier [11]. Many schemes have been suggested for securing communication between IoT devices and cloud servers. Liao and Hsiao [12] proposed a mutual authentication scheme with ID verifier protocol. However as indicated by [13], the scheme suffers from server impersonation attacks.

Kalra and Sood [14] presented a mutual authentication scheme based on ECC. The scheme is resilient to various attacks. However, the scheme has design issues in terms of mutual authentication, insider, and traceability attacks. Chang et al. [15] pointed out the limitations of Kalra and Sood and highlighted its weakness in terms of mutual authentication and mistress of the session key. Chang et al. suggested an improved scheme that overcame the limitations of Kalra and Sood. Kumari et al. [16] pointed out the deficiencies of Kalra and Sood in terms of insider attacks, device anonymity, session key agreement, and mutual authentication. Kumari et al. also suggested an ECC-based scheme to overcome these limitations. Wang et al. [17] highlighted the deficiencies of the Chang et al. scheme in terms of impersonation attacks and suggested an improved scheme. Recently, Rostampour et al. [11] proposed an ECC-based scheme for authentication of IoT edge devices with the cloud server. Rostampour et al. made a detailed review of the schemes of Kalra and Sood, Chang et al., Kumari et al., and Wang et al. and highlighted their limitations. Rostampour et al. pointed out that all the existing schemes suffer from traceability attacks.

When connecting an embedded system to the cloud, security is the main consideration. Mutual authentication must also be established between the cloud server and the embedded devices. To address these security concerns, many authentication systems for IoT and cloud servers have been developed. However, there are several faults in the existing approaches that must be addressed. When memory and power are limited and greater security with a low key length is required, elliptic curve cryptography (ECC) is the best public-key cryptography solution [18, 19]. The existing schemes presented in the literature have been analyzed under Dolev and Yoa (DY) [20, 21]. However, the eCK [22] adversary model is a more stringent model for authentication key agreement schemes [23]. In this paper, a detailed review of the Rostampour et al. scheme has been made. The Rostampour et al. scheme has been formally validated using Scyther-Compromise [24] under the eCK adversary model. Scyther-Compromise validation depicts that the Rostampour et al. scheme is not secure under the eCK adversary model. Furthermore, an improved scheme for authentication and key agreement between IoT devices and the server under the eCK adversary model has also been proposed.

1.1. Contributions. The major highlights of the paper are as follows:

(a) A review of the Rostampour et al. scheme has been made carried out. The scheme has been modeled on Scyther-Compromise and analyzed under the eCK adversary model.

(b) An ECC-based authentication scheme for IoT and cloud servers has been designed. The designed scheme provides better functionality and security specifications.

(c) The proposed scheme has been modeled on Scyther-Compromise. The results indicate that the scheme is safe under the eCK adversary model.

(d) The soundness and correctness of the proposed protocol have also been evaluated using BAN [25] logic.

1.2. Paper Organization. The remainder of the paper is laid out as follows. The preliminaries are presented in Section 2. Section 3 reviews the Rostampour et al. scheme and discusses its formal security verification under the eCK adversary model. In Section 4, the design of the proposed
ECC-based scheme for authentication between IoT devices and cloud servers is presented. In Section 5, an informal security analysis has been carried out. Section 6 provides the simulation details of formal security analysis using the Scyther under the eCK adversary model. In Section 7, security analysis using BAN logic has been carried out. Finally, Section 8 presents the comparison of the proposed scheme with other relevant schemes in terms of functional and security specifications.

2. Preliminaries

2.1. Notations. The notations used in the paper are tabulated in Table 1.

2.2. Adversary Model. An adversary model formulates the potential capabilities an attacker can have. Various adversary models exist that include Dolev and Yoa (DY) [20, 21], Canetti–Krawczyk (CK), and extended Canetti–Krawczyk models (eCK) models [22]. In all the models, the communication channel is completely insecure; however, they differ in their adversary query capabilities. In DY threat model, the communication parties are considered to be honest and can run multiple sessions with each other. The communication channel is completely insecure and totally under the adversary’s control, with the ability to record, delete, replay, redirect, rearrange, and completely control message schedule. The adversary can act as an active adversary in the middle and lead various man-in-the-middle attacks. The CK and eCK models are the most widely accepted models for authentication and key agreement protocols. In this adversary model, an attacker has abilities to compromise PRNG and get access to the secret randomness of the session. It is also assumed that an adversary can compromise the session and get access to it. An attacker can also get access to the long-term keys [26]. The major difference between the CK model and the eCK model is that in the eCK model, the adversary is capable of accessing ephemeral secrets, thus leading to an ephemeral-secret-key-leakage attack. An ephemeral key leakage indicates the weaknesses of a random number generator where the attacker can determine the randomness generated correctly with a high probability [26].

2.3. Elliptical Curve Cryptography. Koblitz [27] and Miller [28] introduced elliptic curve cryptography (ECC). ECC is a powerful cryptographic technology. It establishes security between key pairs for public-key encryption using elliptic curve mathematics. ECC has grown in popularity in recent years due to its smaller key size and ability to maintain security. This trend is expected to continue as the need for devices to be secure develops in response to the rising size of keys, putting pressure on limited mobile resources. This is why it is vital to understand elliptic curve cryptography in context of low power devices [29]. When compared to RSA, ECC is highly efficient with low overheads. With a key size of 160 bits, elliptical curve cryptography provides the same level of security as RSA with a key size of 1,024 bits, making it

![Figure 1: Three-tier architecture of IoT applications.](image-url)
ideal for devices with limited resources [29]. Over a finite prime field $F_p$, an elliptical curve $E_p(a, b)$ is defined as follows [29]:

$$E_p(a, b): y^2 = x^3 + ax + b,$$

where $\Delta = (4a^3 + 27b^2) = 0$.

Some of the essential definitions related to elliptical curve cryptography are listed below [29]:

(a) Point addition: point addition $(+)$ between any two points $[A(x_A, y_A), B(x_B, y_B)] \in E_p(a, b)$ is defined as follows:

$$C(x_R, y_R) = A(x_A, y_A) + B(x_B, y_B),$$

where $C(x_R, y_R)$ is a reflection of the point at which the line joining $A(x_A, y_A)$, and $B(x_B, y_B)$ intersects the curve $E_p(a, b)$. The algebraic computation is given as follows:

$$\begin{pmatrix} x_R = (\lambda^2 - x_A - x_B) \mod p \\ y_R = (\lambda(x_A - x_B) - y_A) \mod p \end{pmatrix},$$

where

$$\lambda = \begin{cases} \frac{(y_B - y_A)}{(x_B - x_A)} \mod p, & \text{if } A! = B, \\ \frac{3x_A^2 + a}{2y_A} \mod p, & \text{if } A = B. \end{cases}$$

(b) Scalar multiplication: for any scalar number $X$ and $A(x, y) \in E(a, b)$, the scalar multiplication is defined as follows:

$$X . A(x, y) = A(x, y) + A(x, y) + A(x, y) + \ldots + A(x, y),$$

where $X$ times.

(c) Elliptical discrete logarithmic property (ECDLP): given points $R(x, y), G(x, y) \in E_p(a, b)$, ECDLP is a computational problem to find a scalar $n$ such that $R(x, y) = n.G(x, y)$. ECDLP has been a prominent field of research in cryptography over many decades, and it is the essential foundation for elliptic curve cryptography (ECC) and pairing-based cryptography. ECDLP has exponential time complexity.

2.4. Scyther Simulation. Scyther is a tool for validating and verifying security protocols that was created by Cremers [24]. It is a software that provides security risk assessments and attack simulation capabilities. Scyther uses an infinite number of sessions to verify security protocols. Scyther also allows multiprotocol assaults to be verified. Figure 2 depicts the Scyther Tool’s design in its most basic form. To verify and validate a security protocol in Scyther, it is modeled using Scyther protocol description language (SPDL). The tool accepts the SPDL model to be validated as well as simulation settings. As an output, the Scyther tool generates a summary report showing if the necessary security assertions are valid. If an attack is discovered, it generates a trace pattern in the form of a visual graph or an XML representation. An SPDL model of security protocol comprises roles that define the communication pattern of sender and receiver principals. The term claim is used to specify the various security requirements. Claim secret declares the parameters that must remain confidential from the adversary. The claim alive ensures that the claimant is executing events with the intended party. Nisynch ensures that all the intended messages in the session are sent and received in a synchronized manner, whereas Niagree indicates that the communicating parties agree on the messages exchanged. Finally, the weakagree ensures resilience against impersonation attacks. SKR claim indicates whether the session key designated using SKR can be revealed using the session key adversary rule. To verify this claim, the session key rule in the adversary model setting must be enabled. More details about Scyther are given in [24].

The adversary model by default in the Scyther standard version is Dolev and Yoo. The Scyther-Compromise provides extended support for various adversary models as compared to the standard Scyther version. In this paper, Scyther-Compromise version 0.9.2 has been used.

3. Review and Security Validation of Rostampour et Al’s Scheme

3.1. Review of Rostampour et Al’s Scheme. The Rostampour et al. scheme comprises two phases given as below:

(1) Registration phase
(2) Login and authentication phase

3.1.1. Registration Phase. The registration phase is carried out between the device and the server over a secure channel. The steps are given below:

(i) $D_i$ generates a random number $X_i$ and computes $PID_i$ as (4):

$$PID_i = X_i \oplus ID_i.$$  

$D_i$ sends $PID_i$ to the server $S$ as follows:

$$D_i \rightarrow S: PID_i.$$  

(ii) The server generates a random number $R_i$ and calculates (7) and (8):

$$CK_i = H(R_i \oplus X_i \oplus ET \oplus PID_i),$$  

$$CK_i = CK_i . G.$$  

Server stores $PID_i, ET$, and $CK_i$ in a secure database.
3.1.2. Login and Authentication Phase

(i) The network server generates a random number $N_1$ and computes (10)–(12):

$$P_1 = N_1G, \quad P_2 = N_1CK', \quad PPID_i = PID_iP_1.$$  \hspace{1cm} (10) \hspace{1cm} (11) \hspace{1cm} (12)

And sends it to the server as follows:

$$D_i \rightarrow S: P_1, P_2, PPID_i.$$  \hspace{1cm} (13)

(ii) The server finds the entry of $D_i$ in its database and verifies $P_2 \neq CK_iP_1$. If the evaluation is false, then the process stops, and if true, then the device is authenticated.

(iii) The server generates a random number $N_2$ and computes (14) and (15):

$$P_3 = N_2G, \quad P_4 = (CK_i' + PPID_i) + (N_2P_1).$$  \hspace{1cm} (14) \hspace{1cm} (15)

Also, it sends it to the device as follows:

$$S \rightarrow D_i: P_3, P_4.$$  \hspace{1cm} (16)

(iv) Device verifies $P_2^2 = (PPID_i, CK_i') + (N_1, P_2)$. If the evaluation is false, then the process stops, and if true, then the device is authenticated. The device computes $V_i$ as in the following equation and sends it to the server:

$$V_i = PPID_iP_3,$$  \hspace{1cm} (17)

$$D_i \rightarrow S: V_i.$$  \hspace{1cm} (18)

(v) The server receives $V_i$ and verifies $V_i \neq N_3, PPID_i$. If the evaluation is false, then the process stops, and if true, then the device is authenticated.

(vi) The server computes its session key: $SK = N_2, P_1$, and the device computes its session key: $SK = N_1, P_3$, such that $SK = N_1, N_2, G$.

The schematics of the Rostampour et al. scheme are given in Figure 3.
Figure 3: Schematics of the Rostampour et al. scheme.
<table>
<thead>
<tr>
<th>Protocol description</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. hashfunction MUL;</td>
<td></td>
</tr>
<tr>
<td>2. hashfunction ADD;</td>
<td></td>
</tr>
<tr>
<td>3. Const G: Nonce;</td>
<td></td>
</tr>
<tr>
<td>4. secret PIDI, CKI, CKI-PRIME;</td>
<td></td>
</tr>
<tr>
<td>5. secret N1, N2;</td>
<td></td>
</tr>
<tr>
<td>6. macro P1=MUL (N1, G);</td>
<td></td>
</tr>
<tr>
<td>7. macro P2=MUL (N1, CKI, CKI-PRIME);</td>
<td></td>
</tr>
<tr>
<td>8. macro P3=MUL (PIDI, P1);</td>
<td></td>
</tr>
<tr>
<td>9. macro P4=MUL (N2, G);</td>
<td></td>
</tr>
<tr>
<td>10. macro P4=ADD (MUL (PIDI, CKI), MUL (N2, P1));</td>
<td></td>
</tr>
<tr>
<td>11. macro VI=MUL (PIDI, P3);</td>
<td></td>
</tr>
</tbody>
</table>

protocol improved (I, R)

role I

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>var Message1: Nonce;</td>
<td></td>
</tr>
<tr>
<td>secret PIDI, CKI, CKI-PRIME;</td>
<td></td>
</tr>
<tr>
<td>send_1 (I, R, PPIDI, P1, P2);</td>
<td></td>
</tr>
<tr>
<td>recv_2 (R, I, P3, P4);</td>
<td></td>
</tr>
<tr>
<td>send_3 (I, R, VI);</td>
<td></td>
</tr>
<tr>
<td>recv_4 (R, I, [Message] MUL (N1, N2, G));</td>
<td></td>
</tr>
<tr>
<td>Claim_i1 (I, Secret, N1);</td>
<td></td>
</tr>
<tr>
<td>Claim_i2 (I, SKR, MUL (N1, N2, G));</td>
<td></td>
</tr>
<tr>
<td>Claim_i3 (I, Nagree);</td>
<td></td>
</tr>
<tr>
<td>claim_i4 (I, Nisynch);</td>
<td></td>
</tr>
</tbody>
</table>

role R

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fresh Message1: Nonce;</td>
<td></td>
</tr>
<tr>
<td>secret PIDI, CKI, CKI-PRIME;</td>
<td></td>
</tr>
<tr>
<td>recv_1 (I, R, PPID, P1, P2);</td>
<td></td>
</tr>
<tr>
<td>send_2 (R, I, P3, P4);</td>
<td></td>
</tr>
<tr>
<td>recv_3 (I, R, VI);</td>
<td></td>
</tr>
<tr>
<td>send_4 (R, I, [Message1] MUL (N1, N2, G));</td>
<td></td>
</tr>
<tr>
<td>Claim_r1 (R, Secret, N2);</td>
<td></td>
</tr>
<tr>
<td>Claim_r2 (R, SKR, MUL (N1, N2, G));</td>
<td></td>
</tr>
<tr>
<td>Claim_r3 (R, Nagree);</td>
<td></td>
</tr>
<tr>
<td>claim_r4 (R, Nisynch);</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: SPDL model of the Rostampour et al. Scheme.

Figure 5: Scyther settings for Dolev and Yoa adversary models.
4.1. Registration Phase. The registration phase includes the registration of an IoT device with the server. As with other schemes, the registration is performed on a secure channel [11]. The need for a secure channel for registration is highlighted as there is no preshared secret in the IoT device or any trusted third party is employed. The steps taken between the device and the server during the registration are listed as below:

(i) Device $D_i$ chooses its private key $K_i$ and its identity $ID_i$.
(ii) The device calculates $PID_i$ as in (19) and sends $PID_i$ to the server:

$$PID_i = K_i.ID_i.G,$$

$$D_i \rightarrow S: PID_i.$$  \hspace{1cm} (19)

(iii) The server calculates the hash of $PID_i$, and splits its private key $X_S$ into two unequal halves $X^1_S$, $X^2_S$ such that $X^1_S \neq X^2_S$. The server further computes (20) and (21):

$$CK^1_i = PID_i.X^1_S,$$ \hspace{1cm} (20)

$$CK^2_i = PID_i.X^2_S.$$ \hspace{1cm} (21)

The server stores $PID_i, CK^1_i, CK^2_i$ and sends $CK^1_i, CK^2_i$ to the device as follows:

$$S \rightarrow D_i: CK^1_i, CK^2_i.$$ \hspace{1cm} (22)

(v) The device receives and stores the parameters $CK^1_i, CK^2_i$ along with $PID_i$ in its memory.

4.2. Login and Authentication Phase. During this phase, the device initiates to log in to the server. Subsequently, the server authenticates the device, and if the device is authenticated, a session key is subsequently established between the device and the server. The steps undertaken between the device and the server during the login and authentication phase is shown as below:

(i) Device $D_i$ chooses its ephemeral secret $S_i$ and calculates $P_1$ and $P_2$ as (23) and (24):

$$P_1 = (CK^1_i + CK^2_i).S_i.H(PID_i),$$ \hspace{1cm} (23)

$$P_2 = (CK^1_i + CK^2_i).S_i.$$ \hspace{1cm} (24)

The device sends $P_1$ and $P_2$ to the server as follows:

$$D_i \rightarrow S: P_1, P_2.$$ \hspace{1cm} (25)
(ii) The server receives $P_1, P_2$ from $D_i$ and authenticates $D_i$ as follows:

(a) The server finds the corresponding entry of $D_i$ and calculates $H(PID_i)$ from $PID_i$ of the device stored in the database.

(b) The server calculates the multiplicative inverse of $H(PID_i)$ as $[H(PID_i)]^{-1}$ and computes $P'_1$ as $P'_1 = P_1[H(PID_i)]^{-1}$.

(c) The server compares $P'_1 \neq P_2$. If true, then Step 3 is performed; else, login request from $D_i$ is rejected.

(iii) The server chooses its ephemeral secret $S_s$ and calculates $P_3$ and $P_4$ as (26) and (27):

$$P_3 = P_1.S_s,$$  \hspace{1cm} (26)

$$P_4 = (CK_i^1 + CK_i^2).S_s.$$  \hspace{1cm} (27)

(iv) The server computes its session key ($SK$) with $D_i$ as follows:

$$SK = P_2.S_s,$$  \hspace{1cm} (28)

$$SK = (CK_i^1 + CK_i^2).S_s.$$

(v) Server sends $P_3, P_4$ to the device as follows:

$$S \rightarrow D_i: P_3, P_4.$$  \hspace{1cm} (30)

(vi) The device receives $P_3, P_4$ from the server and authenticates it as follows:

(a) The device calculates the multiplicative inverse of $H(PID_i)$ as $[H(PID_i)]^{-1}$ and computes $P'_3$ as $P'_3 = P_3[H(PID_i)]^{-1}$.

(b) The device compares $P'_3 \neq P_4$. If true, then $S$ is authenticated; else, login response from $S$ is rejected.
(vii) The device computes its session key with $S$ as follows:
\[ SK = P_4 . S_1 , \]  
\[ SK = ( CK_1 + CK_2 ) S_2 . S_3 , \]  
\[ SK = ( PID_i . X_1 + PID_i . X_2 ) S_4 . S_5 , \]  

(viii) The device encrypts $ESK [ H (SK) ]$ and sends it to the server as follows:
\[ D_4 \rightarrow S : ESK [ H (SK) ] . \] (33)

(ix) The server decrypts $ESK [ H (SK) ]$ as $V = D_5 [ ESK [ SK ] ]$ and verifies that whether $V \neq H [ SK ]$. If true, then the login request is approved, and the device is authenticated.

The schematics of the proposed scheme are given in Figure 9.

5. Security Analysis

5.1. Replay Attack. In a replay attack, an adversary stores the messages exchanged between the communicating parties and retransmits them in the future to gain illegitimate access. In the proposed protocol, three messages are exchanged between the device and the server:
\[ D_i \rightarrow S: P_1 , P_2 , \]
\[ S \rightarrow D_i : P_3 , P_4 , \]  
\[ D_i \rightarrow S : ESK [ H (SK) ] . \] (34)

Let us assume during the initiation of the login and authentication phase, an adversary can eavesdrop on the messages and stores $(P_1, P_2, P_3, P_4)$ and $ESK [ H (SK) ]$ for a replay attack. Let an adversary (A) replay stored $(P_1, P_2)$ of the previous session to gain illegitimate access as follows:
\[ A \rightarrow S : P_1 , P_2 . \] (35)

When the server receives $(P_1, P_2)$, it validates the request and generates $(P_3, P_4)$ with a new ephemeral random secret. The adversary receives $(P_3, P_4)$ a new ephemeral secret. To validate $(P_3, P_4)$ and subsequently generate a session key SK, the adversary needs to know PID, and the ephemeral random secret previously used for computing $(P_1, P_2)$. The adversary does not have access to either due to the exponential time complexity of ECDLP [30–32]. As the adversary cannot generate a session key SK of the current session, $ESK [ H (SK) ]$ cannot be computed by the adversary. Moreover, if the adversary replays the old $ESK [ H (SK) ]$, the server will not authenticate it as the current SK is based on the fresh ephemeral random secret of the server. Thus, the design of the scheme thwarts replay attacks.

5.2. Impersonation Attack. In an impersonation attack, an adversary tries to impersonate a legitimate device. The login and authentication phase starts by computing $(P_1, P_2)$ For computing $(P_1, P_2)$, an adversary must have access and knowledge of $(CK_1 , CK_2 )$ and $H (PID_i )$. The $(CK_1 , CK_2 )$ and $H (PID_i )$ are shared between the device and the server over a secure channel. Moreover, even if the adversary eavesdrops on $(P_1, P_2)$ of any previous login and authentication session between the device and the server, the $(CK_1 , CK_2 )$ and $H (PID_i )$ cannot be extracted from $(P_1, P_2)$ due to the computational difficulty of ECDLP. Thus, without the knowledge of $(CK_1 , CK_2 )$ and $H (PID_i )$, a valid $(P_1, P_2)$ cannot be computed as such an adversary cannot impersonate any legitimate device by computing a malicious $(P_1, P_2)$.

5.3. Traceability Attack. In a traceability attack, the message with constant values may reveal the context of communication or the communication pattern. To avoid traceability attacks, messages exchanges must be randomized by incorporating pseudorandom numbers. In the proposed protocol, the ephemeral random secrets $(S_i , S_j)$ induce the required randomness in the messages for each login session. Thus, the traceability attack is mitigated in the proposed scheme.

5.4. Message Integrity Attack. The message exchanged between the device and the server cannot be masqueraded. During the communication, the following messages are exchanged between the device and the server: $P_1 , P_2 , P_3, P_4$ and $ESK [ H (SK) ]$. Let us say an adversary intercepts the $P_1 , P_2 , P_3, P_4$ and $ESK [ H (SK) ]$ and wants to create malicious: $P'_1, P'_2, P'_3, P'_4$. However, to create $P'_1, P'_2, P'_3, P'_4$, the adversary must have access to $X_2$ and $K_2$. The adversary does not have access to $X_2$ and $K_2$. Moreover, extraction of the private key of $X_2$ from $P_1, P_2, P_3, P_4$ is having exponential time complexity due to the computational complexity of ECDLP. Thus, the integrity of $P_1, P_2, P_3, P_4$ is upheld by the computational infeasibility of ECDLP. Moreover, the message $ESK [ H (SK) ]$ can also not be altered as it is protected by a symmetric encryption technique and one-way hash.

5.5. Man-in-the-Middle Attack. In a man-in-the-middle attack, an attacker can eavesdrop, disguise, and change communication in the middle by forging the key established between the device and the server. An adversary would be successful in executing the man in the middle attack if the adversary in the middle of the communication can create malicious: $P'_1, P'_2, P'_3, P'_4$. However, to create malicious $P'_1, P'_2, P'_3, P'_4$, the adversary must be able to forge $(CK_1, CK_2)$ as follows:
\[ CK_1 = PID_i . X_3 , \]  
\[ CK_2 = PID_i . X_4 . \] (36)
Device: DI, Server: S

**Registration Phase**

- Compute: \( \text{PID}_i = K_i \cdot \text{ID}_i \cdot G \)
- Compute: \( \text{CK}_i^1 = \text{PID}_i \cdot X_1 \)
- Compute: \( \text{CK}_i^2 = \text{PID}_i \cdot X_2 \)

**Login and Authentication Phase**

- Compute: \( P_1 = (\text{CK}_i^1 + \text{CK}_i^2) \cdot S_i \cdot H(\text{PID}_i) \)
- Compute: \( P_2 = (\text{CK}_i^1 + \text{CK}_i^2) \cdot S_i \)
- Compute: \( P_1' = P_1 \cdot H^{-1}[\text{PID}_i] \)
- Compute: \( P_3 = P_1 \cdot S_s \)
- Compute: \( P_4 = (\text{CK}_i^1 + \text{CK}_i^2) \cdot S_s \)

- Process Terminated

- Compute: \( P_3' = P_3 \cdot H^{-1}[\text{PID}_i] \cdot S_i^{-1} \)

- Compute Session Key: \( SK = P_4 \cdot S_i \)
- Compute: \( V = D_{SK} [E_{SK}[H(SK)]] \)

**Device Authentication Successful**

**Server Authentication Successful**

**Figure 9:** Schematics of the proposed scheme.
To forge \((CK_1, CK_2)\), adversary must have access to \(X_5\). The adversary does not have access to \(X_5\). Moreover, from \(P_1, P_2, P_3, P_4, X_5\) cannot be extracted due to the exponential complexity of ECDLP. As the adversary cannot forge \(P_1, P_2, P_3, P_4\) and proper authentication is employed prior to the key establishment, the man-in-the-middle attack is mitigated in the proposed scheme.

6. Formal Validation of the Proposed Protocol Using Scyther

The designed protocol has been modeled using SPDL to validate its security on Scyther-Compromise under the eCK adversary model. The SPDL model of the proposed protocol is shown in Figure 10. The role Device and role Server model the communication pattern of the \(D_i\) and \(S\), respectively. The role Device initiates the login and authentication phase by sending \((P_1, P_2)\) using the send_1 function. On receiving \((P_1, P_2)\) using the recv1 function, the role server sends \((P_3, P_4)\) using send_2 functions to the device. Finally, the device sends \(ESK_i[H(SK)]\) to the server using the send_3 function. The claim_i1 in role Device and claim_r1 in role Server indicate whether the \(X_5\) and \(K_i\) must remain secret during the communication. The claim_i2 in role Device and claim_r2 in role Server indicate whether \(SK = X_5, K_i, ID_i, G, S_i, S\) can be revealed using the session key adversary rule. claim_r3, claim_r4, claim_r3, and claim_r4 is to verify whether the authentication in terms of Nisynch and Niagree is maintained.

The SPDL model of the designed protocol was simulated on the compromise 0.9 under the eCK adversary model as indicated in Figure 6. The verification results are shown in Figure 11. From Figure 11, we can infer that the protocol is safe and does not have any attack under the stringent eCK adversary model.

7. Formal Security Analysis Using BAN Logic

Burrows et al. proposed BAN logic to validate the soundness and correctness of a security protocol. This section employs BAN logic to determine the security validity of the proposed scheme. \(D\) and \(S\) denote the communicating principles, where \(K_i\) and \(X_5\) denote their private keys, respectively. The BAN notations are tabulated in Table 2 [31], and the BAN postulates are given in Table 3. Besides that, as derived in [33], synthesis rules are tabulated in Table 4.

7.1. Assumptions

(A1) \(D|\equiv \rightarrow CK_1, CK_2 S\)
(A2) \(S|\equiv \rightarrow CK_1, CK_2 D\)
(A3) \(D|\equiv \#(S_1)\)
(A2) \(S|\equiv \#(S_2)\)
(A5) \(S|\equiv D| \rightarrow S_1\)
(A6) \(D|\equiv S| \rightarrow S_5\)

7.2. Idealized Form

\[
D \rightarrow S : \{P_1, P_2\}_{K_i}, MSG1
S \rightarrow D : \{P_3, P_4\}_{X_5}, MSG2
\]

7.3. Goals

G1) \(|S| \equiv D \rightarrow^5 KS G2) \(|S| \equiv D| \equiv D \rightarrow^5 KS
G3) \(|D| \equiv D \rightarrow^5 KS G4) \(|D| \equiv S| \equiv D \rightarrow^5 KS

7.4. BAN Analysis. From (MSG1), we get that

(1) \(D| \equiv \{P_1, P_2\}_{K_i}\)
(2) \(D \equiv \{P_1, P_2\}_{K_i}\)
From (2), (A1), and (R1), we get
(3) \(|D| \sim \{P_1, P_2\}_{K_i}\)
\(S_1\) is a part of \(P_1, P_2\); thus, as per (A3) and (R6), we get
(4) \(|D| \equiv \#(P_1, P_2)\)
From (3) and (5), we get
(5) \(|S| \equiv D| \sim P_1, P_2\)
From (7) and (S4), we get
(6) \(|S| \equiv \#(P_1, P_2)\)
From (3) and (6) on applying (R2), we get
(7) \(|S| \equiv D| \equiv P_1, P_2\)
\(S_1\) is the part of the \(P_1, P_2\); thus, on applying (R5), we get
(8) \(|S| \equiv D| \equiv S_1\)
Now as per (A5), (8), and (R3), we get
(9) \(|S| \equiv S_1\)
From (S3) and (3), we get
(10) \(|S| \equiv D| \equiv S_1\)
From (A3) and (10), we get
(11) \(|S| \equiv D| \sim S_1\)
As per (R4) and (11), we get
(12) \(|S| \equiv \#(S_1)\)
\(S_1\) is a part of \((SK)\); thus, as per (R6), we get
(13) \(|S| \equiv \#(SK)\)
From (13), (8), and (R7), we get
(14) \(|S| \equiv D \rightarrow^5 KS (Goal G1)\)
Due to the symmetry of the protocol,
(15) \(|S| \equiv D \equiv S \rightarrow^5 KD (Goal G2)\)
From (MSG2), we infer that
(16) \(|S| \equiv \{P_3, P_4\}_{X_5}\)
(17) \(S \equiv \{P_3, P_4\}_{X_5}\)
From (17), (A2), (R1), we get
(18) \(|D| \equiv S| \sim \{P_3, P_4\}_{X_5}\)
\(S_1\) is a part of \(P_3, P_4\); thus, as per (A4) and (R6), we get
(19) \(|S| \equiv \#(P_3, P_4)\)
From (17) and (19), we get
<table>
<thead>
<tr>
<th>Protocol description</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 hashfunction MUL;</td>
<td></td>
</tr>
<tr>
<td>2 hashfunction H;</td>
<td></td>
</tr>
<tr>
<td>3 hashfunction ADD;</td>
<td></td>
</tr>
<tr>
<td>4 const G: Nonce;</td>
<td></td>
</tr>
<tr>
<td>5 const KI, KS, KS1, KS2, SI, SS;</td>
<td></td>
</tr>
<tr>
<td>6 secret N1, N2;</td>
<td></td>
</tr>
<tr>
<td>7 macro PID=MUL (KI, Device, G);</td>
<td></td>
</tr>
<tr>
<td>8 macro CK1=MUL (PID, KS1);</td>
<td></td>
</tr>
<tr>
<td>9 macro CK2=MUL (PID, KS2);</td>
<td></td>
</tr>
<tr>
<td>10 macro P1=MUL (ADD(CK1, CK2), SI, H (PID));</td>
<td></td>
</tr>
<tr>
<td>11 macro P2=MUL (ADD(CK1, CK2), SI);</td>
<td></td>
</tr>
<tr>
<td>12 macro P3=MUL (ADD(CK1, CK2), SS, H (PID));</td>
<td></td>
</tr>
<tr>
<td>13 macro P4=MUL (ADD(CK1, CK2), SI);</td>
<td></td>
</tr>
</tbody>
</table>

protocol improved(Device, Server)
{
  role Device
  {
    fresh SI, msg: Nonce;
    var SS: Nonce;
    secret KI, KS, KS1, KS2, SI, SS;
    send_1 (Device, Server, P1, P2);
    recv_2 (Server, Device, P3, P4);
    send_3 (Device, Server, {msg} MUL (KI, KS, SI, SS, Device, G));
    claim_i1 (Device, Secret, SI);
    claim_i2 (Device, SKR, MUL (KI, KS, SI, SS, Device, G));
    claim_i3 (Device, Niagree);
    claim_i4 (Device, Nisynch);
  }
  role Server
  {
    fresh SS: Nonce;
    var SI, msg: Nonce;
    fresh Message 1: Nonce;
    secret KI, KS, KS1, KS2, SI, SS;
    recv_1 (Device, Server, P1, P2);
    send_2 (Server, Device, P3, P4);
    recv_3 (Device, Server, {msg} MUL (KI, KS, SI, SS, Device, G));
    claim_r1 (Server, Secret, SS);
    claim_r2 (Server, SKR, MUL (KI, KS, SI, SS, Device, G));
    claim_r3 (Server, Niagree);
    claim_r4 (Server, Nisynch);
  }
}

Figure 10: SPDL model of the proposed scheme.

Figure 11: Scyther verification results for the proposed scheme.
Table 2: BAN symbols.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D</td>
<td>\equiv MSG$</td>
</tr>
<tr>
<td>$D \Leftarrow MSG$</td>
<td>$D$ sent the message $M$ in past</td>
</tr>
<tr>
<td>$D</td>
<td>\sim MSG$</td>
</tr>
<tr>
<td>$D \mapsto F$</td>
<td>$D$ has jurisdiction over $V$</td>
</tr>
<tr>
<td>$#(MSG)$</td>
<td>$M$ is fresh</td>
</tr>
<tr>
<td>$D \mapsto P \cdot r D$</td>
<td>$P_1$ is the public parameter calculated using the private key of $D$</td>
</tr>
<tr>
<td>$D \mapsto ^r K S$</td>
<td>$SK$ is the key used to decrypt $MSG$</td>
</tr>
<tr>
<td>$[X]_{SK}$</td>
<td>$SK$ is the key between $D$ and $S$</td>
</tr>
<tr>
<td>$(F1/F2)$</td>
<td>If F1 is true, then F2 is true</td>
</tr>
</tbody>
</table>

Table 3: BAN postulates.

<table>
<thead>
<tr>
<th>Rule no.</th>
<th>Name</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Message-meaning rule</td>
<td>$(D</td>
</tr>
<tr>
<td>R2</td>
<td>Nonce verification rule</td>
<td>$(D</td>
</tr>
<tr>
<td>R3</td>
<td>Jurisdiction rule</td>
<td>$(D</td>
</tr>
<tr>
<td>R4</td>
<td>Seeing rule</td>
<td>$(D \Leftarrow MSG1, D \Leftarrow MSG2 \mapsto D \Leftarrow MSG1, MSG2))$</td>
</tr>
<tr>
<td>R5</td>
<td>Belief rule</td>
<td>$(D</td>
</tr>
<tr>
<td>R6</td>
<td>Freshness rule</td>
<td>$(D</td>
</tr>
<tr>
<td>R7</td>
<td>Session key rule</td>
<td>$(D</td>
</tr>
</tbody>
</table>

Table 4: Synthesis rules.

<table>
<thead>
<tr>
<th>Rule no.</th>
<th>Synthesis rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>$D \Leftarrow MSG1 D \Leftarrow (MSG1, MSG2)$</td>
</tr>
<tr>
<td>S2</td>
<td>$D</td>
</tr>
<tr>
<td>S3</td>
<td>$D</td>
</tr>
<tr>
<td>S4</td>
<td>$D</td>
</tr>
</tbody>
</table>

(20) $D | \equiv S | \sim P_3, P_4$
From (20) and (S4), we get
(21) $D | \equiv \#(P_3, P_4)$
From (18) and (21) on applying (R2), we get
(22) $D | \equiv S | \sim P_3, P_4$
$S_5$ is the part of the formulae $P_3, P_4$; thus, on applying (R5), we get
(23) $D | \equiv S | \equiv S_5$
Now, as per (A6) and (23) and (R3), we get
(24) $D | \equiv S_5$
From (SR3) and (18), we get
(25) $D | \equiv S | \sim S_5$
From (A4) and (25), we get
(26) $D | \equiv S | \sim S_5$
As per (R4) and (26), we get
(27) $D | \equiv \#(S_5)$
From (R6), we get
(28) $D | \equiv \#(SK)$
From (28) and (22) and on applying (R7), we get
(29) $D | \equiv D \mapsto ^r K S (Goal G3)$
Due to the symmetry of the protocol,
(30) $S | \equiv D | \equiv D \mapsto ^r K S (Goal G4)$

8. Comparison with Other Schemes

8.1. Security Comparison. The security comparison of the proposed scheme with relevant existing schemes is shown in Table 5. Kalra and Sood [14] do not support any security resistance. Chang et al. [15] are not resilient to traceability and impersonation attack. Kumari et al. [16] support only AK3 and AK4 features. Among the existing schemes, Rostampour et al.’s [11] scheme is the only scheme that supports AK1–AK5. However, from Table 5, we can infer that Kalra and Sood [14], Chang et al. [15], Kumari et al. [16], Wang et al. [17], and Rostampour et al. [11] have not been evaluated under the eCK adversary model. The formal validation of Rostampour et al. [11] under eCK adversary indicates that the scheme is not safe under the eCK model. The proposed scheme supports all the security requirements from AK1 to AK5 and is also safe under the eCK model.

8.2. Computation and Communication Overhead. The comparison in terms of computational and communication overhead is shown in Table 6. The time complexities of various critical operations considered include $T_{ECM}$: scalar multiplication, $T_{EADD}$: point addition, $T_{HASH}$: one-way hash, $T_{SENC}/T_{SDEC}$: symmetric encryption, and $T_{inv}$: multiplicative inverse. $T_{ECM}$ is the most computationally intense operation. As IoT devices are resource constraints in nature, the computational overhead on these devices plays an important role in determining the efficiency of the scheme as the server is considered computationally more powerful. From Table 6, we can infer that a computational overhead of $4 T_{ECM}$ for the IoT device and 8 $T_{ECM}$ in total is required in the proposed scheme. The highest device overhead is that of the
However, from Table 5, we can infer that none of the security specifications are supported by Kalra and Sood [14]. In terms of scalar multiplication (TECM) overhead, the proposed scheme is required the same no of scalar multiplication as compared to other schemes considered for comparison. The communication cost is estimated based on the number of bits transmitted. We consider an elliptic curve \(E(a,b)\) of 160 bit. The size of an ECC point \(P(x,y)\) on the 160 bit curve is 320 bits (\(X[160], Y[160]\)). The symmetric encryption considered generates a ciphertext of 128 bit. In the proposed scheme, 3 messages are exchanged that include (\(P_1, P_2\), \(P_3, P_4\)), and \(E_{SK}(H(SK))\). The 3 messages require \((320 + 320),(320 + 320) + 128\) = 1,408 bits.

8.3. Energy Overhead. To compare estimated energy consumed on an IoT device, the time consumed by various critical operations as indicated in [34] on a MicaZ mote [35] is shown in Table 7. The energy is consumed using \(E = V \cdot I \cdot T\), where \(V\) is the voltage, \(I\) is the current drawn, and \(T\) is the time taken for the operation. For a MicaZ mote, \(V = 3\) V and \(I = 8\) mA. The proposed scheme has an energy overhead of 278.16 mJ for the IoT device. The comparison of the energy analysis of the proposed scheme with the relevant existing scheme is shown in Figure 12. The highest energy overhead is that of the Rostampour et al. scheme with 477.6 mJ. Rostampour et al. is not secure under the eCK adversary model. Though Kalra and Sood [12] have the lowest energy overhead for the IoT device, it has a high communication overhead and suffers from various security limitations that include AK1, AK2, AK3, AK4, and AK5. Kalra and Sood [12] have not been evaluated under the eCK model. Chang et al. [15], Kumari et al. [16], and Wang et al. [17] have the energy overheads of 271.44 mJ, 271.2 mJ, and 271.68 mJ, respectively. However, Chang et al. [15], Kumari et al. [16], and Wang et al. [17] do not suffice all security requirements and have not been modeled under the eCK model. The designed scheme supports all the security specifications and is the only scheme that is formally validated under the eCK adversary model. Thus, with the computational requirement of 278.16 mJ and

<table>
<thead>
<tr>
<th>Scheme</th>
<th>AK1</th>
<th>AK2</th>
<th>AK3</th>
<th>AK4</th>
<th>AK5</th>
<th>eCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalra and Sood [14]</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Chang et al. [15]</td>
<td>✘</td>
<td>✘</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Kumari et al. [16]</td>
<td>✘</td>
<td>✘</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Wang et al. [17]</td>
<td>✘</td>
<td>✘</td>
<td>✔</td>
<td>✘</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Rostampour et al. [11]</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✘</td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

AK1: traceability attack; AK2: impersonation attack; AK3: replay attack; AK4: message integrity attack; and AK5: man-in-the-middle attack.

Table 6: Comparison of computational and communication overhead.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Computation overhead</th>
<th>Communication overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kalra and Sood [14]</td>
<td>3 TECM + 4 THASH</td>
<td>8 TECM + 9 THASH</td>
</tr>
<tr>
<td>Chang et al. [15]</td>
<td>4 TECM + 4 THASH</td>
<td>8 TECM + 8 THASH</td>
</tr>
<tr>
<td>Kumari et al. [16]</td>
<td>4 TECM + 3 THASH</td>
<td>8 TECM + 7 THASH</td>
</tr>
<tr>
<td>Wang et al. [17]</td>
<td>7 TECM + TADD</td>
<td>13 TECM + 2 TADD</td>
</tr>
<tr>
<td>Rostampour et al. [11]</td>
<td>TECM + TADD + 2</td>
<td>8 TECM + 2 TADD + 2</td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>T_HASH + T_SENC/T_SDEC + T_inv</td>
<td>T_HASH + T_SENC/T_SDEC + T_inv</td>
</tr>
</tbody>
</table>

Table 7: Time consumed for critical operations [34].

<table>
<thead>
<tr>
<th>Operation</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>TECM</td>
<td>2.82</td>
</tr>
<tr>
<td>T_SENC/T_SDEC</td>
<td>0.000029</td>
</tr>
<tr>
<td>T_HASH</td>
<td>0.0091</td>
</tr>
<tr>
<td>T_inv</td>
<td>0.14</td>
</tr>
<tr>
<td>T_EADD</td>
<td>0.16</td>
</tr>
</tbody>
</table>

Rostampour et al. scheme with 7 TECM and the lowest is that of Kalra and Sood [14]. However, from Table 5, we can infer that none of the security specifications are supported by Kalra and Sood [14]. In terms of scalar multiplication (TECM) overhead, the proposed scheme is required the same no of scalar multiplication as compared to other schemes considered for comparison. The communication cost is estimated based on the number of bits transmitted. We consider an elliptic curve \(E(a,b)\) of 160 bit. The size of an ECC point \(P(x,y)\) on the 160 bit curve is 320 bits (\(X[160], Y[160]\)). The symmetric encryption considered generates a ciphertext of 128 bit. In the proposed scheme, 3 messages are exchanged that include (\(P_1, P_2\), \(P_3, P_4\)), and \(E_{SK}(H(SK))\). The 3 messages require \((320 + 320),(320 + 320) + 128\) = 1,408 bits.

Figure 12: Energy overhead comparison.
communication overhead of 1,408 bits, the proposed scheme supports all the security requirements and is also safe under the eCK adversary.

9. Conclusion

Authentication between IoT devices in the perception layer and the cloud server is critical for developing secure IoT-based smart applications. The existing schemes presented for authenticated key agreements between IoT devices and the cloud services have not been validated using the eCK adversary model. Recently, Rostampour et al. presented a scheme for authenticated key agreement between IoT devices and the cloud server that is secure under the Dolev and Yoo model. A formal security validation of Rostampour et al. has been carried out using Scyther-Compromise under the eCK model. The verification indicates that the scheme is susceptible to session key disclosure attacks under the eCK model. A lightweight ECC-based authentication technique for IoT devices and the cloud server has been presented in this paper. The Scyther-Compromise simulation indicates that the proposed scheme is secure under the eCK model. BAN logic analysis and evaluation indicate that the design of the proposed scheme is sound and correct. The overhead analysis indicates that the proposed scheme requires 199.44 mJ and 512 bits less in terms of energy and communication overhead as compared to the scheme presented by Rostampour et al.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

References


[34] U. Iqbal and A. Hussain Mir, "Secure and practical access control mechanism for WSN with node privacy," *Journal of King Saud University-Computer and Information Sciences*, 2020.