Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 7707543, 17 pages
https://doi.org/10.1155/2022/7707543

Research Article

A Novel Secure Authentication Protocol for IoT and Cloud Servers

Ummer Iqbal,[1] **Aditya Tandon**,[2] **Sonali Gupta**,[3] **Arvind R. Yadav**,[4] **Rahul Neware**,[5] **and Fraol Waldamichael Gelana**[6]

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

Academic Editor: Shalli Rani

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 session 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 motoring 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 offline and online 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 nodes 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.

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

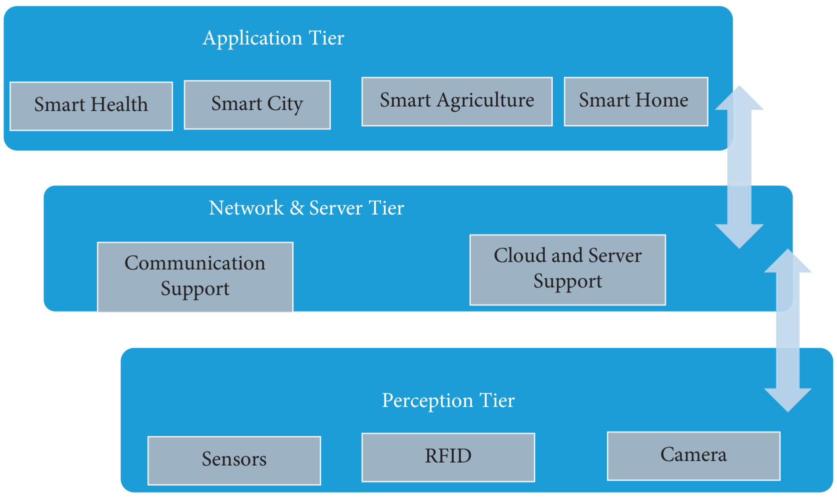


FIGURE 1: Three-tier architecture of IoT applications.

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.

TABLE 1: Notations.

Notation	Meaning
$E_P(a, b)$	Elliptical curve
S	Server
D_i	i-th IoT device
ID_i	Identity of D_i
ID_s	Identity of S
R_i	The random number at the server
N_i	Random number at the device
X_S	Private key of S
K_I	Private key of D_i
S_i	Ephemeral secret of D_i
S_s	Ephemeral secret of S
ET	Expiration time
$G(x, y)$	Generator point of $E_P(a, b)$
SK	Session key between S and D_i
H	Collision resistant hash function
$\oplus$	Xor operation
$P+Q$	Point addition
$X.G$	Scalar multiplication

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 elliptical 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 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]:

$$E_P(a, b)\colon\ y^2 = x^3 + ax + b, \tag{1}$$

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

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:

$$C(x_R, y_R) = A(x_A, y_A) + B(x_B, y_B), \tag{2}$$

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:

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

Where,

$$\lambda = \begin{cases} \dfrac{(y_B - y_A)}{(x_B - x_A)} \bmod p, \text{if } A! = B, \\[2ex] \dfrac{3x_A^2 + a}{2y_A} \bmod p, \text{if } A = B. \end{cases} \tag{3}$$

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

$$\begin{aligned} X.A(x, y) = {} & A(x, y) + A(x, y) + A(x, y) \\ & + \ldots + A(x, y), \\ & X \text{ times.} \end{aligned} \tag{4}$$

(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 a*live* 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 Yoa. 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):

$$PID_i = X_i \oplus ID_i. \tag{5}$$

D_i sends PID_i to the server Sas follows:

$$D_i \longrightarrow S\colon PID_i. \tag{6}$$

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

$$CK_i = H(R_i \oplus X_s \oplus ET \oplus PID_i), \tag{7}$$

$$CK_i' = CK_i.G. \tag{8}$$

Server stores PID_i, ET, and CK_i in a secure database.

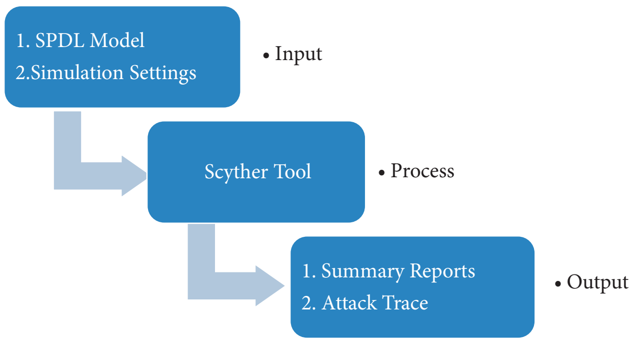


FIGURE 2: Basic architecture of Scyther.

(iii) The server sends CK'_i to the device as follows:

$$S \longrightarrow Di : CK'_i. \tag{9}$$

The device Di stores CK_i along with generated PID_i.

3.1.2. Login and Authentication Phase

(i) D_i generates a random number N_i and computes (10)–(12):

$$P_1 = N_1.G, \tag{10}$$

$$P_2 = N_1.CK'_i, \tag{11}$$

$$PPID_i = PID_i.P_1. \tag{12}$$

And sends it to the server as follows:

$$D_i \longrightarrow S\colon P_1, P_2, PPID_i. \tag{13}$$

(ii) The server finds the entry of D_i in its database and verifies $P_2 \neq CK_i.P_1$. 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):

$$P_3 = N_2.G, \tag{14}$$

$$P_4 = (CK'_i + PPID_i) + (N_2.P1). \tag{15}$$

Also, it sends it to the device as follows:

$$S \longrightarrow D_i\colon P_3, P_4. \tag{16}$$

(iv) Device verifies $P_4 ? = (PPID_i.CK'_i) + (N_1.P_2)$. 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:

$$V_i = PPID_i.P_3, \tag{17}$$

$$D_i \longrightarrow S\colon V_i. \tag{18}$$

(v) The server receives V_i and verifies $V_i \neq N_2.\ PPID_i$. 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.

3.2. Formal Validation of Rostampour Et Al.'s Scheme Using Scyther under the eCK Adversary Model.

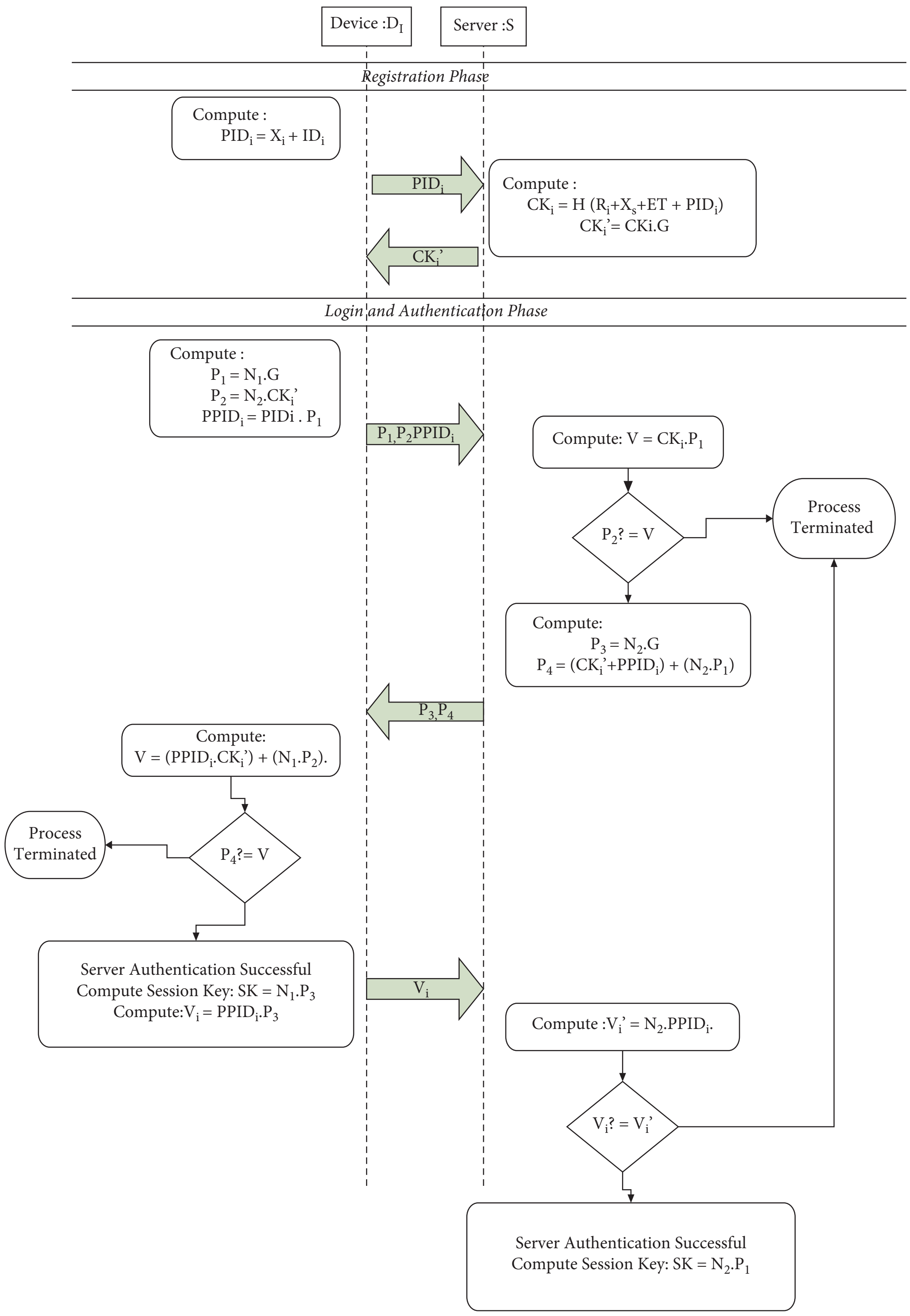
The SPDL model of the Rostampour et al. protocol is shown in Figure 4. The SPDL modeling comprises two roles: role I and role R. role I models the communication of the D_i, and role R models the S. D_i sends P_1, P_2, $PPID_i$ to the S using send_1(). The S receives the P_1, P_2, $PPID_i$ using recv_1() function. S sends P_3, P_4 to the D_i using send_2(). The D_i receives the P_3, P_4 using recv_2() function. Finally, D_i sends V_i to the S using send_3(). The S receives the V_i using recv_3() function. The claim_i1 in role I and claim_r1 in role I and R indicates that the N_1, N_2 must remain secret during the communication. The claim_i2 and claim_r2 in roles I and R indicate that $SK = N_1, N_2, G$ can be revealed using the session key adversary rule. claim_r3, claim_r4, claim_r3, and claim_r4 are to verify whether the authentication in terms of Nisynch and Niagree is maintained.

The SPDL model has been initially executed under Dolev and Yoa setting as shown in Figure 5. The Scyther-Compromise verification results of the Rostampour et al. scheme under the Dolev and Yoa setting indicate that the scheme is safe and does not have any attacks. Subsequently, the model was executed under the eCK adversary setting shown in Figure 6. The Scyther verification results under eCK are shown in Figure 7.

The results indicate that the scheme is not safe. The attack trace is shown in Figure 8. The attack trace indicates that the Rostampour et al. scheme under the eCK adversary model is not resilient to session key reveal attack; thus, the adversary can reveal the session key. This is primarily due to the design issue in the Rostampour et al. scheme for computing the session key. In the Rostampour et al. scheme, the session key is computed as $SK = N1.N2.G$. The session key (SK) depends only on short-term session secrets $N1$ and $N2$. SK must also be dependent on long-term term secrets so that the reveal of short-term secrets does reveal session key.

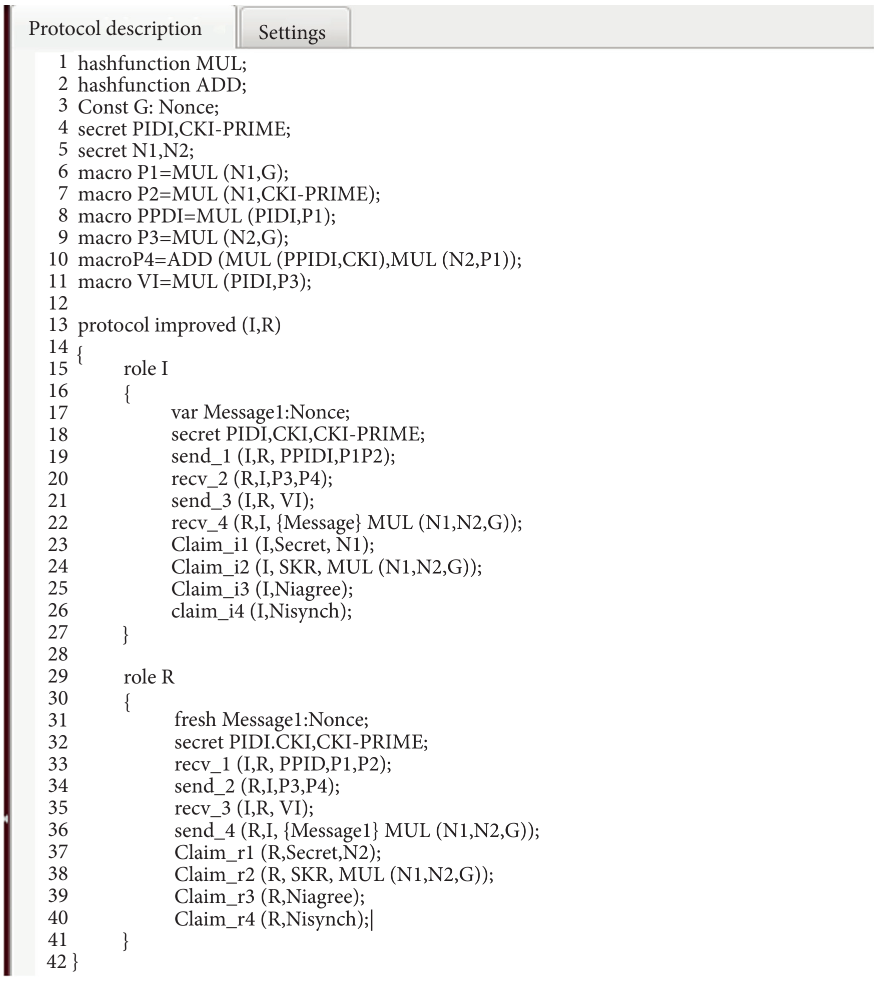
4. Proposed Scheme

The proposed scheme comprises two phases that include



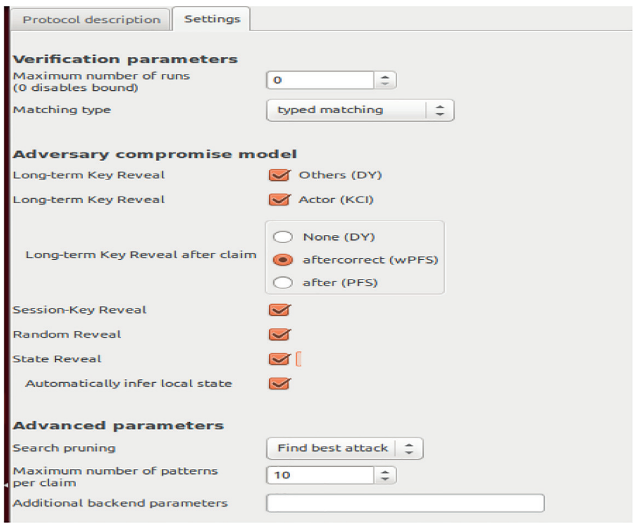
Device :D_I
Server :S
Registration Phase
Compute : $PID_i = X_i + ID_i$
PID_i
Compute : $CK_i = H(R_i + X_s + ET + PID_i)$ $CK_i' = CKi.G$
CK_i'
Login and Authentication Phase
Compute : $P_1 = N_1.G$ $P_2 = N_2.CK_i'$ $PPID_i = PIDi . P_1$
$P_1, P_2 PPID_i$
Compute: $V = CK_i.P_1$
$P_2 ? = V$
Process Terminated
Compute: $P_3 = N_2.G$ $P_4 = (CK_i' + PPID_i) + (N_2.P_1)$
P_3, P_4
Compute: $V = (PPID_i.CK_i') + (N_1.P_2)$.
$P_4 ?= V$
Process Terminated
Server Authentication Successful
Compute Session Key: $SK = N_1.P_3$
Compute:$V_i = PPID_i.P_3$
V_i
Compute :$V_i' = N_2.PPID_i$.
$V_i ? = V_i'$
Server Authentication Successful
Compute Session Key: $SK = N_2.P_1$

FIGURE 3: Schematics of the Rostampour et al. scheme.



```
hashfunction MUL;
hashfunction ADD;
Const G: Nonce;
secret PIDI,CKI-PRIME;
secret N1,N2;
macro P1=MUL (N1,G);
macro P2=MUL (N1,CKI-PRIME);
macro PPDI=MUL (PIDI,P1);
macro P3=MUL (N2,G);
macroP4=ADD (MUL (PPIDI,CKI),MUL (N2,P1));
macro VI=MUL (PIDI,P3);

protocol improved (I,R)
{
        role I
        {
                var Message1:Nonce;
                secret PIDI,CKI,CKI-PRIME;
                send_1 (I,R, PPIDI,P1P2);
                recv_2 (R,I,P3,P4);
                send_3 (I,R, VI);
                recv_4 (R,I, {Message} MUL (N1,N2,G));
                Claim_i1 (I,Secret, N1);
                Claim_i2 (I, SKR, MUL (N1,N2,G));
                Claim_i3 (I,Niagree);
                claim_i4 (I,Nisynch);
        }

        role R
        {
                fresh Message1:Nonce;
                secret PIDI.CKI,CKI-PRIME;
                recv_1 (I,R, PPID,P1,P2);
                send_2 (R,I,P3,P4);
                recv_3 (I,R, VI);
                send_4 (R,I, {Message1} MUL (N1,N2,G));
                Claim_r1 (R,Secret,N2);
                Claim_r2 (R, SKR, MUL (N1,N2,G));
                Claim_r3 (R,Niagree);
                Claim_r4 (R,Nisynch);
        }
}
```

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



Verification parameters

Maximum number of runs (0 disables bound)	0
Matching type	typed matching

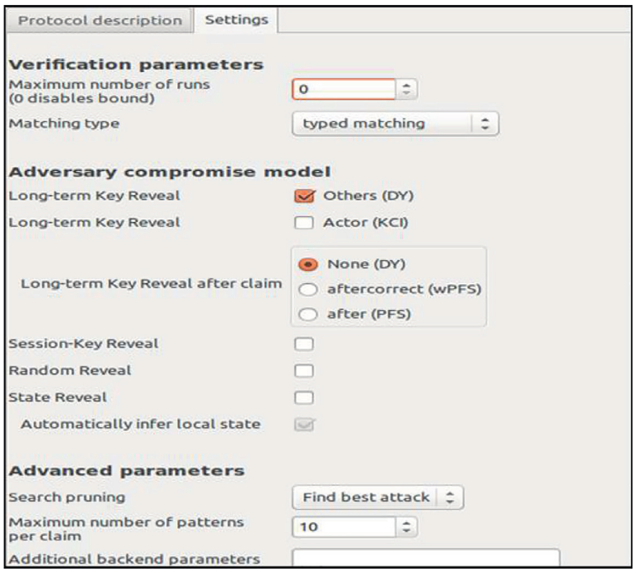
Adversary compromise model

Long-term Key Reveal	✓ Others (DY)
Long-term Key Reveal	✓ Actor (KCI)
Long-term Key Reveal after claim	None (DY) [✓ aftercorrect (wPFS)] after (PFS)
Session-Key Reveal	✓
Random Reveal	✓
State Reveal	✓
Automatically infer local state	✓

Advanced parameters

Search pruning	Find best attack
Maximum number of patterns per claim	10
Additional backend parameters	

Figure 5: Scyther settings for Dolev and Yoa adversary models.



Protocol description | Settings

Verification parameters

Maximum number of runs (0 disables bound)	0
Matching type	typed matching

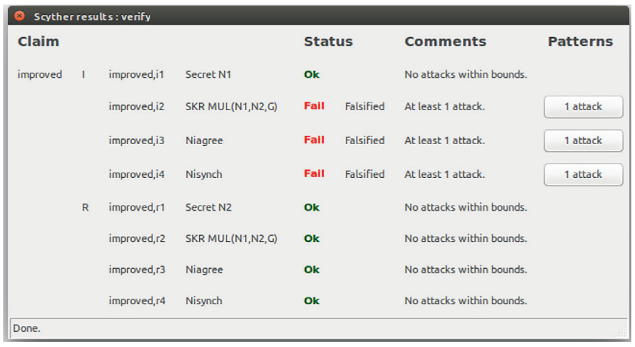
Adversary compromise model

Long-term Key Reveal	[✓ Others (DY)]
Long-term Key Reveal	Actor (KCI)
Long-term Key Reveal after claim	[✓ None (DY)] aftercorrect (wPFS) after (PFS)
Session-Key Reveal	
Random Reveal	
State Reveal	
Automatically infer local state	✓

Advanced parameters

Search pruning	Find best attack
Maximum number of patterns per claim	10
Additional backend parameters	

Figure 6: Scyther settings for the eCK adversary model.



Scyther results : verify

Claim				Status		Comments	Patterns
improved	I	improved,i1	Secret N1	Ok		No attacks within bounds.	
		improved,i2	SKR MUL(N1,N2,G)	Fail	Falsified	At least 1 attack.	1 attack
		improved,i3	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		improved,i4	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	R	improved,r1	Secret N2	Ok		No attacks within bounds.	
		improved,r2	SKR MUL(N1,N2,G)	Ok		No attacks within bounds.	
		improved,r3	Niagree	Ok		No attacks within bounds.	
		improved,r4	Nisynch	Ok		No attacks within bounds.	

Done.

Figure 7: Scyther verification results of the Rostampour et al. scheme under the eCK model.

(1) Registration phase

(2) Login and authentication phase

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 $I\ Di$.

(ii) The device calculates PID_i as in (19) and sends PID_i it to the server:

$$PID_i = K_i.ID_i.G,$$
$$D_i \longrightarrow S\colon PID_i. \tag{19}$$

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

$$CK_i^1 = PID_i.X_S^1, \tag{20}$$

$$CK_i^2 = PID_i.X_S^2. \tag{21}$$

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

$$S \longrightarrow D_i\colon CK_i^1, CK_i^2. \tag{22}$$

(v) The device receives and stores the parameters CK_i^1, CK_i^2 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):

$$P_1 = \left(CK_i^1 + CK_i^2\right).S_i.H\left(PID_i\right), \tag{23}$$

$$P_2 = \left(CK_i^1 + CK_i^2\right).S_i. \tag{24}$$

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

$$D_i \longrightarrow S\colon P_1, P_2. \tag{25}$$

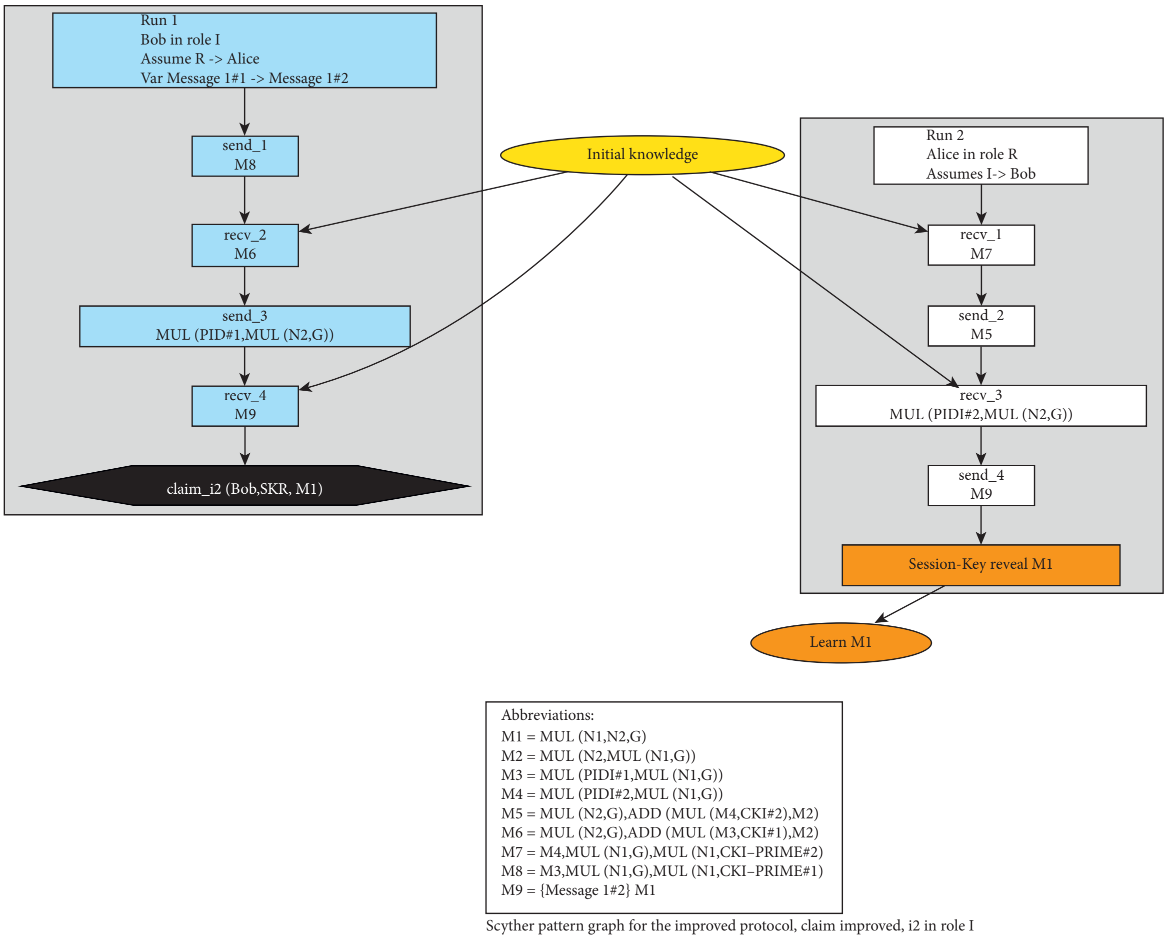


Figure 8: Attack trace under the eCK model for Rostampour et al.' scheme.

(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):

$$P_3 = P_1.S_s, \tag{26}$$

$$P_4 = \left(CK_i^1 + CK_i^2\right).S_s. \tag{27}$$

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

$$SK = P_2.S_s, \tag{28}$$

$$\begin{aligned} SK &= \left(CK_i^1 + CK_i^2\right)S_i.S_s, \\ SK &= \left(PID_i\,.\,X_S^1 + PID_i.\,X_S^2\right)S_i\,.S_s, \\ SK &= \left(X_S^1 + X_S^2\right).\,PID_i.\,S_i.S_s, \\ SK &= X_S.\,K_i.ID_i.G.S_i.S_s. \end{aligned} \tag{29}$$

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

$$S \longrightarrow D_i:\ P_3, P_4. \tag{30}$$

(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 $P_3' = P_3.[H(PID_i)]^{-1}.S_i^{-1}$.

(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:

$$SK = P_4.S_i, \tag{31}$$

$$\begin{aligned} SK &= \left(CK_i^1 + CK_i^2\right)S_s.S_i, \\ SK &= \left(PID_i.\, X_S^1 + PID_i.X_S^2\right)S_s.S_i, \\ SK &= \left(X_S^1 + X_S^2\right).PID_iS_i.S_s, \\ SK &= X_S.\, K_i.\, ID_i.G.S_i.S_s. \end{aligned} \tag{32}$$

(viii) The device encrypts $E_{SK}[H(SK)]$ and sends it to the server as follows:

$$D_i \longrightarrow S\colon E_{SK}[H(SK)]. \tag{33}$$

(ix) The server decrypts $E_{SK}[H(SK)]$ as $V = D_{SK}[E_{SK}[SK]]$ 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:

$$\begin{aligned} Di &\longrightarrow S\colon P_1, P_2, \\ S &\longrightarrow D_i\colon P_3, P_4, \\ D_i &\longrightarrow S\colon E_{SK}[H(SK)]. \end{aligned} \tag{34}$$

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 $E_{SK}[H(SK)]$ for a replay attack. Let an adversary (A) replay stored (P_1, P_2) of the previous session to gain illegitimate access as follows:

$$A \longrightarrow S\colon P_1, P_2. \tag{35}$$

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 $\mathrm{PID_i}$ 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 legitimate session key SK of the current session, $E_{SK}[H(SK)]$ cannot be computed by the adversary. Moreover, if the adversary replays the old $E_{SK}[H(SK)]$, 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_i^1, CK_i^2) and $H(PID_i)$. The (CK_i^1, CK_i^2) and $H(PID_i)$ are shared between the device and the server over a secure channel. Moreover, even if the adversary eavesdrop on (P_1, P_2) of any previous login and authentication session between the device and the server, the (CK_i^1, CK_i^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_i^1, CK_i^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_s) 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 $E_{SK}[H(SK)]$. Let us say an adversary intercepts the P_1, P_2, $P_3\,P_4$ and $E_{SK}[H(SK)]$ and wants to create malicious: $P_1^S, P_2^S, P_3^S, P_4^S$. However, to create $P_1^S, P_2^S, P_3^S, P_4^S$, the adversary must have access to X_S and K_i. The adversary does not have access to X_S and K_i. Moreover, extraction of the private key of X_S 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 $E_{SK}[H(SK)]$ 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: $P_1^S, P_2^S, P_3^SP_4^S$. However, to create malicious $P_1^S, P_2^S, P_3^SP_4^S$, the adversary must be able to forge (CK_i^1, CK_i^2) as follows:

$$\begin{aligned} CK_i^1 &= PID_i.X_S^1, \\ CK_i^2 &= PID_i.X_S^2. \end{aligned} \tag{36}$$

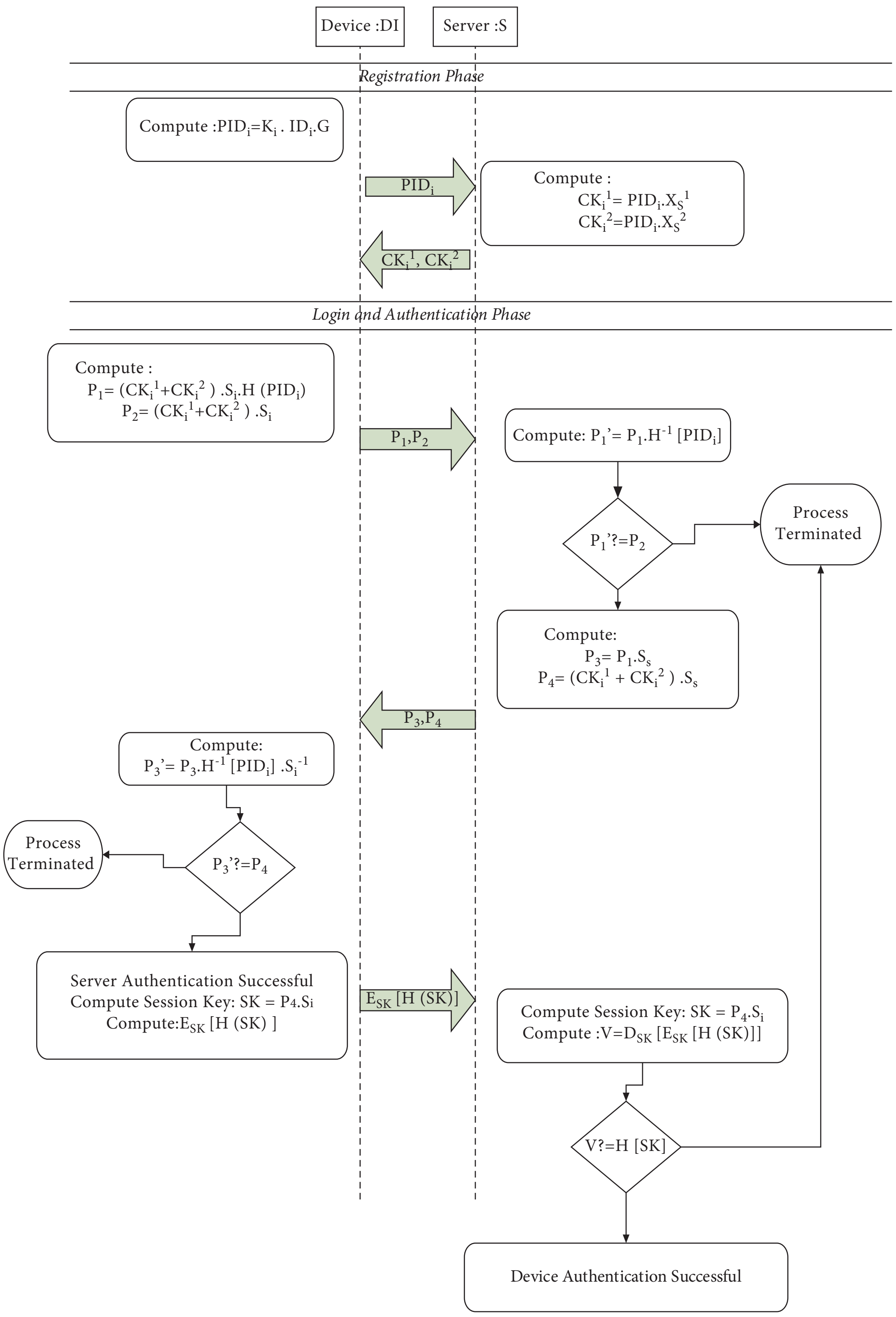

Device :DI
Server :S
Registration Phase
Compute :PID_i=K_i . ID_i.G
PID_i
Compute : CK_i^1= PID_i.X_S^1 CK_i^2=PID_i.X_S^2
CK_i^1, CK_i^2
Login and Authentication Phase
Compute : P_1= (CK_i^1+CK_i^2) .S_i.H (PID_i) P_2= (CK_i^1+CK_i^2) .S_i
P_1,P_2
Compute: P_1'= P_1.H^{-1} [PID_i]
P_1'?=P_2
Process Terminated
Compute: P_3= P_1.S_s P_4= (CK_i^1 + CK_i^2) .S_s
P_3,P_4
Compute: P_3'= P_3.H^{-1} [PID_i] .S_i^{-1}
Process Terminated
P_3'?=P_4
Server Authentication Successful Compute Session Key: SK = P4.Si Compute:E_{SK} [H (SK)]
E_{SK} [H (SK)]
Compute Session Key: SK = P_4.S_i Compute :V=D_{SK} [E_{SK} [H (SK)]]
V?=H [SK]
Device Authentication Successful

Figure 9: Schematics of the proposed scheme.

To forge (CK_i^1, CK_i^2), adversary must have access to X_S. The adversary does not have access to X_S. Moreover, from P_1, P_2, P_3 P_4, X_S 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 $E_{SK}[H(SK)]$ to the server using the send_3 function. The claim_i1 in role Device and claim_r1 in role Server indicate that the X_S and K_i must remain secret during the communication. The claim_i2 in role Device and claim_r2 in role Server indicate whether $SK = X_S.K_i.ID_i.G.S_i.S_s$ 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_S 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) $D| \equiv \; \longrightarrow^{CK_i^1, CK_i^2} S$

(A2) $S| \equiv \; \longrightarrow^{CK_i^1, CK_i^2} D$

A3) $D| \equiv \#(S_I)$

(A2) $S| \equiv \#(S_S)$

A5) $S| \equiv \; D| \longrightarrow S_I$

(A6) $D| \equiv \; S| \longrightarrow \; S_S$

7.2. Idealized Form

$D \longrightarrow S$; $\{P_1, P_2\}_{K_I}$ $MSG1$

$S \longrightarrow D$; $\{P_3, P_4\}_{X_S}$ $MSG2$

7.3. Goals

G1) $S| \equiv \; D \longrightarrow^S K\; S$ G2) $S| \equiv \; D| \equiv D \longrightarrow^S K\; S$

G3) $D| \equiv \; D \longrightarrow^S K\; S$ G4) $D| \equiv S| \equiv D \longrightarrow^S K\; S$

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

(1) $D| \equiv \{P_1, P_2\}_{K_I}$

(2) $D \Leftarrow \{P_1, P_2\}_{K_I}$

From (2), (A1), and (R1), we get

(3) $D| \equiv \; S| \sim \; \{P_1, P_2\}_{K_I}$

S_I 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_I is the part of the P_1, P_2; thus, on applying (R5), we get

(8) $S| \equiv D| \equiv S_I$

Now as per (A5), (8), and (R3), we get

(9) $S| \equiv S_I$

From (S3) and (3), we get

(10) $S| \equiv \; D| \sim \; S_I$

From (A3) and (10), we get

(11) $S| \equiv D\| \sim \; S_I$

As per (R4) and (11), we get

(12) $S| \equiv \#(\,S_I)$

S_I is a part of (SK); thus, as per (R6), we get

(13) $S| \equiv \#(SK)$

From (13), (8), and (R7), we get

(14) $S| \equiv \; D \longrightarrow^S K\; S$*(Goal G1)*

Due to the symmetry of the protocol,

(15) $S| \equiv D| \equiv S \longrightarrow^S K\; D$*(Goal G2)*

From (MSG2), we infer that

(16) $S| \equiv \{P_3, P_4\}_{X_S}$

(17) $S \Leftarrow \{P_3, P_4\}_{X_S}$

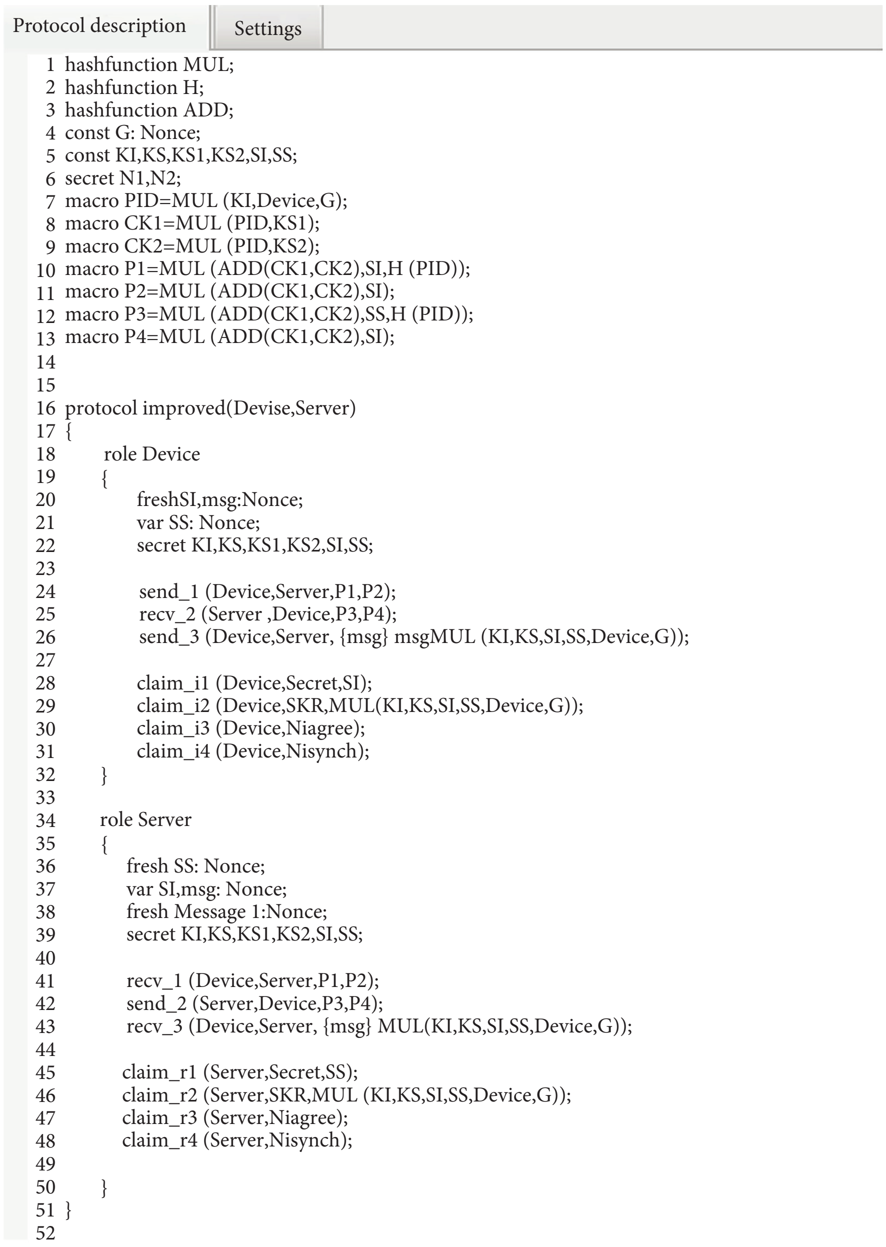
From (17), (A2), (R1), we get

(18) $D| \equiv S| \sim \{P_3, P_4\}_{X_S}$

S_s is a part of P_3, P_4; thus, as per (A4) and (R6), we get

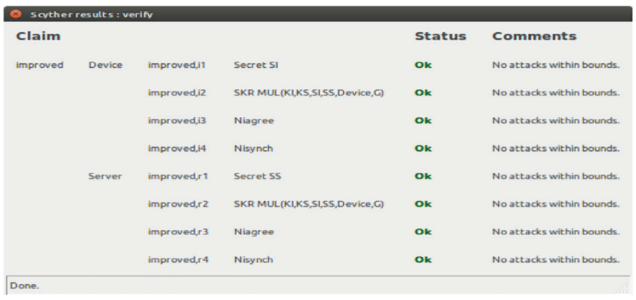
(19) $S| \equiv \#(P_3, P_4)$

From (17) and (19), we get


Protocol description | Settings

```
hashfunction MUL;
hashfunction H;
hashfunction ADD;
const G: Nonce;
const KI,KS,KS1,KS2,SI,SS;
secret N1,N2;
macro PID=MUL (KI,Device,G);
macro CK1=MUL (PID,KS1);
macro CK2=MUL (PID,KS2);
macro P1=MUL (ADD(CK1,CK2),SI,H (PID));
macro P2=MUL (ADD(CK1,CK2),SI);
macro P3=MUL (ADD(CK1,CK2),SS,H (PID));
macro P4=MUL (ADD(CK1,CK2),SI);


protocol improved(Devise,Server)
{
     role Device
     {
          freshSI,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} msgMUL (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.


Scyther results : verify

Claim				Status	Comments
improved	Device	improved,i1	Secret SI	Ok	No attacks within bounds.
		improved,i2	SKR MUL(KI,KS,SI,SS,Device,G)	Ok	No attacks within bounds.
		improved,i3	Niagree	Ok	No attacks within bounds.
		improved,i4	Nisynch	Ok	No attacks within bounds.
	Server	improved,r1	Secret SS	Ok	No attacks within bounds.
		improved,r2	SKR MUL(KI,KS,SI,SS,Device,G)	Ok	No attacks within bounds.
		improved,r3	Niagree	Ok	No attacks within bounds.
		improved,r4	Nisynch	Ok	No attacks within bounds.

Done.

Figure 11: Scyther verification results for the proposed scheme.

Table 2: BAN symbols.

Notation	Description
$D\| \equiv MSG$	D believes M
$D \Leftarrow MSG$	D receives the message M
$D\| \sim MSG$	D sent the message M in past
$D\|\| \sim MSG$	D sent the message to M currently
$D\| \longrightarrow F$	D has jurisdiction over V
$\#(MSG)$	M is fresh
$\longrightarrow^{P} r\ D$	P_{r} is the public parameter calculated using the private key of D
$D \longrightarrow^{S} K\ S$	SK is the key between D and S
$\{X\}_{SK}$	SK is the key used to encrypt X
$(F1/F2)$	If F1 is true, then F2 is true

Table 3: BAN postulates.

Rule no.	Name	Representation
R1	Message-meaning rule	$(D\| \equiv\ \longrightarrow^{P_r} S,\ D \Leftarrow \{MSG\}XS/D\| \equiv S\| \sim MSG)$
R2	Nonce verification rule	$(D\| \equiv \#(MSG), D \equiv S\| \sim MSG/D\| \equiv S\| \equiv MSG)$
R3	Jurisdiction rule	$(D\| \longrightarrow MSG, D\| \equiv S\| \equiv MSG/D\| \equiv MSG)$
R4	Seeing rule	$(D \Leftarrow MSG1, D \Leftarrow MSG2/D \Leftarrow (MSG1, MSG2))$
R5	Belief rule	$(D\| \equiv MSG1, D\| \equiv MSG2/D\| \equiv (MSG1, MSG2))$
R6	Freshness rule	$(D\| \equiv \#(MSG1)/D\| \equiv \#(MSG1, MSG2))$
R7	Session key rule	$(D\| \equiv \#(SK), D\| \equiv S\| \equiv X/D\| \equiv D \longrightarrow^{S} K\ S)$ *X is part of SK*

Table 4: Synthesis rules.

Rule no.	Synthesis rule
S1	$D \Leftarrow MSG1\ D \Leftarrow (MSG1, MSG2)$
S2	$D\| \equiv S\| \sim MSG1 D\| \equiv S\| \sim (MSG1, MSG2)$
S3	$D\| \equiv S\| \sim (MSG1, MSG2)\ D\| \equiv S\| \sim MSG1$
S4	$D\| \equiv S\| \sim MSG1 D\| \equiv \#(MSG1)$

(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) $D| \equiv S| \equiv P_3, P_4$

S_S is the part of the formulae P_3, P_4; thus, on applying (R5), we get

(23) $D| \equiv S| \equiv S_S$

Now, as per (A6) and (23) and (R3), we get

(24) $D| \equiv S_S$

From (SR3) and (18), we get

(25) $D| \equiv S| \sim S_S$

From (A4) and (25), we get

(26) $D| \equiv S|| \sim S_S$

As per (R4) and (26), we get

(27) $D| \equiv \#(S_S)$

From (R6), we get

(28) $D| \equiv \#(SK)$

From (28) and (22) and on applying (R7), we get

(29) $D| \equiv D \longrightarrow^{S} K\ S$*(Goal G3)*

Due to the symmetry of the protocol,

(30) $S| \equiv D| \equiv D \longrightarrow^{S} K\ S$*(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

Table 5: Security comparison.

Scheme	AK1	AK2	AK3	AK4	AK5	eCK
Kalra and Sood [14]	✗	✗	✗	✗	✗	✗
Chang et al. [15]	✗	✗	✓	✓	✓	✗
Kumari et al. [16]	✗	✗	✓	✓	✗	✗
Wang et al. [17]	✗	✓	✓	✓	✗	✗
Rostampour et al. [11]	✓	✓	✓	✓	✓	✗
Proposed scheme	✓	✓	✓	✓	✓	✓

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.

Scheme	Computation overhead			Communication overhead
	Device	Server	Total	
Kalra and sood [14]	$3\ T_{ECM} + 4\ T_{HASH}$	$4T_{ECM} + 5\ T_{HASH}$	$8T_{ECM} + 9\ T_{HASH}$	1,760
Chang et al. [15]	$4T_{ECM} + 4\ T_{HASH}$	$4T_{ECM} + 4\ T_{HASH}$	$8T_{ECM} + 8\ T_{HASH}$	1,632
Kumari et al. [16]	$4T_{ECM} + 3\ T_{HASH}$	$4T_{ECM} + 4\ T_{HASH}$	$8T_{ECM} + 7\ T_{HASH}$	1,760
Wang et al. [17]	$4T_{ECM} + 5\ T_{HASH}$	$4T_{ECM} + 3\ T_{HASH}$	$8T_{ECM} + 8\ T_{HASH}$	1,632
Rostampour et al. [11]	$7T_{ECM} + T_{EADD}$	$6T_{ECM} + T_{EADD}$	$13\ T_{ECM} + 2\ T_{EADD}$	1,920
Proposed scheme	$T_{ECM} + T_{EADD} + 2\ T_{HASH} + T_{SENC}/T_{SDEC} + T_{inv}$	$4\ T_{ECM} + T_{EADD} + 2\ T_{HASH} + T_{SENC}/T_{SDEC} + T_{inv}$	$8\ T_{ECM} + 2T_{EADD} + 4\ T_{HASH} + 2\ T_{SENC}/T_{SDEC} + 2\ T_{inv}$	1,408

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

Operation	Seconds
T_{ECM}	2.82
T_{SENC}/T_{SDEC}	0.000029
T_{HASH}	0.0091
T_{inv}	0.14
T_{EADD}	0.16

Rostampour et al. scheme with 7 T_{ECM} 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 (T_{ECM}) 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 elliptical 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, 03 messages are exchanged that include (P_1, P_2), (P_3, P_4), and $E_{SK}[H(\mathrm{SK})]$. The 03 messages requires $[(320 + 320),(320 + 320)+128] = 1{,}408$ 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 $E = V * I * T$, 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

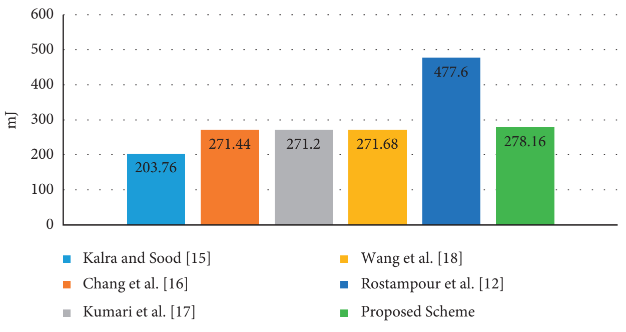


Figure 12: Energy overhead comparison.

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 to 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 communication overhead of 1,408 bits, the proposed scheme supports all the security requirements and is also safe under 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 Yoa 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 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

[1] S. Saxena, S. Vyas, B. Kumar, and S. Gupta, "Survey on online electronic payments security," in *Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI)*, Dubai, UAE, February 2019.

[2] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[3] J. Bhola, S. Soni, and G. K. Cheema, "Recent trends for security applications in wireless sensor networks-a technical review," in *Proceedings of the 6th IEEE International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 707–712, New Delhi, India, March 2019.

[4] "Global IoT Market to be Worth USD 1,463.19 Billion by 2027 at 24.9% CAGR; Demand for Real-time Insights to Spur Growth, says Fortune Business Insights," https://www.globenewswire.com/newsrelease/2021/04/08/2206579/0/en/Global-IoT-Market-to-be-Worth-USD-1-463-19-Billion-by-2027-at-24-9-CAGR-Demand-for-Real-time-Insights-to-Spur-Growth-says-Fortune-Business-Insights.html.

[5] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on IoT security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[6] R. Porkodi and V. Bhuvaneswari, "The Internet of Things (IoT) applications and communication enabling technology standards: an overview," in *Proceedings of the 2014 International Conference on Intelligent Computing Applications*, pp. 324–329, Coimbatore, Tamilnadu, March 2014.

[7] M. R. Abdmeziem, D. Tandjaoui, and I. Romdhani, "Architecting the internet of things: state of the art," *Robots and Sensor Clouds*, vol. 3, pp. 55–75, 2015.

[8] U. Iqbal and A. H. Mir, "Secure and scalable access control protocol for IoT environment," *Internet of Things*, vol. 12, Article ID 100291, 2020.

[9] G. Rastogi and R. Sushil, "Cloud Computing Implementation: Key Issues and Solutions," in *Proceedings of the 2nd IEEEInternational Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 320–324, New Delhi, India, March 2015.

[10] G. P. Gupta, "Security issues and its countermeasures in examining the cloud-assisted IoT," *Advances in Wireless Technologies and Telecommunication*, vol. 5, pp. 91–115, 2018.

[11] S. Rostampour, M. Safkhani, Y. Bendavid, and N. Bagheri, "ECCbAP: a secure ECC-based authentication protocol for IoT edge devices," *Pervasive and Mobile Computing*, vol. 67, 2020.

[12] Y.-P. Liao and C.-M. Hsiao, "A secure ECC-BASED RFID authentication SCHEME integrated with Id-verifier transfer protocol," *Ad Hoc Networks*, vol. 18, pp. 133–146, 2014.

[13] P. Roel and J. Hermans, "Attack on liao and Hsiao's secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol," *IACR Cryptology*, vol. 399, 2013.

[14] S. Kalra and S. K. Sood, "Secure authentication scheme for IoT and cloud servers," *Pervasive and Mobile Computing*, vol. 24, pp. 210–223, 2015.

[15] C.-C. Chang, H.-L. Wu, and C.-Y. Sun, "Notes on "Secure authentication scheme for IoT and cloud servers"," *Pervasive and Mobile Computing*, vol. 38, pp. 275–278, 2017.

[16] S. Kumari, M. Karuppiah, A. K. Das, X. Li, F. Wu, and N. Kumar, "A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6428–6453, 2017.

[17] K.-H. Wang, C.-M. Chen, W. Fang, and T.-Y. Wu, "A secure authentication scheme for Internet of Things," *Pervasive and Mobile Computing*, vol. 42, pp. 15–26, 2017.

[18] N. Saqib and U. Iqbal, "Security in wireless sensor networks using ECC," *IEEE International Conference on Advances in Computer Applications (ICACA)*, pp. 270–274, 2016.

[19] Z. Zhao, C. Hsu, L. Harn, Q. Yang, and L. Ke, "Lightweight privacy-preserving data sharing scheme for Internet of medical Things," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 8402138, 13 pages, 2021.

[20] D. Dolev and A. Yao, "On the security of public-key protocols," in *Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, pp. 350–357, Nashville, TN, USA, October 1981.

[21] D. Dolev and A. Yao, "On the security of public-key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[22] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," *Lecture Notes in Computer Science*, vol. 2045, pp. 453–474, 2001.

[23] W. Wen, L. Wang, and J. Pan, "Unified security model of authenticated key exchange with specific adversarial capabilities," *IET Information Security*, vol. 10, no. 1, pp. 8–17, 2016.

[24] C. J. Cremers, “The scyther tool: verification, falsification, and analysis of security protocols,” *Computer Aided Verification*, vol. 5123, pp. 414–418, 2008.
[25] M. Burrows, M. Abadi, and R. M. Needham, “A logic of authentication,” *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 426, pp. 233–271, 1989.
[26] Z. Clement Chan, C. Chuah, and J. Alawatugoda, “Review on leakage resilient key exchange security model,” *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 11, no. 1, 2019.
[27] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
[28] V. S. Miller, “Use of elliptic curves in cryptography,” *Lecture Notes in Computer Science Advances in Cryptology*, vol. 1985, pp. 417–426, 1986.
[29] D. Hankerson and A. Menezes, “Elliptic curve cryptography,” *Encyclopedia of Cryptography and Security*, vol. 397, 2011.
[30] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, “Comparing elliptic curve cryptography and RSA on 8-bit CPUs,” *Lecture Notes in Computer Science*, vol. 3156, pp. 119–132, 2004.
[31] S. H. Islam and G. P. Biswas, “A pairing-free identity-based two-party authenticated key agreement protocol for secure and efficient communication,” *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 1, pp. 63–73, 2017.
[32] D. J. Malan, M. Welsh, and M. D. Smith, “Implementing public-key infrastructure for sensor networks,” *ACM Transactions on Sensor Networks*, vol. 4, no. 4, pp. 1–23, 2008.
[33] L. Buttyan, S. Staamann, and U. Wilhelm, “A simple logic for authentication protocol design,” in *Proceedings of the 11th IEEE Computer Security Foundations Workshop*, pp. 153–162, Rockport, MA, USA, June 1998.
[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 Scnces*, 2020.
[35] Mote Works, *Getting Started Guide*, Memsic, Inc., 2013.