The Journal of Supercomputing (2023) 79:12191–12218
https://doi.org/10.1007/s11227-023-05135-0

ECC-reliant secure authentication protocol for cloud server and smart devices in IoT

K. Selvi[1] · K. Muthumanickam[2] · P. Vijayalakshmi[3] · P. C. Senthil Mahesh[4]

Accepted: 22 February 2023 / Published online: 7 March 2023

Abstract
The Internet of Things (IoT) designates a network that helps to relate a diversity of heterogeneous devices, various technologies, and other items to the Internet for flexible access and data exchange. Recent smart real-time application design requires the integration of ‘things’ in IoT with cloud infrastructure to offer valuable services to end-users. However, such a combination could raise various security concerns which become the most critical problem nowadays. For protected communication between smart devices interconnected through IoT and cloud servers, authentication becomes one of the crucial security requirements. There exist many strategies specifically for authentic key exchange between smart devices in the IoT environment and the cloud server. But according to the improved Canetti–Krawczyk (xck) rival model which is considered a more appropriate model for evaluating authentication-based security systems, none of the systems is safe and is vulnerable to a variety of assaults. Thus, we explored xck rival model to prove the limitations of the existing approach and presented an Elliptic Curve Cryptographic reliant strategy to overcome such limitations. The soundness and correctness of our approach were evaluated using scyther verification method. The evaluation results confirm that our method is robust and secure under xck model and incurs minimal overhead.

Keywords Authentication · ECC · Cloud · IoT · Security

1 Introduction

IoT is a grid of interconnected computer-reliant ‘things’ such as smart gadgets, digital machines, sensors, heart monitors, or people, who are assigned an identifier to be exclusive among all entities and have the tendency to transport data across a network automatically. Entities in IoT environment can be used for unlawful boundary

✉ K. Selvi
selvikphd@gmail.com

Extended author information available on the last page of the article

contact, recognition of dangerous radiation and chemical leakage, and revealing gas leakage in industrialized environments with regard of emergency and secure applications [1]. The entities can be used for soil quality monitoring, greenhouse climate control, and smart irrigation in smart agriculture. Surveillance, continuous monitoring of real-time appliances, water saving, and energy are just a few of the smart home uses. IoT can be used to generate applications like wireless body sensor networks, automatic monitoring of smart devices in the medical industry, and geriatric aid [2].

Important security concepts and services that should be provided and ensured while accessing real-time applications through an open network are:

(i) *Trust*—The distributed and dynamic nature of IoT systems necessitates trust. The importance of ensuring the trustworthiness of interacting devices cannot be overstated. Restriction of power usage is also an important aspect of developing a trust management system.
(ii) *Confidentiality* — Ensuring the confidentiality of messages exchanged between two entities in an open network must be protected. Data monitoring, gathering, sharing, and security are all areas where privacy is a problem.
(iii) *Reliability* – This is a must-have feature that ensures data and service accessibility.
(iv) *Session Key Usability*—In any authentication strategy, the usage of a session key is critical. It is primarily used to protect communications against third-party assaults.

Smart real-time applications built on the Internet of Things must be secure [3]. Traditional security methods can be used to provide secure communication with smart devices to the cloud server. However, because IoT devices have limited resources, simple message exchanges between smart devices residing in the sensitivity layer and server are becoming a hot topic of research [2]. Additional security services offered between the perception layer and server are built on the foundation of authentication, integrity, and session key exchange [4]. Many approaches for protecting communication exchange between smart devices residing in IoT environments and cloud servers have been proposed in the past. A mutual authentication technique with a unique identifier verification protocol was suggested by Liao and Hsiao [5]. However, Roel and Hermans [6] point out, the method is vulnerable to server impersonation assaults.

Kalra and Sood [7] presented an Elliptic Curve Cryptography (ECC) reliant mutual authentication system to resist a wide range of attacks. The approach, however, contains design flaws with regard to reciprocal verification, insider assaults, and the discovery of their traces. Chang et al. [8] discovered flaws in the work proposed by Kalra and Sood, particularly with regard to shared authentication along with the mistress of sensitive information like session key. Thus, Chang et al. proposed a better strategy that circumvented Kalra and Sood's flaws. Similarly, about concerning device anonymity, insider attacks, session/secret key agreement, and mutual authentication. Kumari et al. [9] discovered the flaws in Kalra and Sood's scheme concerning impersonation-reliant attacks and proposed a better scheme. To circumvent these restrictions, Kumari et al. proposed an ECC-reliant approach. Wang et al. [10] discovered the flaws concerning impersonation-based attacks in Chang et al. scheme and proposed a better solution.

Rostampour et al. [4] suggested an ECC-reliant technique for IoT edge device authentication with the cloud server. Rostampour et al. examined various existing solutions [8–10] in detail and claimed that they were vulnerable to traceability attacks. In [13], Ummer et al. explored the flaws present in Rostampour's scheme and presented a better solution. When linking an implant device to the cloud, the most important factor to consider is security. Shared authentication between the implant devices and the cloud server is also required. ECC scheme is the popular and the strongest public-key cryptography option when storage space, memory area, and power are restricted and more security by a short key is desired [11, 12]. Furthermore, using the xck adversary model (ad-model), an enhanced approach for proving authentication and a secret key contract between smart devices in IoT and the server, namely sd-to-cs has been developed.

1.1 Contributions

Three previous authentication schemes, Kumari, Rostampour, and Ummer, addressed security concerns between fog computing devices and the cloud server by leveraging ECC to defend against various attacks. Although session key exposing is one of the most important security issues that aids in achieving mutual validation and device secrecy, we show in this paper that Kumari, Rostampour, and Ummer's strategy fails in this regard. We have proposed an improved ECC-reliant authentication mechanism that has been designed specifically for IoT and cloud servers to address the aforementioned weakness of the Kumari, Rostampour, and Ummer schemes. The suggested approach was modeled after the Scyther compromise. The results of the XCK rival prototype demonstrate the security and safety of the method.

1.2 Paper organization

The rest of the paper is formatted as follows. Section 2 contains the preliminaries. Three existing approaches, namely Kumari et al., Rostampour et al., and Umar Iqbal et al. approaches are reviewed in Sect. 3 along with its recognized security verification using the xck rival model. The proposed ECC-reliant system for authenticating between smart devices residing in the IoT environment and the cloud server is detailed in Sect. 4. The outcome of a formal security investigation utilizing the scyther against the xck ad-model is provided in Sect. 5. Section 6 briefs about security analysis of the suggested scheme by exploring BAN logic. The proposed scheme is compared to other appropriate authentication schemes given in Sect. 7. Finally, in Sect. 8, we conclude our work.

2 Preliminaries

Table 1 lists the various notations used in the next sections.

2.1 Rival model

An ad-model expresses the attacker's prospective capabilities. Dolev and Yoa [14, 15], namely DY, Canetti–Krawczyk referred to as CK, and its improved version (xck) models [16] are examples of adversarial models. The communication channel is completely unsafe in all of the models; nevertheless, they contrast in their challenger query abilities. The communicating parties are deemed honest in the DY threat model and can have several sessions between them. The communication medium is entirely unsafe and completely controllable by the opponent, who can able to perform operations like the record, delete, replay, reroute, reshuffle, and manage the message list. In the middle, the adversary can behave as a legitimate user and conduct different kinds of attacks. The xck and CK schemes for key sharing and authentication procedures are the most extensively utilized.

In this adversary scenario, an attacker can breach the pseudo-random number generator (PRNG) and gain access to the session's secret randomness. An opponent is also expected to be able to conciliate the session and gain access to it. Long-term keys can also be obtained by an attacker [18]. The xck paradigm differs from the CK model in that the adversary can obtain ephemeral secrets, resulting in an ephemeral-secret-key-leakage attack.

Table 1 Notations used

Notation	Remark
$E_P(a, b)$	Elliptical curve
CS	Cloud server
SD_i	i^th smart device in IoT
SID_i	Identity of SD_i
PD_i	Passcode of SD_i
ID_{CS}	Identity of CS
N_i	Random number
X_{CS}	Private secret key of CS
K_{CS}	Private secret key of SD_i
ES_i	Short-lived secret of SD_i
E_t	Expiration time
T_i	Timestamp
$G(x,y)$	Generator point of $E_P(a, b)$
SK	Session key shared between CS and SD_i
h(.)	Hash-based function
$\oplus$	Exclusive-OR operation
$P+Q$	Point addition over elliptical curve
X, G	Scalar multiplication over elliptical curve

2.2 Elliptical curve cryptography

The application of elliptic-curve in cryptographic systems was first advocated by Koblitz [19] and Miller [20]. ECC is one of the most general and widely utilized encryption algorithms, yet it is also one of the least well-known. It is the famous next-generation and more secure public-key cryptography method. When compared to a first-generation public key cryptographic system like RSA, it provides far more security. Elliptical curve cryptography, with a key size of 160 bits, provides a similar level of security as RSA, which makes it perfectly suitable for power-constrained devices [21]. Because of its lower key length and capability to preserve security, ECC has gained prominence in recent years. This tendency is projected to remain as the requirement for secure devices grows in response to the growing key length, affecting the vital resources of smart devices. This is the reason why understanding the encryption process in ECC in the context of low-power devices is critical [21].

ECC is one of the popular and secure public cryptography schemes that relied on the algebraic organization of elliptical curves well-defined over a finite field. When compared to non-ECC encryption, ECC delivers equivalent safety for a lower key. The elliptic curve cryptosystem was first established as the cornerstone of the public-key cryptographic system, and suggestion has demonstrated that it is a critical component of the system. Cryptography in the modern era ECC has relied on the fundamentals of elliptic curves. The ECC's security relied on the effort of handling the elliptical curve's complex algorithm. Over a finite field, K an elliptical curve, i.e., E over K is defined as given in Eq. 1 [21].

$$E_K(x,\ y) \leftarrow y^2 :\ x^3 + ax + b.\ \text{where } x,\ y\ \in K \tag{1}$$

A few important operations to be applied on elliptic curves are itemized below.

(i) *Addition of two points*—The addition of two different points $M_1 = (x_1, y_1)$ and $M_2 = (x_2, y_2)$ of an elliptic curve can be calculated using the formulas as follows.

$$M_1 + M_2 = M_3 = \left(x_3,\ y_3\right) \tag{2}$$

In equation 2, x_3 is computed as $\lambda^2 - x_1 - x_2$, the value of y_3 is computed as $\lambda\ (x_1 - x_3) - y_1$, and

$$\lambda = \left\{ \begin{array}{l} \frac{(y_2 - y_1)}{(x_2 - x_1)},\ \mathit{if}\ M_{1 \neq} M_2 \\ \frac{(3x_1^2 + a)}{(2y_1)},\ \mathit{if}\ M_{1 =} M_2 \end{array} \right\}$$

(b) *Scalar Multiplication*—An elliptic curve's scalar multiplication represents an operation which adds a unique point 'P' to the curve k number of times.

$Q = kM = M + M + \cdots + M$, k number of times.

Where M is an elliptical curve point, and k denotes a large positive integer.

(iii) *Discrete Logarithm on Elliptic Curve*—Let an elliptic curve, E, and its points, A and B, with $B = kA = (A + A + \ldots + A)$ – k times for some k. The discrete logarithm issue for elliptic curves is the task of locating such a k. There is no efficient algorithm or good general attacks for computing discrete logarithm problems for elliptic curves. The cryptography based on elliptic curves is grounded on these facts.

2.3 Scyther simulation

Simulation of a Scyther Cremers [17] designed Scyther, a tool for assessing and certifying security protocols. It's software that can perform security risk assessments and attack simulations. Scyther verifies security protocols using an endless number of sessions. Scyther also provides for the verification of multiprotocol assaults. In Scyther, a security protocol is represented using the SPDL (Scyther protocol description language) to verify and validate it. When attacks are found, attack graphs are generated in scyther tool but not in AVISPA tool. This is one of the noted features of scyther tool.

When an attack is detected, a trace pattern is produced as either an XML representation or a visual graph. The sender and receiver principals' communication pattern is defined by roles in the SPDL representation of security protocol. To express the various security needs, the phrase claim is utilized. Dolev and Yoa are the default opponent models in the Scyther version. In comparison to the conventional Scyther version, the ScytherCompromise provides more support for diverse opponent models. Scyther Compromise tool version 0.9.2 was utilized in this paper to generate different claims and attacks.

3 Review of appropriate existing schemes

3.1 Preview of Kumari et al. method

Kumari et al. [9] proposed an ECC cryptosystem-reliant authentication approach for smart devices in the Internet of Things. The protocol is divided into three phases, which are detailed below.

3.1.1 Registration phase

The registering stage takes place through a secure connection between the SD and CS. The steps are as follows:

(i) SD_i computes $L_i = h(SID_i \,||\, PD_i)$ and transmits it to CS over a secure communication medium for registration purposes.

(ii) After the registration is successfully processed, CS generates a random number N_i and computes the pseudo-identity for SID_i as $h(N_i \| ID_{CS} \| L_i)$. Afterward, it computes a few other vital secret information as follows: $Ci = h(Ni||XCS||Et||Li), C_i = C_i \times G, T_i = N_i \oplus h(X_C S||SD_i), M_i = h(N_i \oplus h(X_C S||SD_i) \oplus L_i \oplus C_i), M_i = M_i \times G$. In addition, SD_i calculates $t_i = T_i \oplus X_{CS}$,$m_i = M_i \oplus X_{CS}$, and $e_t = (E_t \oplus X_{CS})$ and keeps it in the database. Then, 'CS' transmits SID_i, $C_i^{|}$ to SD using a secure channel. After$\{SID_i, C_k^{|})$ is received by the SD, it keeps all data safely in a protected memory area. After the time expiration of cookies, C_i is re-computed as$C_i = h(N_i||X_{CS}||E_t||SID_i)$
.

3.1.2 Login phase and validation phase

(i) Initially, SD_i selects a random number N_1 and then determines (M_1, M_2) where $M_1 = (N_1.G)$ and $M_2 = h\left(N_1.C_i^{|}\right)$ and asks for login demand to SD.

(ii) After successfully receiving login credentials, SD computes $N_s = T_i \oplus h(X_{CS}||SID_i)$ *and* $C_i = h\left(N_s||X_{CS}||E_t||SID_i\right)$. SD re-computes M_2^* value as $h(M_1. C_i)$ and cross-check with the received M_2. If both are the same, the SD_i proceeds further otherwise the request was discarded.

(iii) Now, the SD selects a random number N_2, and calculates another point over ECC as $M_3 = (N_2. G)$ and $M_4 = (N_2. M_i)$ and transmits (M_3, M_4) along with T_i to SD_i.

(iv) Upon receiving M_3, M_4, and T_i, SD assess$M_i = T_i L_i C_i)$,$M_4^* = M_3.M_i$. Then, SD_i validates the received M_4 value against the computed M_4 value. If matches, login is successful, otherwise asks to re-login again.

(v) Then, SID_i generates the new session key i.e., $SY = (N_1. M_3) = (N_1.N_2.G)$, and $O_i = h(N_1. C_i) \| SY$ and then transmits O_i to the cloud server.

(vi) After receiving O_i, CS re-computes the session key to validate against the received SY. If matches, authentication is successful.

3.2 Security attacks and flaws in Kumari scheme

3.2.1 Security attacks

(i) Exposing session-key information–If the random number selected for generating a session key is exposed to the adversary in some means, SK's secrecy is jeopardized. In Kumari et al. approach's the session key is computed using only the session's ephemeral secrets, which is $(N_1.N_2.G)$ where N_1 and N_2 are random numbers chosen by SID_i and CS, respectively.

The random numbers created for a particular session are one-of-a-kind, and they must be deleted once the procedure has been completed. Assume that attacker *A* has access to the random numbers N_1 and N_2. As G is a public key, *A* will be able to calculate the session key with ease, as it does not rely depend on other confidences. As a result, the authentication mechanism is vulnerable to this attack.

(b) Denial of Service attack (DoSA) – *A* can perform DoSA by performing the following operations.

a. *A* intercepts login operation is involved between SD_i and CS.
b. At any point, after the original message has been transmitted, *A* can explore the captured message to CS and launch a DoSA assault.
c. Due to accurate {M_1, M_2, and SID_i}, CS will validate *A*. As a result, *A* can pretend to be a legitimate user. Then, CS replies to *A* with the message {M_3, M_4, T_i}.

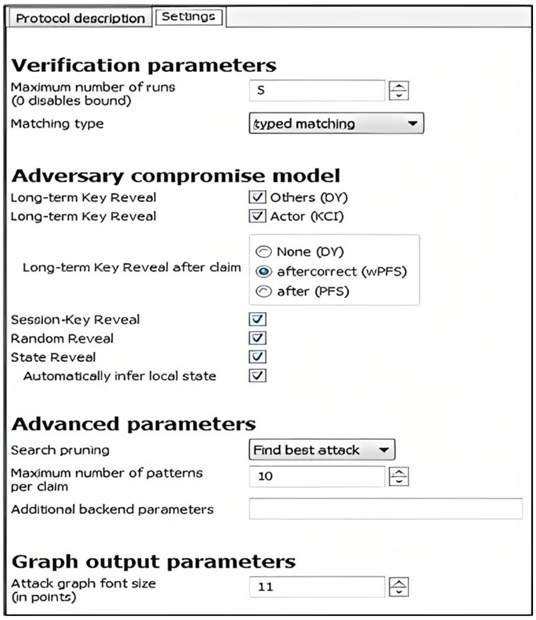
3.2.2 Ineffective authentication and login phases

(i) Let's say *A* obtains the device, but the device does not know this because it never confirms its user. ED_i will continue to carry through the procedures even if the user enters the password, identity, or both, incorrectly. It immediately generates a random number to start the procedure, then it does $M_1 = (N_1.G) and M_2 = h(N1.C_i^{|})$. Now that M_1 has been stored, *A* sends the request message {M_1, M_2, SID_i} to the server, which appears to be an appropriate request for a login to the control server. Because it allows any user to log in as a legal user, the strategy becomes ineffective.
(ii) *A* can perform a replay attack to compromise the authentication phase. After CS accepts the login request, CS finds N_s matching to SIDi as$N_s = T_i \oplus h\left(X_{CS}||SID_i\right)$). Afterward, CS computes C_i and M_2^* and verifies$M_2? = M_2^*$. It holds $M_2^* = h\left(M_1.C_i\right) = M_2$. This operation helps to authenticate A legally. This is one of the design flaws of the Kumari scheme.
(iii) An authentication protocol must include the password-changing phase. This is necessary if the user forgets or loses their password, or if the old password is vulnerable to attack. Once the password is revealed, there should be a way to prevent it from being used illegally. By regularly changing passwords, you lower the chance that *A* will gain access. However, Kumari et al., do not provide a password-changing method that could stop an attacker from impersonating a legitimate user.

3.3 Formal security investigation of Rostampour's scheme

Rostampour's method consists of two different phases, namely (i) Registration phase (ii) Login phase followed by the authentication phase. The SPDL using property-driven model checking is utilized to reveal the Rostampour scheme as given in Fig. 1.

Fig. 1 Scyther setting for DY adversary models



Protocol description | Settings

Verification parameters

Maximum number of runs (0 disables bound): 5

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:

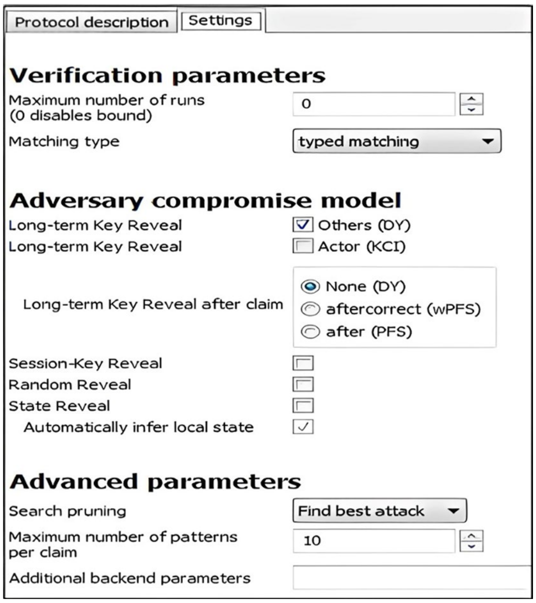
Graph output parameters

Attack graph font size (in points): 11

Roles I and R are included in the SPDL modeling where the communication of the SD_i is modeled by role I, while the S is modeled by role R. Using send1() method, SD_i sends (M_1, M_2, SID_i) to the server. Using the recv1() function, the server obtains (M_1, M_2, SID_i). S uses send2() to send (M_3, M_4) to the DD_i. Using the recv2() function, the SD_i obtains the (M_3, M_4). Finally, SD_i uses send3() to send V_i to the server. Utilizing the recv3() method, the server obtains the V_i. The N_1, N_2 must remain a secret throughout the conversation, according to the claim_i1 that belongs to role I besides claim_r1 that resides roles I and R. The SPDL properties-driven model has been primarily conducted under DY model setup as indicated in Fig. 1.

The authentication outcome of the Rostampour et al. system under the DY setup in Scyther Compromise version suggests that the method is secure and involves no attacks. Next, the method was performed under the xck rival scenario illustrated in Fig. 2. The Scyther authentication findings under xck are displayed in Fig. 3. The outcomes show that the system is not secure.

Figure 4 displays the assault trace of the Rostampour technique beneath the xck rival model failed to resist against session key disclose, according to the attack trail; as a result, the adversary can leak the secret key. In addition, the Rostampour scheme's design flaw in computing the session key is the main cause of this. The session key used by Rostampour et al. is calculated using the formula (N1.N2.G). Only the temporary session secrets N_1 and N_2 are a source of the session key. For the revelation of short-standing secrets to expose the session key, the session key must also be reliant on long-standing secrets.



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: [✓]

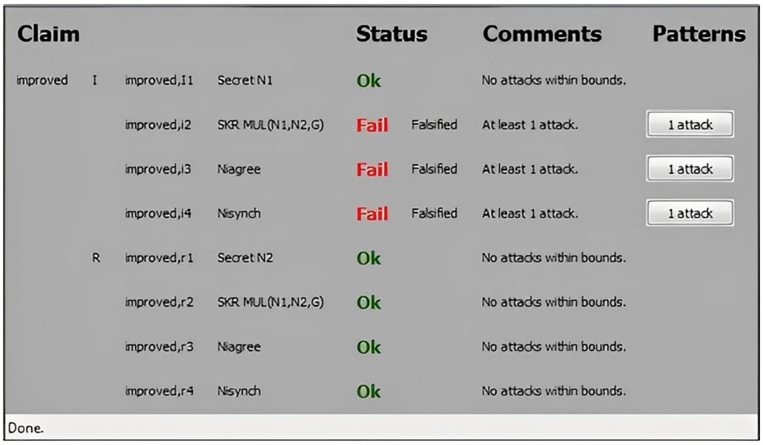
Advanced parameters

Search pruning: Find best attack

Maximum number of patterns per claim: 10

Additional backend parameters:

Fig. 2 Scyther settings for the xck adversary model



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.

Fig. 3 Scyther verification results of the Rostampour et al. scheme under the xck model

3.4 Formal security investigation of Ummer Iqbal et al. scheme

Ummer Iqbal et al. presented a design of an ECC-reliant technique to offer authentication between smart devices in IoT and the cloud server under the xck challenger model is put forth to get over the drawback of Rostampour's method. However, using the Scyther Compromise simulation version, we automated the security validation and verification scheme of the Ummer Iqbal et al. under the xck challenger model. The validation shows that, according to the xck adversary model, the technique is

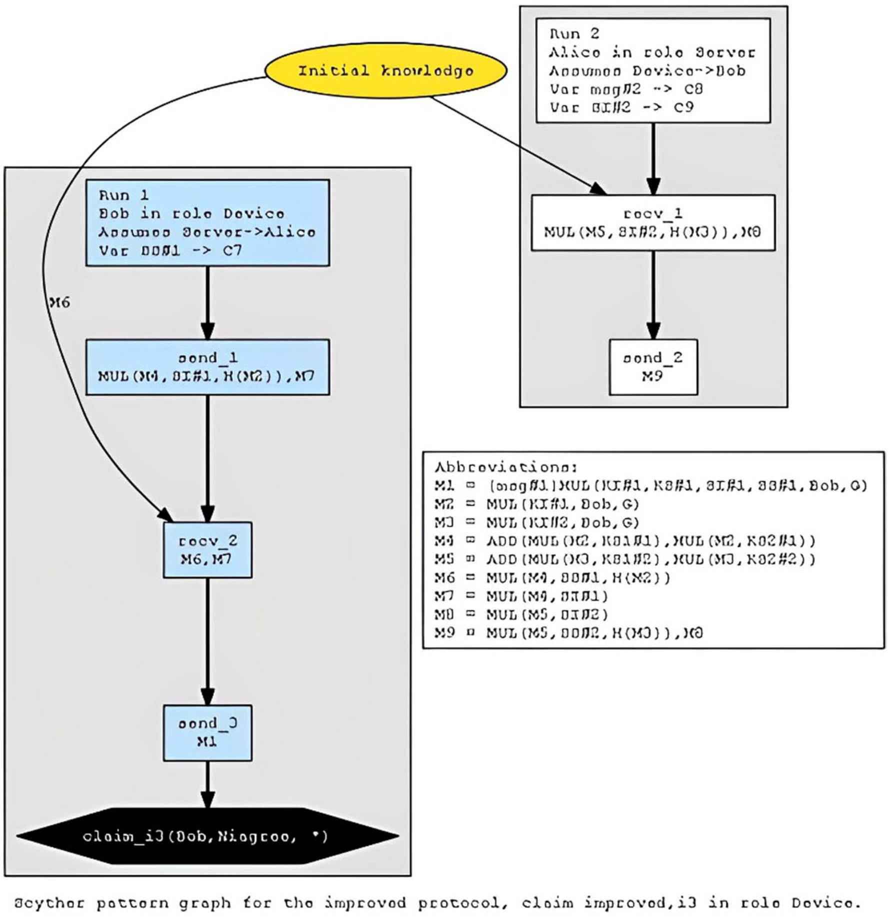
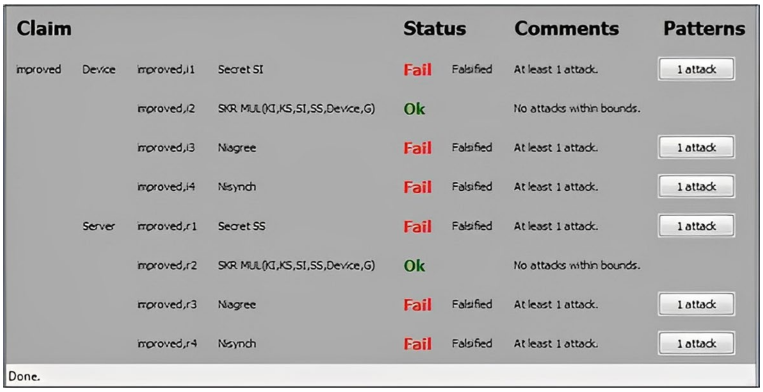


Fig. 4 Attack graph for Rostampour et al. scheme under the xck model



Claim				Status		Comments	Patterns
improved	Device	improved,i1	Secret SI	Fail	Falsified	At least 1 attack.	1 attack
		improved,i2	SKR MUL(KI,KS,SI,SS,Device,G)	Ok		No attacks within bounds.	
		improved,i3	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		improved,i4	Nisynch	Fail	Falsified	At least 1 attack.	1 attack
	Server	improved,r1	Secret SS	Fail	Falsified	At least 1 attack.	1 attack
		improved,r2	SKR MUL(KI,KS,SI,SS,Device,G)	Ok		No attacks within bounds.	
		improved,r3	Niagree	Fail	Falsified	At least 1 attack.	1 attack
		improved,r4	Nisynch	Fail	Falsified	At least 1 attack.	1 attack

Done.

Fig. 5 Scyther authentication results of the Ummer Iqbal et al. scheme under the xck model

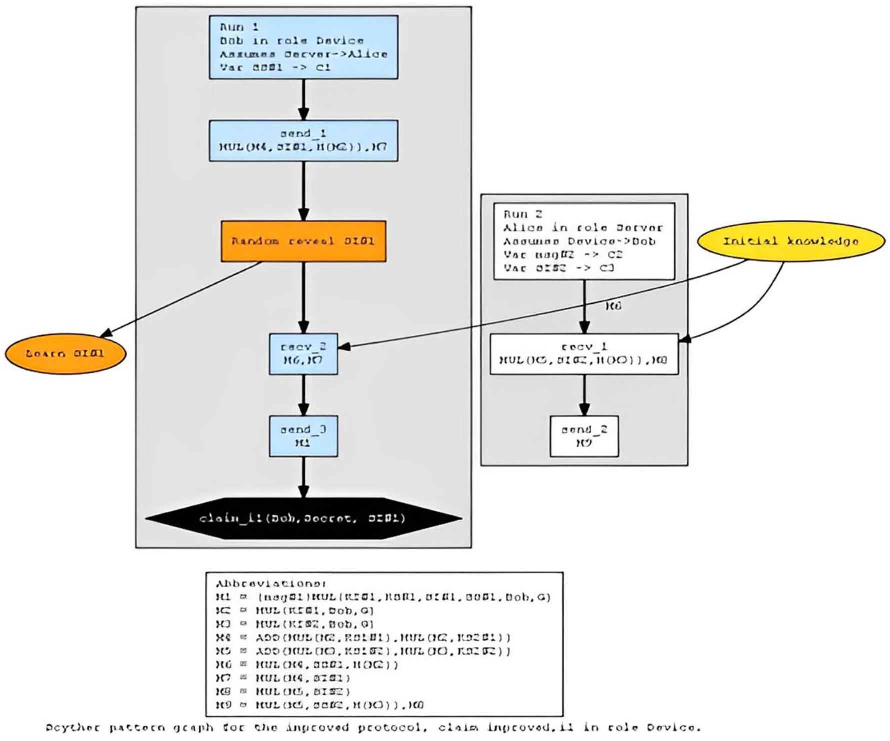


Fig. 6 Attack graph for Ummer Iqbal et al. scheme under the xck model

not safe and is defenseless to various attacks. The Scyther proof of the Ummer Iqbal et al. scheme under the xck model is displayed in Fig. 5, and attack graph is given in Fig. 6. The outcomes indicate that the system proposed by Ummer Iqbal et al. is also not secure.

Figure 6 shows the trace of the Ummer et al. method under the xck rival model which is weak against replay and session key disclosing, according to the attack simulation. As a result, the adversary can act as a legitimate device. In Ummer et al. scheme, M_1, M_2, M_3, and M_4 values are sent to the server in plain format, thus, the challenger can determine the session key which becomes a main cause to compromise the entire system.

4 Proposed security scheme

The suggested scheme covers two different phases, namely (i) Registration phase, (ii) Login phase, and Validation through the Authentication phase. An IoT device must be registered with the server during the registration step. The registration is carried out over a secure channel, much like with other methods [4]. As there are no shared credentials or a trusted third party used in the smart device, the requirement for a secure registration method is stressed. The following is a list of the actions that were conducted between the smart device in the IoT environment and the cloud server after the registration phase begins.

4.1 Registration phase

Step 1: Device SD_i selects its identification ITY_i and secret key K_{CS}.

Step 2: The smart device computes SID_i according to Eqs. 3 and 4 and transmits SID_i to CS as given in Eqs. 3 and 4.

$$SID_i = K_{cs} \cdot ITY_i \cdot G \tag{3}$$

$$ITY_i \leftarrow CS : E_{K_{CS}}(SID_i || T_1) \tag{4}$$

Step 3: CS calculates $h\left(SID_i\right)$

and generates two subkeys i.e.,S^1_{CS} and S^2_{CS} of the same size from X_{CS}. Also, ensures that $S^1_{CS} \neq S^2_{CS}$. The CS computes:

$$C^1_i = SID_i \cdot S^1_{CS} \tag{5}$$

$$C^2_i = SID_i \cdot S^2_{CS} \tag{6}$$

The cloud server stores $((SID_i, h(SID_i))$ pair along with C^1_i and C^2_i in its database and transmits:$S \rightarrow SD_i : E_{K_{CS}}(C^1_i||C^2_i)$.

Step 4: SD_i decrypts the received message and stores both C^1_i and C^2_i in it write protected memory area.

4.2 Login phase and authentication phase

The smart device starts the process of logging in to the cloud server at this phase. The server then verifies the device's identity and if it passes muster, a session key is created which can be used between SD_i and CS. The following important steps are carried out between SD_i and CS during the period of login and authentication.

Step 1: Device SD_i selects ephemeral secret (ES_i) and computes M_1 value and M_2 value as listed in Eqs. 7 and 8

.

$$M_1 = (C^1_i + C^2_i).ES_i \cdot h\left(SID_i\right) \tag{7}$$

$$M_2 = (C_i^1 + C_i^2) \cdot ES_i \tag{8}$$

Then, SD_I transmits (M_1, M_2) to CS as given in Eq. 9.

$$ITY_i \rightarrow CS : E_{K_{CS}}(M_1||M_2) \tag{9}$$

Step 2: Upon receiving the message, CS authenticates ITY_i

by executing the steps as follows.

(i) The cloud server, CS computes $h(SID_i)$ using SID_i which exists in its database.
(ii) Also, CS computes the multiplicative inverse of $h(SID_i)$ as $h(SID_i)^{-1}$ and determines($M_1.[h(SID_i)]^{-1}) = M_1^{|}$.
(iii) CS checks if ($M_1^{|} \neq M_2$) then
(iv) {

$$M_3 = M_1.ES_s \tag{10}$$

$$M_4 = (C_i^1 + C_i^2).ES_i \tag{11}$$

}

else
discard "login request".

(v) CS generates a session key,$SK_k = (M_2.ES_s) = (X_{CS}.K_{sc}.SID_i.G.ES_i.ES_s)$.
(w) CS transmits M_3 and M_4 to SD_i i.e.,$CS \rightarrow SD_i$: $E_{K_i}(M_3||M_4)$. Afterward, SDi authenticates the received data by performing the following operations.
(x) SD_i computes the multiplicative inverse of $h(SID_i)$ as $h(SID_i)^{-1}$ and determines($M_3.[h(SID_i)]^{-1}$). $ES_i^{-1} = M_3^{|}$.
(y) CS: if ($M_3^{|} \neq M_4$) then
(z) Authentication "Successful"
() else
() discard "login response"
() SD_i generates a session key,$SK_k = (M_4.ES_i) = (X_{CS}.K_{CS}.SID_i.G.ES_i.ES_s)$.
() SD_i transmits SK to SD_i i.e., $SD_i \rightarrow CS$: $E_{SK_i}(h(ES))$.
() Finally, CS decrypts the received $E_{SN_i}(h(ES))$ and checks whether the acknowledged 'h' value is not equal to the computed 'h' value i.e., $h^{|}$. If so, the smart device is legitimate.

The overall graphical diagram of the suggested method is given in Fig. 8, and Fig. 7 depicts its SPDL model.

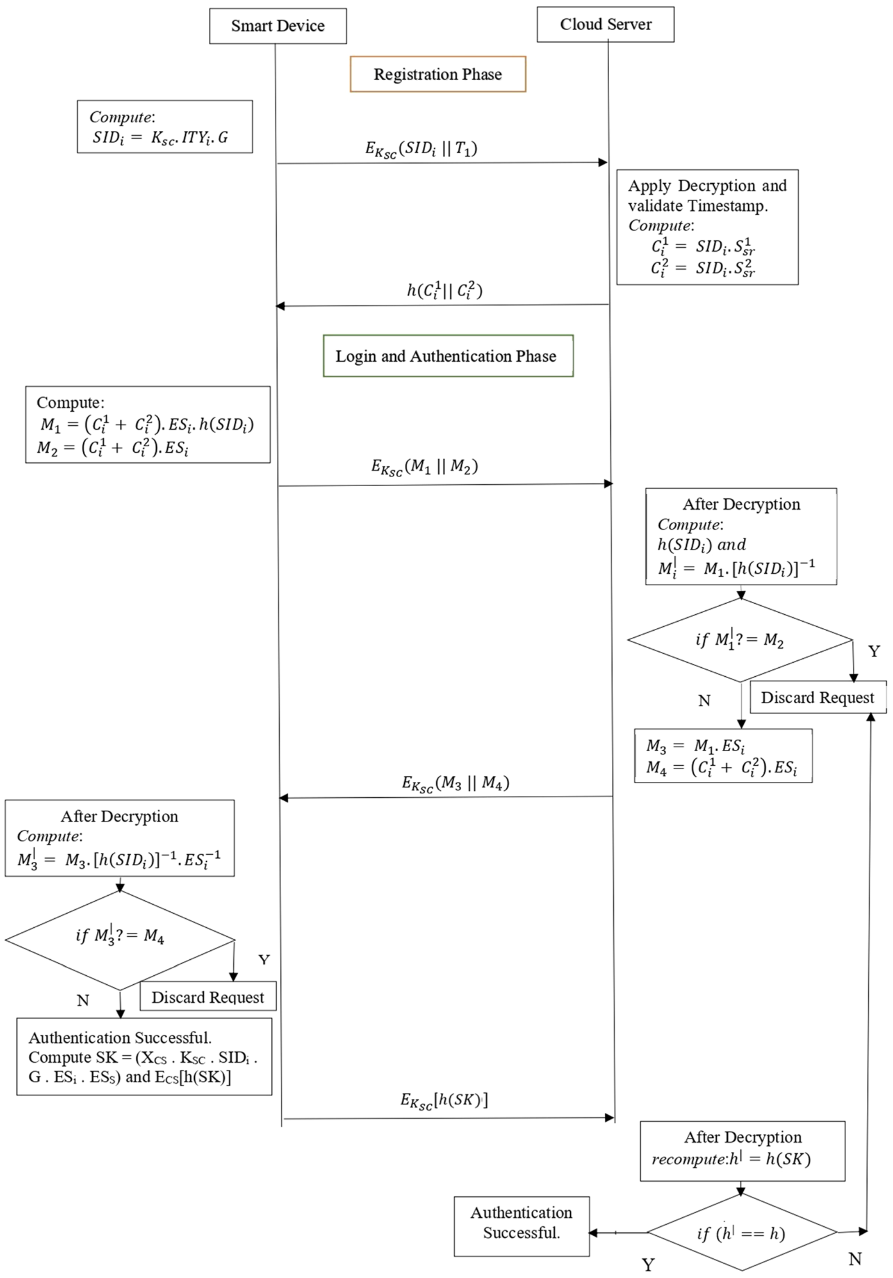
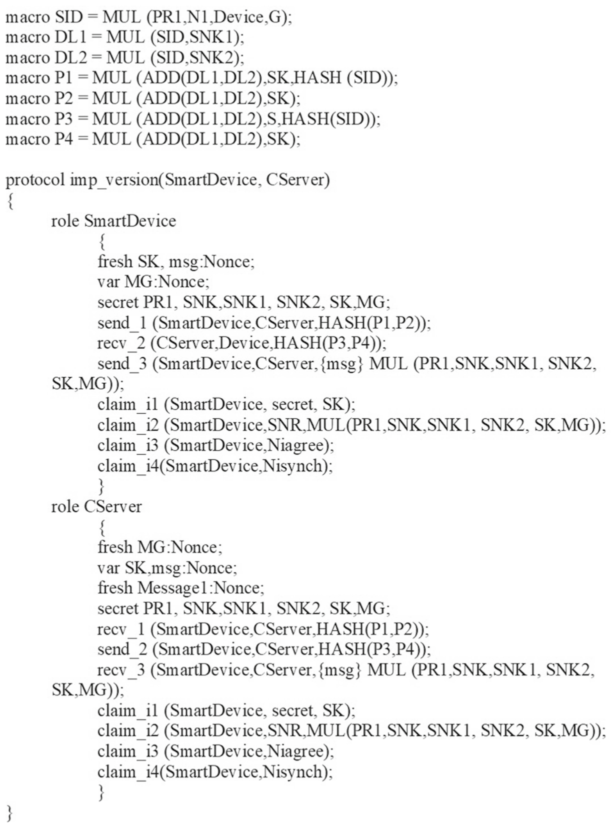

Smart Device
Cloud Server
Registration Phase
Compute: $SID_i = K_{sc}.ITY_i.G$
$E_{K_{SC}}(SID_i \| T_1)$
Apply Decryption and validate Timestamp.
Compute: $C_i^1 = SID_i.S_{sr}^1$ $C_i^2 = SID_i.S_{sr}^2$
$h(C_i^1 \| C_i^2)$
Login and Authentication Phase
Compute: $M_1 = (C_i^1 + C_i^2).ES_i.h(SID_i)$ $M_2 = (C_i^1 + C_i^2).ES_i$
$E_{K_{SC}}(M_1 \| M_2)$
After Decryption Compute: $h(SID_i)$ and $M_i^| = M_1.[h(SID_i)]^{-1}$
$if\ M_1^| ?= M_2$
Y
N
Discard Request
$M_3 = M_1.ES_i$ $M_4 = (C_i^1 + C_i^2).ES_i$
$E_{K_{SC}}(M_3 \| M_4)$
After Decryption Compute: $M_3^| = M_3.[h(SID_i)]^{-1}.ES_i^{-1}$
$if\ M_3^| ?= M_4$
Y
N
Discard Request
Authentication Successful. Compute SK = (X$_{CS}$. K$_{SC}$. SID$_i$. G . ES$_i$. ES$_S$) and E$_{CS}$[h(SK)]
$E_{K_{SC}}[h(SK)]$
After Decryption $recompute$: $h^| = h(SK)$
Authentication Successful.
$if\ (h^| == h)$
Y
N

Fig. 7 Graphical representation of the proposed method

5 Formal proof of the suggested scheme

To validate the proposed protocol's security strength on Scyther Compromise 0.9.2



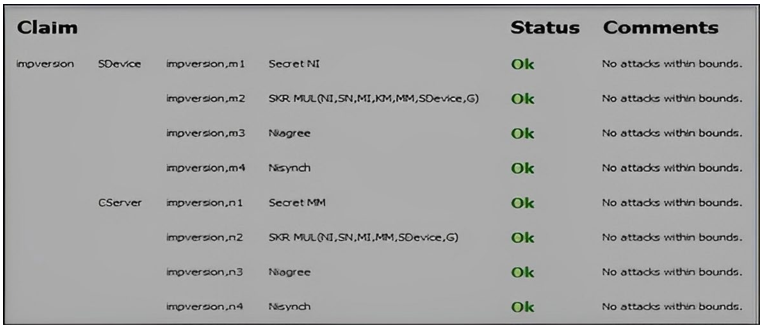
```
macro SID = MUL (PR1,N1,Device,G);
macro DL1 = MUL (SID,SNK1);
macro DL2 = MUL (SID,SNK2);
macro P1 = MUL (ADD(DL1,DL2),SK,HASH (SID));
macro P2 = MUL (ADD(DL1,DL2),SK);
macro P3 = MUL (ADD(DL1,DL2),S,HASH(SID));
macro P4 = MUL (ADD(DL1,DL2),SK);

protocol imp_version(SmartDevice, CServer)
{
        role SmartDevice
                {
                fresh SK, msg:Nonce;
                var MG:Nonce;
                secret PR1, SNK,SNK1, SNK2, SK,MG;
                send_1 (SmartDevice,CServer,HASH(P1,P2));
                recv_2 (CServer,Device,HASH(P3,P4));
                send_3 (SmartDevice,CServer,{msg} MUL (PR1,SNK,SNK1, SNK2,
        SK,MG));
                claim_i1 (SmartDevice, secret, SK);
                claim_i2 (SmartDevice,SNR,MUL(PR1,SNK,SNK1, SNK2, SK,MG));
                claim_i3 (SmartDevice,Niagree);
                claim_i4(SmartDevice,Nisynch);
                }
        role CServer
                {
                fresh MG:Nonce;
                var SK,msg:Nonce;
                fresh Message1:Nonce;
                secret PR1, SNK,SNK1, SNK2, SK,MG;
                recv_1 (SmartDevice,CServer,HASH(P1,P2));
                send_2 (SmartDevice,CServer,HASH(P3,P4));
                recv_3 (SmartDevice,CServer,{msg} MUL (PR1,SNK,SNK1, SNK2,
        SK,MG));
                claim_i1 (SmartDevice, secret, SK);
                claim_i2 (SmartDevice,SNR,MUL(PR1,SNK,SNK1, SNK2, SK,MG));
                claim_i3 (SmartDevice,Niagree);
                claim_i4(SmartDevice,Nisynch);
                }
}
```

Fig. 8 SPDL code of the proposed method

beneath the xck challenger model, it was modeled using SPDL. The roles Device and Server represent the D_i and S's communication patterns, respectively. The send1 function is utilized by the role Device to start the login phase and authentication phase. When the role server receives (M_1, M_2) using the recv1 function, it transmits (M_3, M_4) to the device using the send2 function. Finally, the smart device uses the send3 function to send $E_{SK}[H(S_K)]$ to the server. The claim_i1 belongs to the Device role and the claim_r1 resides in the server role specifying that the K_i and X_S be kept private during communication.

The claim_i2 belongs to the role device and claim_r2 resides in the role server specify whether the session key of the adversary rule can reveal $S_k = X_s.K_{sc}.SID_i.G.S_i.S_s$. The purpose of all claims is to check if the authentication operation concerning the security claim (Nisynch, Niagree) is still working. As shown in Fig. 8, the SPDL prototype of the proposed method was validated and verified on the Scyther Compromise 0.9.2 using the xck challenger model and Fig. 8 depicts its outcome. Under the strict xck adversary model,



Claim				Status	Comments
impversion	SDevice	impversion,m1	Secret NI	Ok	No attacks within bounds.
		impversion,m2	SKR MUL(NI,SN,MI,KM,MM,SDevice,G)	Ok	No attacks within bounds.
		impversion,m3	Niagree	Ok	No attacks within bounds.
		impversion,m4	Nisynch	Ok	No attacks within bounds.
	CServer	impversion,n1	Secret MM	Ok	No attacks within bounds.
		impversion,n2	SKR MUL(NI,SN,MI,MM,SDevice,G)	Ok	No attacks within bounds.
		impversion,n3	Niagree	Ok	No attacks within bounds.
		impversion,n4	Nisynch	Ok	No attacks within bounds.

Fig. 9 Verification results of the proposed scheme

Fig. 9 shows that the suggested protocol is secure and does not weak against any attacks.

5.1 Security analysis

5.1.1 Replay attack

This type of attack permits a challenger to archive and retransmit messages being exchanged between communicating parties in the future to acquire unauthorized access. Three messages are exchanged between the smart device residing in the IoT environment and the cloud server in the proposed protocol:

$$SD_i \rightarrow CS : E_{K_{sc}}(M_1||M_2) \tag{12}$$

$$CS \rightarrow SD_i : E_{K_{sc}}(M_3||M_4) \tag{13}$$

$$SD_i \rightarrow CS : E_{K_{SC}}[h(ES)] \tag{14}$$

Assume that a challenger can eavesdrop on the messages exchanged and keep (M_1, M_2, M_3, M_4) along with $E_{K_{SC}}[h(ES)]$ during the commencement of the login phase and validation through the authentication phase to perform a replay attack. Allow a challenger to replay the session values (M_1, M_2) to acquire unauthorized access as given in Eq. 15.

$$Adversary \rightarrow CS : E_{K_{sc}}[h(M_1||M_2)] \tag{15}$$

The server checks the request using the time stamp and produces M_3 and M_4 with a fresh ephemeral secret when it receives (M_1, M_2). A fresh ephemeral secret is given to the opponent. The adversary has to identify SID_i and the ephemeral secret formerly used for computation to validate (M_3, M_4) and then produce a session key. Due to the elliptic curve discrete logarithm (ECDL) problem's exponential time complexity [22–24], the attacker does not have access to either. Because the adversary is unable to provide a valid session key SK for the recent session, $E_{K_{SC}}[h(ES)]$ cannot be computed. Furthermore, the server will not validate the adversary's old $E_{K_{SC}}[h(ES)]$ because the present SK is created on the server's new ephemeral secret. In addition, a small change in one bit in ES will affect many bits in the receiver side when re-computing the hash code. As a result, the scheme's design prevents replay attacks.

5.1.2 Impersonation attack

A challenger attempts to mimic a legitimate smart device in an impersonation attack. Computing is the first step in the login process and verification process (M_1, M_2). An opponent must have access to information about (C_i^1, C_i^2) and H in order to compute (M_1, M_2) and (SID_i). Over a secure channel, the (C_i^1, C_i^2), and H(SID_i) are exchanged between the smart device residing in the IoT environment and the server. Furthermore, because of the computational difficulties of the ECDL problem, although the challenger eavesdrops on (M_1, M_2) of any preceding login process and verification of session information between the smart device residing in IoT and CS, the (C_i^1, C_i^2), and H(SID_i) can never be derived from (M_1, M_2). Thus, a result, a valid (M_1, M_2) cannot be computed without knowledge of (C_i^1, C_i^2), and H(SID_i), as a challenger cannot mimic any genuine smart device by calculating a malevolent (M_1, M_2).

5.1.3 Message integrity-based attack

The communication sent from the smart device to the cloud server cannot be disguised. During the conversation, the smart device resides in the IoT layer and CS exchange the following messages: M_1, M_2, M_3, M_4 and $E_{K_{SC}}[h(ES)]$. Assume an attacker intercepts M_1, M_2, M_3, M_4, and E_{SK}[H(SK)] and wishes to construct malevolent: MS_1, MS_2, MS_3, MS_4. However, the enemy must access X_S and K_i to generate MS_1, MS_2, MS_3, and MS_4. X_S and K_{sc} are unavailable to the adversary. Furthermore, due to the computational difficulty of the ECDL problem, extracting the private key of X_S from M_1, M_2, M_3, and M_4 takes an exponential amount of time. As a result, the computational infeasibility of the ECDL problem protects the integrity of M_1, M_2, M_3, and M_4. Furthermore, because it is secured using the symmetric key encryption algorithm and the one-way hash, the message $E_{K_{CS}}[h(ES)]$ cannot be changed.

5.1.4 Man in middle (MIM) attack

In this type, a remote invader can snoop, masquerade, and modify communication in the middle of a MIM assault by forging any sensitive information shared between the smart device and the server. If the opponent in the middle of the transmission can construct malicious: MS_1, MS_2, MS_3, and MS_4 then the MIM attack will be successful. However, the opponent must be able to forge (C_i^1, C_i^2) in order to generate malicious MS_1, MS_2, MS_3, and MS_4:

Table 2 Symbols used in BAN logic

Symbol	Remark
$X\| \equiv MG$	D trusts M
$X\| \leftarrow MG$	D obtains a message M
$X\| \sim MG$	The previously sent message by D to M
$X\|\| \sim MG$	The recent message by D to M
$X1 \rightarrow F$	D has control over Z
$\#(MG)$	M is new
$\rightarrow^{PU_r X}$	PU_r denotes the public parameter generated using PR_X
$X \rightarrow^{Y} N$	N denotes the key shared between X and Y
$\{X\}_N$	N is the key used to enciphering X
(C1 / C2)	If C1 is true then C2 also true

Table 3 BAN claims

Rules	Representation
R1	$(X\| \equiv \rightarrow^{PU_r} Y, X \leftarrow \{MG\}Y/X\| \equiv Y\| \sim MG)$
R2	$(X\| \equiv \#(MG), X \equiv Y\| \sim MG/X\| \equiv Y\| \equiv MG)$
R3	$(X\| \rightarrow MG, X\| \equiv Y\| \equiv MG/X\| \equiv MG)$
R4	$(X\| \leftarrow MG1, X\| \leftarrow Y\| \equiv MG2/X\| \leftarrow (MG1, MG2))$
R5	$(X\| \equiv MG1, X\| \equiv MG2/X\| \equiv (MG1, MG2))$
R6	$(X\| \equiv \#(MG1)/X\| \equiv \#(MG1, MG2))$
R7	$(X\| \equiv \#(N), X\| \equiv Y\| \equiv Z/X\| \equiv X \rightarrow^{Y} PR_X Y$. Z is part of N
R8	$(X\| \leftarrow MG1X\| \leftarrow (MG1, MG2)$
R9	$(X\|\equiv Y\| \sim MG1X\| \equiv Y\| \sim (MG1, MG2)$
R10	$X\|\equiv Y\| \sim (MG1, MG2)X\| \equiv Y\| \sim MG1$
R11	$X\|\equiv Y\| \sim MG1X\| \equiv \#(MG1)$

$$C_i^1 = \mathrm{SID}_i \cdot M_1 \cdot S \tag{16}$$

$$C_i^2 = \mathrm{SID}_i \cdot M_2 \cdot S \tag{17}$$

The adversary must access X_S to forge (C_i^1, C_i^1, X_s) is not accessible to the opponent. Furthermore, due to the ECDL problem's exponential complexity, X_S cannot be retrieved from M_1, M_2, M_3, or M_4. The MIM attack is alleviated in the proposed approach since the attacker cannot fake M_1, M_2, M_3, or M_4 and sufficient authentication is used prior to key formation.

Table 4 Security comparison

Scheme	Type of attack (TA)					
	TA1	TA2	TA3	TA4	TA5	TA6
Chang et al	No	No	Yes	Yes	Yes	No
Kumari et al	No	No	Yes	Yes	No	No
Rostampour et al	Yes	Yes	Yes	Yes	Yes	No
Ummer Iqbal et al	Yes	Yes	Partial	Partial	Yes	Yes
Proposed scheme	Yes	Yes	Yes	Yes	Yes	Yes

TA1- Traceability; TA2- Impersonation; TA3- Replay; TA4- Message Integrity; TA5- MIM; TA6- xck

5.1.5 DoS attack

The login processes taking place between SDi and CS can be intercepted by eavesdropper ‘A’. A can investigate the captured message to CS and attempt to conduct a DoS attack at any time after the original message has been broadcast. A cannot launch a denial-of-service attack because all sensitive data transferred between SDi and CS is protected by enciphering and hashing functions.

6 Security study through BAN logic

The suggested scheme’s security validity is assessed using BAN logic. The communication principles are denoted by X and Y, while their private keys are denoted by PRI and PRJ, respectively. The BAN symbols are listed in Table 2 [23], and Table 3 lists the BAN claims. Additionally, as deduced in [27], Table 4 lists the synthesis rules.

Assumptions A1 : $X1 \equiv\rightarrow C_i^1, C_1^2{}_I$.

A2 : $Y1 \equiv\rightarrow C_j^1, C_1^2{}_J$.

A3 : $X| \equiv \#(Y_I)$.

A4 : $Y| \equiv \#(Y_Y)$.

A5 : $Y|\equiv X| \rightarrow Y_I$.

A6 : $X|\equiv Y| \rightarrow Y_Y$.

Idealized form

$$X \rightarrow Y; \{Q_1, Q_2\}_{PR_I} MG1$$

$$Y \rightarrow X; \{Q_3, Q_4\}_{PR_J} MG2$$

Goals

$$G1 : Y| \equiv X \to^{Y} PR_X Y$$

$$G2 : Y| \equiv X| \equiv X \to^{Y} PR_X Y$$

$$G3 : X| \equiv X \to^{Y} PR_X Y$$

$$G4 : X| \equiv Y| \equiv X \to^{Y} PR_X Y$$

BAN analysis

With MG1, we obtain

$$1 : X| \equiv \{Q_1, Q_2\}_{PR_I}$$

$$2 : X \equiv \{Q_1, Q_2\}_{PR_I}$$

From (2), A1 and Rule1, we obtain

$$3 : X| \equiv Y| \sim \{P_1, P_2\}_{PR_I}$$

Y_I is part of $Q_{1,}$ Q_2. So, as per A3 and Rule6, we obtain

$$4 : X| \equiv \#(Q_1, Q_2)$$

From 3 and 4, we obtain

$$5 : Y| \equiv X| \sim (Q_1, Q_2)$$

From 7 and S4, we obtain

$$6 : Y| \equiv \#(Q_1, Q_2)$$

From 3 and 6 by applying Rule2, we obtain

$$7 : Y| \equiv X| \equiv Q_1, Q_2$$

Y_I is the part of Q_1 and Q_2. As a result, by using Rule 5, we obtain

$$8 : Y| \equiv X| \equiv Y_I$$

As a result of A5, A8, and Rule3, we now have

$$9 : Y| \equiv Y_I$$

Rules 3 and 10 give us the following

$$10 : Y| \equiv X| \sim Y_I$$

A3 and A10 gives the following

$$11 : Y| \equiv X|| \sim Y_I$$

Rule4 and Rule11 gives the following

$$12 : Y| \equiv \#(S_I)$$

S_I is a component of N. As a result, we achieve

$$13 : Y| \equiv \#(N)$$

From 8, 13 and Rule7, we obtain.
$14 : Y| \equiv X \rightarrow^{Y} PR_X\ Y$(Goal G1).
The protocol's symmetry characteristic makes it possible for,
$15 : Y| \equiv X| \equiv Y \rightarrow^{Y} PR_X\ X$(Goal G2).
From MG2, we infer that

$$16 : Y| \equiv \{Q_3, Q_4\}_{PR_J}$$

$$17 : Y \leftarrow \{Q_3, Q_4\}_{PR_J}$$

Form 17, Rule1 and A2, we obtain

$$18 : X| \equiv Y| \sim \{Q_3, Q_4\}_{PR_J}$$

The PR_J is a component of Q3 and Q4. Thus, in accordance with A4 and Rule6, we obtain

$$19 : Y| \equiv \#(Q_3, Q_4)$$

From 17 and 19, we obtain

$$20 : X| \equiv Y|| \sim Q_3, Q_4$$

From 20 and Rule11, we obtain

$$21 : X| \equiv \#(Q_3, Q_4)$$

Applying Rule 1 to 18, 21, we obtain

$$22 : X| \equiv Y| \equiv Q_3, Q_4$$

The Y_Y component of the Q3 and Q4 formulas. As a result, while using Rule 5, we obtain

$$23 : X| \equiv Y| \equiv Y_Y$$

As a result of A6, 23 and Rule3, we now have

$$24 : X| \equiv Y_Y$$

Rule 10 and Rule 18 give us

$$25: X|\equiv Y| \sim Y_Y$$

A4 and 25 give us

$$26: X|\equiv Y|| \sim Y_Y$$

From Rule4 and 26, we obtain

$$27: X| \equiv \#(Y_Y)$$

Using Rule 6, we obtain

$$28: X| \equiv \#(N)$$

Using 22, 28, and Rule 7 in combination, we arrive at

$$29: X| \equiv X\rightarrow^{Y}PR_{X}Y$$

The protocol's symmetry characteristic makes it possible for,

$$30: Y|\equiv X| \equiv X\rightarrow^{Y}PR_{X}Y$$

7 Performance comparison with existing schemes

7.1 Based on security

Table 4 compares the proposed scheme's security to that of relevant existing systems. Chang et al. solution is susceptible to impersonation and traceability attacks and only TA3 and TA4 traits are supported by Kumari et al. Rostampour et al.'s design is the only one that offers TA1–TA5 among the available schemes. Rostampour et al., Kumari et al., and Chang et al. schemes have been analyzed using the xck adversary model, according to Table 1. Rostampour et al. formal validation under xck adversary demonstrate that the technique is not secure against the xck model. In Ummer et al. scheme, values exchanged between the smart device and the server are not secured. This permits the opponent to violate the message integrity and replay attack possible (TA3 – TA4). The suggested method meets all of the TA1 to TA5 security requirements and is secure under the xck paradigm.

7.2 Based on computational overhead

Table 5 shows the assessment concerning computational overhead. T_{SM}: scalar multiplication, T_{AP}: point addition, T_{OH}: one-way hash, T_{SE}/T_{SD}: symmetric encryption, and T_{MI}: multiplicative inverse is among the time complexities considered. The most computationally intensive operation is T_{ECM}. Because smart devices in IoT are resource constrained, the computational cost of these smart devices has a significant impact on the scheme's efficiency, as the cloud server becomes computationally influential. Table 5 shows that the suggested solution requires a computational

Table 5 Comparison of computational Overhead [13]

Scheme	Computational overhead		
	Device	Server	Total
Chang et al	$4T_{SM}+4\ T_{OH}$	$4T_{SM}+4\ T_{OH}$	$8T_{SM}+8\ T_{OH}$
Kumari et al	$4T_{SM}+3\ T_{OH}$	$4T_{SM}+4\ T_{OH}$	$8T_{SM}+7\ T_{OH}$
Rostampour et al	$7T_{SM}+T_{PA}$	$6T_{SM}+T_{AP}$	$13\ T_{SM}+2\ T_{AP}$
Ummer Iqbal et al	$T_{SM}+T_{PA}+2$	$4\ T_{SM}+T_{AP}+2$	$8\ T_{SM}+2T_{AP}+4\ T_{OH}+2$
Proposed scheme	$T_{SM}+T_{PA}+1$	$3\ T_{SM}+T_{AP}+2$	$6\ T_{SM}+2T_{AP}+4\ T_{OH}+2$

T_{SM}: scalar multiplication, T_{AP}: point addition, T_{OH}: one-way hash, $T_{SE/SD}$: symmetric encryption, and T_{MI}: multiplicative inverse

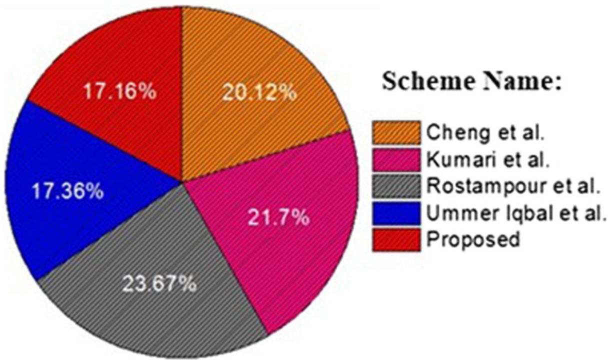


Fig. 10 Performance comparison based on Communication overhead

overhead of 1 T_{SM} for the smart device, 4 T_{SM} for the server, and 8 T_{SM} in total. Both, Chang et al. and Kumari et al. schemes require $8T_{SM}$ + 8 T_{OH} in total. The Rostampour et al. method has the largest device overhead (7 T_{SM}). As the smart device is power constrained, the proposed scheme incurs less computation compared to Ummer et al. scheme. When it comes to scalar multiplication (T_{SM}) overhead, the suggested system requires the same number of scalar multiplications as the other strategies under consideration.

7.3 Based on communication overhead

The cost of communication is calculated based on the length of bits shared among the smart device in the IoT layer and the cloud server. We assume a 160-bit elliptical curve E(a,b). The symmetric encryption used generates a 128-bit ciphertext. Three messages are exchanged in the proposed method between SD_i and the server, including (M_1, M_2), (M_3, M_4), and E_{SK}[H(SK)]. [(320 + 320), (320 + 320)+128] = 1,408 bits are required for completion. Fig. 10 shows the communication cost of the proposed scheme including existing solutions.

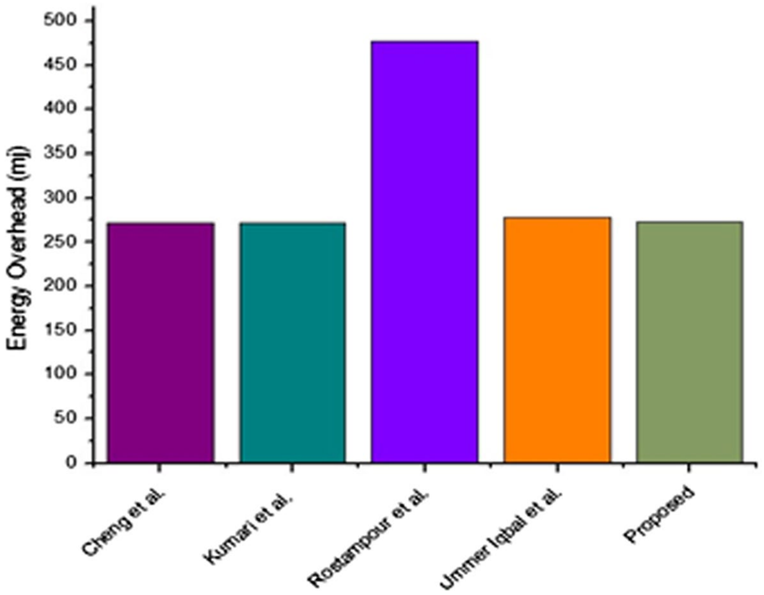


Fig. 11 Performance Analysis based on Energy overhead

7.4 Based on energy overhead

Fig. 11 shows the time spent by different important actions as mentioned in [25] on a MicaZ [26] for assessing the energy spent on a smart IoT device. E = I × V × T [13] is used to calculate how much energy is wasted, where I refers to the current strained, V denotes the voltage, and T refers to the time it takes to complete the process. For a MicaZ mote value i.e., when I = 8 mA and V = 3 V, the proposed approach has a 278.16 mJ energy overhead. Ummer et al. also incur almost the same energy overhead. Figure 11 shows a comparison of the suggested plan's energy expenditure with that of the relevant current scheme. The Rostampour et al. method has incurred maximum energy overhead at 477.6 mJ.

Under the xck rival model, Rostampour et al. are insecure. The energy overheads of Ummer et al., Rostampour et al., Kumari et al., and Chang et al. are 278.16 mJ, 477.6 mJ, 271.2 mJ, and 278.16 mJ correspondingly. In addition, Rostampour et al., Kumari et al., besides Chang et al. on the other hand, do not satisfy all security necessities and Ummer et al. scheme partially incorporated replay and message integrity security features. The proposed technique complies with all security requirements and is the only one that has been formally validated by exploring the xck rival model. Therefore, with a computational cost of 272.07 mJ and a communication cost of 1,408 bits, our suggested method is also secure against the xck adversary and satisfies all vital security requirements.

8 Conclusion

For establishing secure IoT-based smart applications, providing authentication between smart devices that specifically reside in the IoT environment and the cloud server is essential. The xck adversary model has not been used to validate the existing approaches for authentic keys shared between smart devices in IoT environments and cloud services. This article presents an improved and lightweight ECC-reliant authentication method between smart devices residing in the IoT environment and the cloud server. The suggested approach is safe and protected under the xck model, according to the Scyther Compromise simulation. In comparison to existing schemes, the proposed method incurs low communication and energy overhead. As, the cloud server is used as a trusted authority, the whole system is dependent on the cloud server which becomes the limitation of the suggested work. Hence, in future we would like to strengthen the security of the cloud server.

Authors' contributions KS: Conceptualization, Methodology, Validation, Formal analysis, Writing Supervision, Project administration. KM: Conceptualization, Software, Investigation, Data curation, Writing—review. PV: Conceptualization, Methodology, Software, Validation, Investigation, Writing—original draft, Supervision. PCSM: Validation, Investigation, Writing—editing.

Funding This research received no specific grant from any funding agency.

Data availability There is no data involved and the code is available upon request.

Declarations

Conflict of interest The authors declare no conflict of interest.

Ethical approval No human subjects or animals are involved in the study.

References

1. Porkodi R and Bhuvaneswari V, (2014) 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
2. Hassija V, Chamola V, Saxena V, Jain D, Goyal P, Sikdar B (2019) A survey on IoT security: application areas, security threats, and solution architectures. IEEE Access 7:82721–82743
3. Gupta GP (2018) Security issues and its countermeasures in examining the cloud-assisted IoT. Adv Wireless Technol Telecommun 5:91–115
4. Rostampour S, Safkhani M, Bendavid Y, Bagheri N (2020) ECCbAP: a secure ECC-reliant authentication protocol for IoT edge devices. Pervasive Mob Comput 67:101194
5. Liao Y-P, Hsiao C-M (2014) A secure ECC-RELIANT RFID authentication scheme integrated with Id-verifier transfer protocol. Ad Hoc Netw 18:133–146
6. Roel P and Hermans J, 2013 Attack on liao and Hsiao's secure ECC-reliant RFID authentication scheme integrated with IDverifier transfer protocol, IACR Cryptology, vol 399
7. Kalra S, Sood SK (2015) Secure authentication scheme for IoT and cloud servers. Pervasive Mob Comput 24:210–223

8. Chang C-C, Wu H-L, Sun C-Y (2017) Notes on "Secure authentication scheme for IoT and cloud servers." Pervasive Mob Comput 38:275–278
9. Kumari S, Karuppiah M, Das AK, Li X, Wu F, Kumar N (2017) A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. De J Supercomput 74(12):6428–6453
10. Wang K-H, Chen C-M, Fang W, Wu T-Y (2017) A secure authentication scheme for Internet of Things. Pervasive Mob Comput 42:15–26
11. Saqib N and Iqbal U, Security in wireless sensor networks using ECC, IEEE International Conference on Advances in Computer Applications (ICACA), pp 270–274, 2016.
12. Zhao Z, Hsu C, Harn L, Yang Q, Ke L (2021) 2021 Lightweight privacy-preserving data sharing scheme for Internet of medical Things. Wireless Commun Mobile Comput 8402138:13
13. Iqbal U, Tandon A, Gupta S, Yadav AR, Neware R, Gelana FW (2022) A Novel secure authentication protocol for IoT and cloud servers. Wireless Commun Mobile Comput 2022:1–17
14. Dolev D and Yao A., (1981) 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
15. Dolev D, Yao A (1983) On the security of public-key protocols. IEEE Trans Inf Deory 29(2):198–208
16. Canetti R, Krawczyk H (2001) Analysis of key-exchange protocols and their use for building secure channels. Lect Notes Comput Sci 2045:453–474
17. Cremers CJ (2008) The scyther tool: verification, falsification, and analysis of security protocols. Comput Aided Verification 5123:414–418
18. Clement Chan Z, Chuah C, Alawatugoda J (2019) Review on leakage resilient key exchange security model. Int J Commun Netw Inf Security IJCNIS 11:1
19. Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48(177):203–209
20. Miller VS (1986) Use of elliptic curves in cryptography. Lecture Notes Comput Sci Adv Cryptol 1985:417–426
21. Hankerson D, Menezes A (2011) Elliptic curve cryptography. Encyclopaedia of Cryptography and Security, Boston, Springer, US, p 397
22. Gura N, Patel A, Wander A, Eberle H, Shantz SC (2004) Comparing elliptic curve cryptography and RSA on 8-bit CPUs. Lect Notes Comput Sci 3156:119–132
23. Islam SH, Biswas GP (2017) A pairing-free identity-based two-party authenticated key agreement protocol for secure and efficient communication. J King Saud Univ Comput Inf Sci 29(1):63–73
24. Malan DJ, Welsh M, Smith MD (2008) Implementing public-key infrastructure for sensor networks. ACM Trans Sen Netw 4(4):1–23
25. Iqbal U, Hussain Mir A (2020) Secure and practical access control mechanism for WSN with node privacy. J King Saud Univ Comput Inf Sci 34(6):3630–3646
26. Mote Works, Getting Started Guide, Memsic, Inc., 2013.
27. L. Buttyan, S. Staamann, and U. Wilhelm, (1998) "A simple logic for authentication protocol design," In: Proceedings of the 11th IEEE Computer Security Foundations Workshop, pp 153–162

Authors and Affiliations

K. Selvi[1] · K. Muthumanickam[2] · P. Vijayalakshmi[3] · P. C. Senthil Mahesh[4]

K. Muthumanickam
muthumanickam@kongunadu.ac.in

P. Vijayalakshmi
viji.vietw@gmail.com

P. C. Senthil Mahesh
pcsmvmkv@gmail.com

[1] Department of Information Technology, Paavai Engineering College, Pachal, Namakkal, Tamilnadu, India

[2] Department of Information Technology, Kongunadu College of Engineering and Technology, Tholurpatti, Tiruchirappalli, Tamilnadu, India

[3] Department of Computer Science and Engineering, Knowledge Institute of Technology, Salem, Tamilnadu, India

[4] Department of Computer Science and Engineering, Excel Engineering College, Komarapalayam, Namakkal, Tamilnadu, India