Wireless Personal Communications (2021) 116:1867–1896
https://doi.org/10.1007/s11277-020-07769-2

ECC-CoAP: Elliptic Curve Cryptography Based Constraint Application Protocol for Internet of Things

Suman Majumder[1] · Sangram Ray[1] · Dipanwita Sadhukhan[1] · Muhammad Khurram Khan[2] · Mou Dasgupta[3]

Published online: 8 September 2020

Abstract
Constraint Application Protocol (CoAP), an application layer based protocol, is a compressed version of HTTP protocol that is used for communication between lightweight resource constraint devices in Internet of Things (IoT) network. The CoAP protocol is generally associated with connectionless User Datagram Protocol (UDP) and works based on Representational State Transfer architecture. The CoAP is associated with Datagram Transport Layer Security (DTLS) protocol for establishing a secure session using the existing algorithms like Lightweight Establishment of Secure Session for communication between various IoT devices and remote server. However, several limitations regarding the key management, session establishment and multi-cast message communication within the DTLS layer are present in CoAP. Hence, development of an efficient protocol for secure session establishment of CoAP is required for IoT communication. Thus, to overcome the existing limitations related to key management and multicast security in CoAP, we have proposed an efficient and secure communication scheme to establish secure session key between IoT devices and remote server using lightweight elliptic curve cryptography (ECC). The proposed ECC-based CoAP is referred to as ECC-CoAP that provides a CoAP implementation for authentication in IoT network. A number of well-known cryptographic attacks are analyzed for validating the security strength of the ECC-CoAP and found that all these attacks are well defended. The performance analysis of the ECC-CoAP shows that our scheme is lightweight and secure.

Keywords Internet of Things (IoT) · Elliptic curve cryptography (ECC) · Constraint application protocol (CoAP)

1 Introduction

Internet of Things and CoAP: Internet of Things (IoT) is an infrastructure of the connected smart objects like—sensor(s), actuator(s), RFID Tags, tiny microprocessor(s), communication device(s), power source(s) etc. called things which are connected through

✉ Sangram Ray
sangram.ism@gmail.com

Extended author information available on the last page of the article

wireless (IEEE 802.15.4, WiFi, Bluetooth Low Energy, Internet, cellular communication etc.) or wired connection for data communication [1–4]. The term 'Internet of Things' was initially recommended by Kevin Ashton in the year 1999 [3]. It is a global dynamic network infrastructure with self-configuring capabilities and supported by various protocols used in communication [1, 2]. IoT uses unique addressing schemes where IoT devices are able to interact with each other for common goals [2, 3]. In this regard, IPv6 is used to provide a unique IP address to each IoT device in the network [1, 3, 5]. For the non-IP situation, ZigBee, Z-Wave etc. are used for setting up connection and communication purposes [3] and RFID (Radio Frequency Identification) is used to identify the physical objects and track its location [2]. In general, following protocols are used in the five layers of IoT [1, 4, 6, 7]—(i) IEEE 802.15.4 protocol is used for both physical and MAC layer, (ii) IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) is used in adaptation layer [2, 7], (iii) Routing Over Low Power & Lossy (ROLL) and IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) are used in network layer and (iv) CoAP is used in application layer. However, due to some limitations of IEEE 802.15.4 protocol and to get high data transfer rate, it is better to associate IPv6 with the 6LoWPAN protocol instead of IEEE 802.15.4 protocol in IoT network. It is to be noted that the Maximum Transfer Unit (MTU) of an IPv6 packet associated with 6LoWPAN is minimum 1280 bytes while the MTU of an IPv6 packet associated with IEEE 802.15.4 protocol is maximum 127 bytes [5, 8].

In application layer, the CoAP is mainly used for secure communication between the constraint smart IoT devices and server because MQTT protocol [9] has some limitations such as it can be used only for very low processor devices and can communicate mainly for Amazon cloud applications for server [10]. CoAP uses RESTful architecture [7] to access the resources from server through URI(Universal Resource Identifier) and message communication. Thus, CoAP architecture is divided into two layers—(1) message layer and (2) request/response layer. The *message layer* is responsible for controlling the exchange of messages between devices over UDP (User Datagram Protocol). The *request/response layer* is responsible for handling the requests of IoT devices and corresponding responses from other devices/server through message communication and also maintains the status of the messages like *out of order*, *lost* or *duplicated* etc. [2, 3, 5, 6, 11]. The *request/response layer* is also responsible for manipulating the resources by using one of the various transmission methods such as GET, PUT, POST and DELETE [6, 7, 12].

Literature Review: Initially, Villaverde et al. [13] proposed that the combination of CoAP/UDP can be associated with DTLS for reliable negotiation of a session, verification and exchanging of packets between IoT devices. However, UDP is unreliable since it does not maintain any specific procedure for setting up reliable connection between two devices. On the other hand DTLS packets cannot be translated directly to TLS and vice versa. To resolve these issues authors further incorporates a proxy, 6LoWPAN Border Router (6LBR), for direct mapping between HTTP and CoAP i.e. application conversion between DTLS and TLS respectively. However, ensuring end-to-end security and key management of CoAP are not considered in this scheme [13].

Further, Moritz et al. [14] and Schneider et al. [15] proposed that both SOAP (Service Oriented Architecture) protocol and the XML standard (version utf-8) can be used along with TCP/UDP for secure communication with server. However, SOAP has some limitations like it affords—(1) huge number of cross domain protocol features, (2) complex data representation and (3) composite data transportation mechanisms compared to CoAP that implies an overhead for web services. Hence, SOAP cannot be applied in IoT network [14–16]. To overcome these issues, the combination of CoAP/UDP and a data compression

technique such as EXI (Efficient XML Interchange) can be used for possible minimization of payloads for constrained IoT devices.

In 2015, Bhattacharyya et al. [17] proposed that DTLS uses any one of the three security modes—(1) pre-shared key mode, (2) raw public key mode and (3) certificate mode. Among these modes pre-shared key mode is the most low-overhead option since it is based on symmetric key based encryption technique [17]. Authors also mentioned that DTLS has a major drawback for IoT network since it is not compatible with lightweight multicast security [17]. As a solution, they have mentioned Lightweight Establishment of Secure Session (LESS) protocol which may be used to maintain secure communication for message exchange and establish a secure session between IoT devices and server.

In 2015, Granjal et al. [18] proposed that the IPsec protocols [8, 18] along with a security standard X.805 can be used as a replacement of DTLS for the secure CoAP implementation. However, there are some limitations such as—(1) the combination of IPSec and X.805 may not be compatible to meet the security requirements in CoAP, (2) the space limitation and complexity of IPSec and (3) CoAP provides lightweight reliability due to transport of the messages over UDP [18]. Further, considering the functionality limitations of IPSec in wireless communication and web applications, Ray et al. [19], Johnson et al. [20] and Levi et al. [21] focused on the use of Wireless Application Protocol (WAP) along with Wireless Transport Layer Security (WTLS) protocol [19, 21] as a replacement of IPSec/TLS/DTLS. WAP is mainly used in various constraint devices like mobile phone, PDA etc. However, WAP has some limitations such as—(1) functional complexity and (2) security related problems of WTLS like end-to-end security weakness and man-in-the-middle attack at the WAP gateway [16, 19–21]. To remove these limitations and reduction of bandwidth requirements, CoAP protocol can be used.

In 2016, Rahman and Shah [6] and later on in 2017 Raza et al. [22] proposed that CoAP/UDP combination along with the encryption techniques—DTLS/IPSec are used to afford confidentiality, integrity, authentication and non-repudiation by means of the following four security modes—(1) *NoSec*: In this mode, no security service is offered. (2) *Pre-shard Key:* symmetric keys are generally used in this approach. (3) *Raw Public Key*: In this mode, authentication is based on public keys and no certificate is used for authentications purposes. A session can be set off for DTLS with pre-shared list of keys for the communicating devices. (4) *Certificates:* In this mode asymmetric key along with the certificate standard X.509 is used for validation purpose. However, ECC is another public-key cryptography which supports for both the Certificates and Raw Public Key modes. Hence, ECC based system can be used for pre-shared key (PSK) based system while it is integrated with CoAP related environment. Moreover, DTLS can be integrated in *Raw Public Key* mode for server communication using CoAP [6, 22]. However, the drawback of the CoAP security is key management [7] and the multi-cast messaging which is used to transmit between two hops/objects [6, 22] is not supported by DTLS.

In 2017, Iglesias-Urkia et al. [23] proposed that different open source libraries from different platforms like—C, Java, Python, Java Script etc. are used for the implementation of CoAP related environment. However, based on the performance of server, authors have also analyzed that the C language based open source libraries like ‘lipcoap’ or ‘smcp’ are very user friendly and most of them are surrounded by inbuilt libraries and it handles the response code and maintains handler for those responses. On the other hand, ‘lipcoap’ or ‘smcp’ are the fastest libraries among the other libraries from different platforms like Java, Java Script etc. In 2017 Alabas et al. [24] anticipated a review based on the different existing architecture related to IoT security, vulnerabilities and CoAP based communication with server. Some of the architectures are—(1) SDN Architecture, (2) SEA Architecture,

(3) Smart City, (4) Service Oriented Architecture (SOA), (5) OSCAR (Object Security Architecture) with CoAP, (6) Black SDN Architecture etc. However, based on the review of authors it is found that those architectures are still facing some problem related to multicasting, asynchronous data communication, caching, lack of strong encryption techniques, authentication and key management issues related to CoAP header etc.

In 2018, Albalas et al. [25] presented the performance evaluation between the CoAP using ECC and CoAP using RSA based on three different factors—(i) message length, (ii) security services and (iii) residual energy. It is mentioned that the CoAP using the ECC is 47% efficient in saving energy than CoAP using RSA due to smaller key size of ECC.However, the CoAP using ECC is still facing some problems related to multicasting, asynchronous data communication and key management issues related to CoAP [25]. This study has motivated us to develop ECC-CoAP protocol eliminating most of the aforementioned limitations.

In 2019, Harish et al. [26] establishes a secure connection using HTTP between IoT nodes and handles the HTTP request through a proxy which is referred to 6LoBR and maintains the security issues for the CoAP layer by encrypting/decrypting the payload of the corresponding CoAP request/response using ECC. It is managed by an IoT controller which maintains the whole traffic of the wireless network. However, it is found that the scheme [26] is suffering from some key management issues of CoAP.

Recently in 2019, Dey and Hossain [27] have proposed session key establishment for smart home network using LESS [17] protocol and shown that existing LESS protocols are not safe against relevant security attacks [28].

Contribution of the Research: To address all the issues raised from the above discussion on existing CoAP related schemes, we are motivated to design a secure and efficient CoAP using ECC for end-to-end communication and efficient key management between the IoT devices and remote server. We have referred this scheme as ECC-CoAP.

Organization of the Paper: The rest of the paper is organized as follows. Section 2 provides the basic overview of ECC and CoAP. The step-wise ECC-CoAP is proposed in Sect. 3. The security and performance analysis of the proposed scheme are discussed in Sects. 4 and 5 respectively. In Sect. 6, the simulation result of formal security verification of ECC-CoAP using AVISPA tool is conferred. Finally, Sect. 7 concludes the paper.

2 Preliminaries

In this section, the fundamental concepts of ECC and CoAP are illustrated briefly.

2.1 CoAP Overview

In application layer, the constraint application protocol CoAP is mainly used for secure communication between the IoT devices and server. CoAP architecture is divided into two layers—(i) message layer and (ii) request/response layer. The *message layer* is responsible for controlling the exchange of messages between devices over User Datagram Protocol (UDP). The *request/response layer* is accountable for handling the requests of objects and corresponding responses from other objects/server through message communication [2, 3, 5, 6, 9] by using one of the following transmission methods such as GET, POST, PUT and DELETE [6, 7, 10].

GET	It is used for CoAP related applications to retrieve resources through URI requests
POST	It generates a new request for the creation of new subroutine to the server under the parent URL for communication with the resource.
PUT	This method is used to communicate with the resources based on the request transferred within the message body. So, there is no separate URL request is required
DELETE	It is used for deletion of the URI requests

In IoT architecture, instead of HTTP protocol CoAP is generally used due to the following:

(i) Due to limited size and bandwidth requirement of constraint devices of IoT architecture, HTTP cannot be used. Being a compressed version of HTTP, CoAP provides required size and bandwidth for constrained devices.
(ii) HTTP protocol incurs more space consumptions where as CoAP is a lightweight protocol in terms of space overhead.
(iii) HTTP generally uses connection oriented reliable TCP protocol where as CoAP uses connectionless UDP.UDP is unreliable that simply carries messages and transmits lost packets.
(iv) HTTP protocol is designed for internet-based applications and devices where there is no constraint of power consumption. In IoT environment the nodes are generally power-constrained this makes CoAP suitable for IoT environment.

In general, CoAP uses a pre-shared key (PSK) along with DTLS for secure communication between IoT devices and server [12]. For transfer of data between IoT devices and server, CoAP messages are developed and used as per the specific formats [7] such as—Java Script Object Notation (JSON), Extensible Markup Language (XML), Concise Binary Object Representation (CBOR) etc. These can be secured by using various encryption formats such as—JavaScript Object Signing and Encryption (JOSE), XML-Security (XMLS), Constrained Object Signing and Encryption (COSE) etc. [7] (Table 1).

In 2015, Bhattacharyya et al. [17] proposed LESS (Lightweight Establishment of Secure Session) protocol for constraint devices which are useful for secure communication between the application layer and transport layer as well as establishment of secure session for CoAP. It pursues the following six steps of action for the establishment of secure session and switches the control from CoAP to DTLS in order to afford the security of the respective channel.

In our scheme we have reduced the number of steps to five of LESS protocol proposed by Bhattacharyya et al. [17] and incorporated ECC to develop ECC-CoAP protocol.

2.2 The Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) was proposed by Victor Miller and Neal Koblitz in 1985 and 1987 respectively [29–32].An elliptic curve E over a prime finite field F_p which is denoted as E/F_p and is defined by the following elliptic curve equitation:

$$y^2 mod\, p = \left(x^3 + ax + b\right) mod\, p \tag{i}$$

where $a, b, x, y \in F_p$ hat satisfies the equation of the discriminant D where $D=4a^3+27b^2 \ (mod\ p)\neq 0$. The additive elliptic curve group curve group G_p is defined

Table 1 Different steps of LESS protocol [17]

Step No.	Action	Description
Step 0	Pre-sharing secret	Secret is shared between the user and server offline
Step 1	Session initiation	Client sends a HELLO message which contains a unique random number generated from user/client and client ID
Step 2	Server challenge	Server sends a challenge (server_rand) to user along with session key to be used for client-side encryption
Step 3	Client response and challenge	Client sends the encrypted message with client challenge (client_rand) along with the server challenge (server_rand) to server. Here server challenge denotes the challenge sent by the server to the client in *Step 2*
Step 4	Client authentication & server response	Server verifies the server_rand with its own copy and authenticates the client and sends message encrypted with session key along with client_rand
Step 5	Server authentication	Client verifies the client_rand and authenticates the server

as $\{(x, y) : x, y \in F_p \text{ and } (x, y) \in E/F_p\} U \{O\}$ where the point *"O"* is known as *"point at infinity"*.

Point Addition: Let P, Q be two points on the elliptic curve given in equation(i) then, the straight line joining P and Q i.e. $P+Q=R$, where the straight line intersects the equation (i) at the point $(-R)$ which reflects at point R with respect to x-axis [33].

Point Subtraction: If the point $Q=(-P)$, then the line formed by joining by P and Q, i.e. $P+Q=P+(-P)=O$, i.e. the line joined by P and $(-P)$ which intersects the equation (i) at the point O which is called point of infinity [34].

Point Doubling: It is the process of addition of point P on equation (i) with itself to obtain another point Q on equation (i). Let $P+P=2P$ and $Q=2P$. If a tangent straight line is drawn at point P, then it intersects the curve of equation (i) at point $(-Q)$. The reflection of this point with respect to x-axis is at point Q [35].

Scalar point Multiplication: The scalar point multiplication is based on the concept of cyclic group G_P which is defined as $Q = x.P = P + P + P \cdots + P$ *(x times)*, where $x \in {}_RZ_p^*$ and P is a generator of the cyclic group [35].

Security strength of *ECC* reclines on the difficulty of solving the *Elliptic Curve Discrete Logarithm Problem (ECDLP)* that provides same level of security strength like *RSA* but with lesser bit-size key. Similar to *Diffie–Helman Problem* (*DHP*), ECDLP is based on the discrete logarithm problem and does not pursue any polynomial time algorithm. In *ECDLP*, two elements P and Q are taken from a random instance $(P, Q) \in G_p$, where G_P is a cyclic group. It is impossible to find an integer $q \in {}_RZ_p^*$ such that $Q=q.P$ by a polynomial time bounded algorithm where P is the generator for the cyclic group G_P.

3 The Proposed Scheme

In this paper, we have overcome the limitations of key management of CoAP [18] using ECC and a fresh ECC-CoAP protocol is developed by improving the efficiency of LESS protocol [17] with reduced the number of steps. This scheme is mainly useful for resource constrained IoT environment for secure communication between IoT devices and server with reduced communication overhead. The proposed scheme contains five steps—(i) Session initiation (ii) Server challenge phase (iii) Client response and challenge phase (iv)Client authentication and server response phase and (v) Key negotiation and server authentication phase. The entire working procedure is explained in the following subsections where the subsequent notations illustrated in Table 2 are used. The steps involved in proposed ECC-CoAP protocol are demonstrated in briefly in Table 3.

Now the pre-requisite of the proposed ECC-CoAP protocol and its step-wise working procedure are explained in the following sub-sections.

3.1 Pre-requisite of ECC-CoAP

Initially, the server selects an elliptic curve $E_p(a,b)$ over a prime finite field F_p, where P is the generator of order n. Next, the private key as $q_S \in \mathbb{Z}_p^*$ selected by the server and calculates its public key as $Q_S=q_S.P$ using ECC based scalar point multiplication (ECPM). Similarly, user/IoT device randomly selects a large random number $q_U \in \mathbb{Z}_p^*$ such as $0<q_U<n$ as a private key of the user/IoT device and generates the public key Q_U as $Q_U=q_U.P$. The user/IoT device then gets the ECC based public key certificate CA_U, combining its identity ID_U and public key Q_U from the certificate authority CA.

Table 2 Notations and respective descriptions

Notation	Description
U	User/IoT device
S	Server
F_p	A large prime finite field over p
$E_p(a,b)$	An elliptic curve is defined on F_p
P	A generator point on $E/F_p(a,b)$ with order n
ID_U	User's identity
PW_U	Random password selected by the user
DID_U	Dynamic login identity of the user/device generated by server
K	Common key used for Encryption/Decryption for both user/IoT device and server end
S_K	Dynamic session identity generated from server end
r_U	Random number selected by the user
r_S	Random number selected by the server
R_U	Random value produced by the user where $R_U = r_U.Q_U$
R_S	Random value produced by the server where $R_S = r_S.Q_S$
$h(.)$	One way secure hash function such as *SHA1*
E/D	Symmetric encryption/decryption algorithm
(q_S, Q_S)	Private–public key pair of server where $Q_S = q_S.P$
(q_U, Q_U)	Private–public key pair of user/IoT device where $Q_U = q_U.P$
|	Concatenation

3.2 Working Procedures of ECC-CoAP

The detail step-wise working procedures of ECC-CoAP for communication between the user/IoT device and server is shown in Fig. 1 and illustrated below where $X \rightarrow Y$: M denotes that sender X sends a message M to receiver Y.

Step 0: $\mathrm{U} \rightarrow \mathrm{S}: ID_U, CA_U, E_{K_X}(H_U), T_1$

Initially, user/IoT device generates a random high entropy password PW_U. Then user/IoT device computes (i) the symmetric shared key K between user/IoT device and server as $K = q_U.Q_S = q_U.q_S.P = (K_X, K_Y)$ where q_U and Q_S are the private key of user/IoT device and public key of server respectively, and (ii) $H_U = h\,(ID_U||PW_U||q_U)$where h is a one way irreversible cryptographic hash function and encrypts H_U using K_X. Finally, it sends a *session initiation request* containing ID_U, CA_U, encrypted H_U and T_1 to server.

Step 1: $\mathrm{S} \rightarrow \mathrm{U}: ID_S, E_{K_X}(DID_U||R_S), T_2$

After receiving the session initiation request from user/IoT device in time T_2, server checks $|T_2 - T_1| \leq \Delta T$? If yes the server retrieves the user's identity ID_U and public key Q_U from CA_U and checks retrieved ID_U = received ID_U? If fails the communication is terminated; Otherwise, the server (i) calculates the symmetric shared key $K = q_S.Q_U = q_S.q_U.P = (K_X, K_Y)$,(ii) decrypts the encrypted message using K_X and gets H_U, (iii) generates

Table 3 Different steps of ECC-CoAP

Step No.	Action	Description
Step 0	Session initiation	User certificate along with random hashed value of user's masked identity is shared to server for the session initiation as HELLO message
Step 1	Server challenge	Server sends the dynamic identity and server's random value to user as a server challenge
Step 2	Client response and challenge	User/IoT device sends its own random value and calculated dynamic identity as a response challenge to the server
Step 3	Client authentication and server response	Server authenticates the legitimacy of the user/IoT device and based on the authentication parameters calculates the session key to encrypt secret values to user/IoT device
Step 4	Key negotiation and server authentication	User/IoT device also calculates the session key and verifies the integrity of the decrypted server's secret values and the successful verification confirms the authenticity of the server

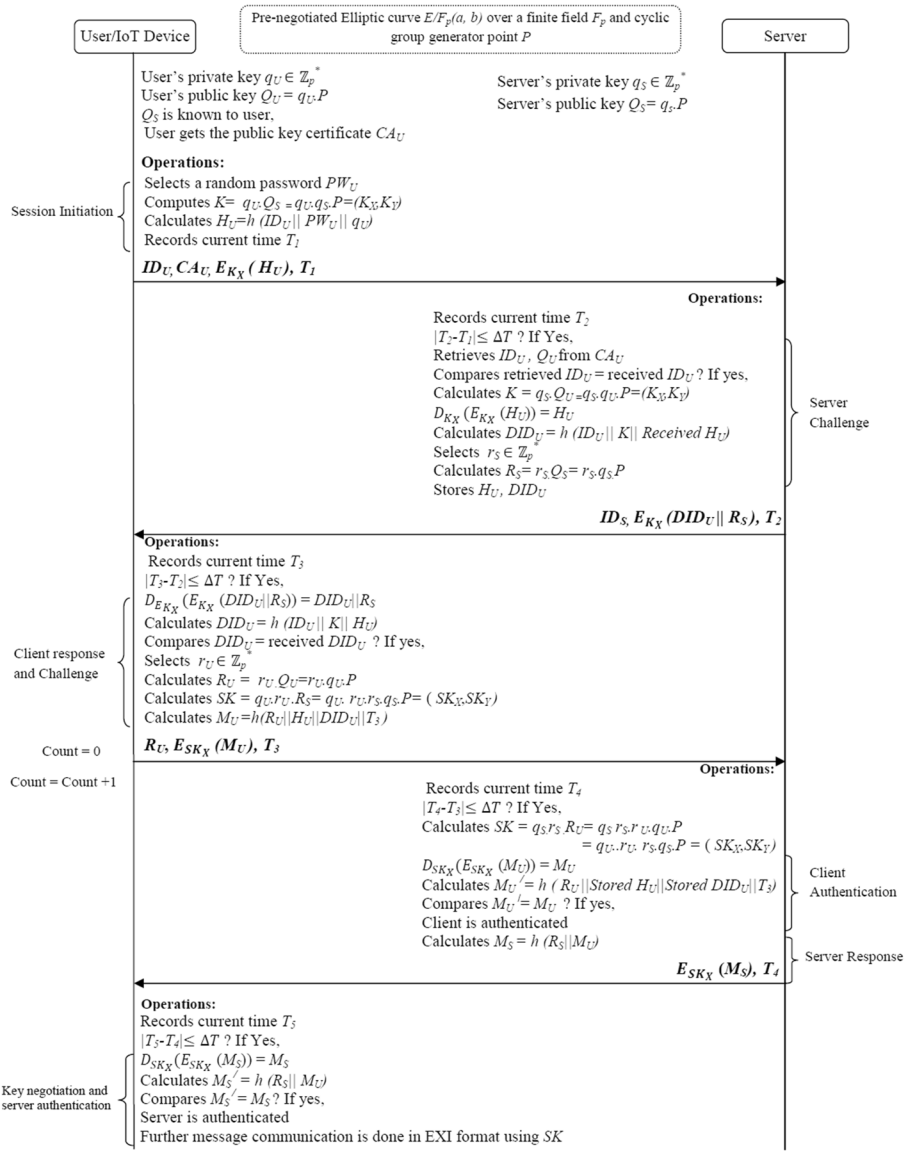

Fig. 1 ECC-CoAP diagram for maintenance of secure session and authentication

a dynamic identity of the user/IoT Device $DID_U = h(ID_U||K||H_U)$,(iv) selects a random number $r_S \in \mathbb{Z}_p^*$ to calculate the server's random point $R_S = r_S.Q_S = r_S.q_S.P$ using ECPM,(v) stores H_U and DID_U at the server's database for future reference,(vi) concatenates DID_U and R_S, then the concatenated message is encrypted using symmetric key K_X and finally (vii) sends the ID_S, encrypted message and T_2 to IoT device as *server challenge*.

Step 2: U → S:R_U,$E_{SK_X}(M_U)$, T_3

The IoT device receives the server's challenge in time T_3 and verifies the legitimacy of the server's challenge i.e. checks $|T_3 - T_2| \leq \Delta T$? If yes, the IoT device decrypts the encrypted server challenge using K_X and gets DID_U and R_S. Now, it(i) calculates dynamic identity $DID_U = h\ (ID_U||K||H_U)$and (ii) compares the calculated DID_U with received DID_U. If the comparison is unsuccessful the communication is terminated; otherwise, the IoT device selects a random number $r_U \in \mathbb{Z}_p^*$ and calculates a random point $R_U = r_U.Q_U = r_U.q_U.P$. It then calculates the session key as $SK = q_U.r_U.R_S = q_U.r_U.r_S.q_S.P = (SK_X, SK_Y)$ and $M_U = h(own\ R_U||own\ H_U||DID_U||T_3)$. Now it encrypts M_U using the recently calculated session key SK_X and sends the encrypted message with R_U and T_3 as a response to server's challenge.

A variable *count* is initialized with *0* and incremented with *1* after each unsuccessful response message transmission. Each IoT device is allowed to get *3* attempts to authenticate to server otherwise the device will be blocked for a specific period of time. This method is implemented to stop cryptographic attacks like brute-force attack.

Step 3: S → U:$E_{SK_X}(M_S)$, T_4

The server receives the client's challenge in time T_4 and checks $|T_4 - T_3| \leq \Delta T$? If yes, server (i) calculates the session key $SK = q_S.r_S.R_U = q_S.r_S.r_U.q_U.P = q_U.r_U.r_S.q_S.P = (SK_X, SK_Y)$, (ii) decrypts the encrypted client challenge and gets M_U as $D_{SK_X}(E_{SK_X}(M_U)) = M_U$ (iii) calculates $M'_U = h(received\ R_U\ ||Stored\ H_U||Stored\ DID_U||T_3)$and (iv) checks $M'_U = M_U$? If both are equal, the IoT device is authenticated to server.

Now the server calculates $M_S = h\ (R_S||M_U)$, encrypts M_S using session key SK_X and finally sends the encrypted M_S and the current timestamp T_4 as a server's response to IoT device.

Step 4: U → S : *Message communication M is done in EXI format*

The IoT device receives the server response in time T_5 and checks $|T_5 - T_4| \leq \Delta T$? If yes, IoT device decrypts the encrypted server's response using session key SK_X and gets M_S as $D_{SK_X}(E_{SK_X}(M_S)) = M_S$. Now it calculates $M'_S = h(received\ R_S||sent\ M_U)$ and checks $M'_S = M_S$? If both are equal, the server is authenticated to IoT device; otherwise the communication is terminated. All the further message communication M is done in *EXI* format using SK_X between the server and IoT device.

4 Security Analysis

All the relevant security features and security attacks are considered in this section to prove the robustness of the proposed ECC-CoAP. The ECC-CoAP is formally verified using well-known BAN logic as well as using mathematical procedures. Finally the result demonstrates that the scheme is well protected against all relevant security breaches and preserves all the significant security features. The following subsections describe—(i) Informal security analysis and (ii) Formal security analysis using BAN logic.

4.1 Informal Security Analysis

This section illustrates informal security analysis of ECC-CoAP protocol using mathematical procedures. Some practical assumptions are taken into account for proving the security strength of the protocol as given in the literature [36–42].

4.1.1 Man-in-the-Middle Attack

Let an adversary $\tilde{A}$, present between user/IoT device and server, intercepts the session initiation message containing $ID_U, CA_U, E_{K_x}(H_U), T_1$ and intends to modify it in such a way that it seems to be coming from a legitimate user containing valid identity ID_U of the legitimate user but with the replaced value of CA_U and H_U of the adversary. However, after receiving the message, server retrieves ID_U from CA_U and checks retrieved ID_U=received ID_U?. It results failed verification and communication will be terminated. Moreover, if the adversary $\tilde{A}$ only tries to modify the parameter HU it will not be possible as it is communicated by encrypting using ECDH based contributory symmetric key which is hard to forge in polynomial time. Hence, the ECC-CoAP scheme is robust against Man-in-the-Middle Attack.

4.1.2 Denial-of-Service (DoS) Attack

In the client response and challenge phase of ECC-CoAP scheme, if the IoT device fails to be authenticated by server within three attempts then the IoT device will be blocked for a specific period of time. A variable *count* is initialized with 0 and incremented by 1 after each of the unsuccessful response message transmission by the server. Every IoT device gets at most 3 attempts to be authenticated. Hence, an adversary $\tilde{A}$ will not be able to send multiple fuzzy requests (more than three) to make the system resource overloaded to make the services unavailable to the legitimate user, thus ECC-CoAP restricts the DoS attack.

4.1.3 Replay Attack

In the client response and challenge phase of the proposed ECC-CoAP scheme, if an adversary $\tilde{A}$ acquires the authentication message of the user $\{R_U, E_{SK_x}(M_U), T_3\}$ where $M_U = h(R_U||H_U||DID_U||T_3)$ and tries to replay it in later session just changing the current recorded time from T_3 to T_3' $\{R_U, E_{SK_x}(M_U), T_3'\}$. After receiving this authentication request by the server, it will check $||T4 - T3'| \leq \Delta T$, which would be successful. However, after checking the timestamp it will calculate $M_U' = h(R_U||H_U||DID_U||T_3')$ which will not be same as the received M_U. Hence, the session will be terminated. As in the proposed scheme, current timestamp is not only sent as a parameter of the message it also included as a parameter of M_U it is resilient to reply attack.

4.1.4 Insider Attack

Users provide their valid credentials to be authenticated to the remote server by assuming the remote server is trusted. However, sometimes it is noted that any insider of the remote server acts as an adversary $\tilde{A}$ after getting some crucial credentials of the user stored into the remote server. In proposed ECC-CoAP, the server stores H_U and DID_U as the crucial credentials for further authentication of IoT device. In this scenario, if H_U and DID_U are

acquired by the insider, still it cannot be authenticated as a legitimate user. For generating a valid authentication request, it is required to generate a random nonce say $R_U'=r_U'.Q_U$ and $M_U'=h(R_U'||H_U||DID_U||T_3)$.Then M_U' is encrypted using SK where SK is ECDH based session key calculated as $SK=q_U.r_U.R_S$ where q_U is the private key of the valid user. So, it is impossible for the insider to somehow calculate the session key SK due to hardness of ECC as well as it includes private key of the valid user. Hence ECC-CoAP is safe against insider attack.

4.1.5 User Impersonation Attack

If an adversary $\tilde{A}$ pretends to be an authorized user of the system. The adversary $\tilde{A}$ impersonates the transmitted message and re-transmits it pretending as a valid user. User impersonation attack cannot be possible in client side due to the following reasons:

(i) At the time of session initiation, user/IoT device sends the session initiation message $\{ID_U, CA_{U,}E_{K_X}(H_U), T_1\}$ to server. If the identity of the IoT device is modified then the server can easily track it from the ECC based public key certificate CA_U (containing identity ID_U and public key Q_U) as it is certified from the certificate authority and cannot be forged. .

Moreover, hash digest of the identity of the user H_U (containing identity ID_U, password PW_U and private key q_U) cannot be replaced by the adversary $\tilde{A}$ as it is transmitted in encrypted form by using the symmetric key K_X. However, K_X cannot be calculated due to hardness of ECDLP. So, H_U cannot be decrypted.

(ii) In client's response and challenge phase of ECC-CoAP, user/IoT device sends authentication request message containing $\{R_U, E_{SK_X}(M_U), T_3\}$.If the adversary $\tilde{A}$ intends to generate the masked identity of the user M_U it will not be able to compute it as it is encrypted using SK which is ECDH based session key where $SK=q_U r_U.\ r_S.\ q_S.P$ composed of private of the user q_U.

Hence, proposed ECC-CoAP is CoAP scheme is robust against user impersonation attack.

4.1.6 Server Impersonation Attack

In this type of attack, an adversary $\tilde{A}$ acts as a server by knowing some secret credentials of the server and further communicates with the user to exchange the messages. At first, the server sends a challenge message $ID_{S,}E_{K_X}(DID_U||R_S), T_2$ as a response of the session initiation request $\{ID_U, CA_U, E_{K_X}(H_U), T_1\}$ of the user/IoT device. However, to forge the server challenge to user the adversary $\tilde{A}$ needs to decrypt the value of H_U to compute valid $DID_U=h(ID_U||K||H_U)$using the symmetric key K_X. However, K is tough to compromise due to the hardness of ECDLP. So, the adversary $\tilde{A}$ cannot be able to determine the dynamic identity of IoT device valid DID_U. So, ECC-CoAP is safe against server impersonation attack.

4.1.7 Offline Password Guessing Attack

This is one of the most popular attacks that mainly occur at the password based authentication schemes due low entropy passwords chosen by the user. So, a strong password based scheme can restrict this type of attack. In ECC-CoAP, password PW_U is only used to calculate H_U where $H_U = h\ (ID_U\ ||PW_U||q_U)$ which stored for further communication. Hence, the adversary $\tilde{A}$ cannot be able to generate H_U only by randomly guessing the password the user/IoT device as H_U requires q_U, the private key of the user. Thus, ECC-CoAP is protected against offline password guessing attack.

4.1.8 Known Session Specific Temporary Attack

To avoid the occurrence of known session-specific temporary information attack, session key in ECC-CoAP is calculated in IoT device end as $SK = q_U.r_U.R_S = q_U.r_U.r_S.q_S.P$ and from server end as $SK = q_S.r_S.R_U = q_S.r_S.r_U.q_U.P = q_U.r_U.r_S.q_S.P$ which contains the private keys of each end. Although any one of the secret random values like r_s or $r_{U\ \text{of}}$ the server and user respectively are accidentally exposed to adversary $\tilde{A}$, still the session key cannot be generated due to the unavailability of the private keys. So, ECC-CoAP is free from known session specific temporary attack.

4.1.9 Session Key Computation Attack

ECC-CoAP is designed to agree upon a common secret session key $SK = q_U.r_U.R_S = q_S.r_S.R_U = q_U.r_U.r_S.q_S.P$ to carry out further data exchange securely between the user/IoT device and server. The proposed scheme provides ECDLP based secure session key which is hard to compromise due to hardness of ECDLP. Further, the session key cannot be computed it is generated based on two private keys and two random numbers both from user/IoT device and server end. If any of the secret parameters are somehow guessed or acquired in polynomial time, the other parameters are not available to the adversary $\tilde{A}$ for session key computation. Hence, ECC-CoAP is resilient to session key computation attack.

4.1.10 Efficient Mutual Authentication

ECC-CoAP provides a mutual authentication between the user/IoT devices and server based on two secret credentials M_U and M_S which are calculated based on secret values, mutually shared between them. During client authentication the server receives M_U encrypted using negotiated session key SK_X. M'_U is then calculated by the server using stored parameters DID_U and H_U as $M'_U = h(received\ R_U\ ||Stored\ H_U||Stored\ DID_U||T_3)$. If M'_U and M_U are equivalent then only the user/IoT device is authenticated. Similarly, during server authentication M'_S is calculated by the user/IoT device $M'_S = M'_S = h(received\ R_S||sent\ M_U)$. If M'_S and M_S are equivalent then only the server is authenticated. From the above discussion it is clear both server and client validate each other with the prior knowledge as well as received values. So, ECC-CoAP comprises of efficient mutual authentication.

4.1.11 Non-repudiation

Non-repudiation is a property which prevents a sender or entity from denying sending a message to the receiver. Use/IoT device sends the session initiation message containing $\{ID_U, CA_U, E_{K_X}(H_U), T_1\}$ to server. As the message contains the public key certificate of the message includes public key certificate containing valid identity of the user/IoT device it cannot deny about the sending of the message. On the other hand, in server challenge phase, server sends the reply composed of $\{ID_S, E_{K_X}(DID_U || R_S), T_2\}$ to user/IoT devices with server identity ID_S. So, in case also the server cannot deny the sending of message. So, ECC-CoAP comprises of non-repudiation.

4.1.12 Perfect Forward Secrecy

In the proposed ECC-CoAP the symmetric contributory key K is compromised the adversary $\tilde{A}$ cannot calculate the session key SK where $SK = q_U.r_U.R_S = q_S.r_S.R_U = q_U.r_U.r_S.q_S.P$ because with the knowledge of symmetric key K the adversary $\tilde{A}$ does not know the secret private key (q_U, q_S) or the random number of the particular session (r_U, r_S). Even if the adversary $\tilde{A}$can decrypt the message using the compromised symmetric key K to obtain random nonce R_U and R_S where $R_U = r_U.Q_U$ and $R_S = r_S.Q_S$, it cannot acquire the knowledge of session specific random numbers (r_U, r_S) due to the hardness of ECDLP. So, ECC-CoAP achieves the property of perfect forward secrecy.

4.2 Formal Security Analysis

In formal security analysis we have analyzed the security of ECC-CoAP protocol by using through Burrows–Abadi–Needham (BAN) logic [43, 44].For analyzing security related to key agreement and authentication protocol, BAN logic is most widely used mathematical model.

BAN Logic Based Authentication Proof

Table 4 Notations for BAN logic

Notations	Meanings
$X \mid\equiv Y$	The statement Y is believed by X
$X \triangleright Y$	X sees the statement Y
$X \mid\sim Y$	X once said the statement Y, sometimes ago
$X \Rightarrow Y$	X has got jurisdiction over Y
$\#Y$	The message Y is taken to be fresh
$\langle U \rangle V$	The formulae U is used in combination of formulae V
(U,V)	U or V being part of message (U,V)
$\{U,V\}K$	U or V is encrypted with symmetric key K
$\langle U, V \rangle_{k \mapsto X}$	U or V is encrypted with public key K of X
$(U,V)_K$	U or V is being hashed using key K
$X \overset{K}{\rightarrow} Z$	X and Z can securely contact using shared key K

Table 5 Primitive formulae used in BAN logic

Rules	Definitions
Message meaning rule	$\frac{X\|\equiv X\overset{K}{\longleftrightarrow}Z, X\lhd\langle Y\rangle_V}{X\|\equiv Z\|\sim Y}$
Nonce verification rule	$\frac{B\|\equiv\#(V), B\|\equiv D\|\sim V}{B\|\equiv D\|\equiv V}$
Jurisdiction rule	$\frac{B\|\equiv D\Rightarrow V, B\|\equiv D\|\equiv V}{B\|\equiv V}$
Freshness conjuncatenation rule	$\frac{B\|\equiv\#(V)}{B\|\equiv\#(V,T)}$
Belief rule	$\frac{B\|\equiv(V), B\|\equiv(T)}{B\|\equiv(V,T)}$
Session key rule	$\frac{B\|\equiv\#(V), B\|\equiv D\|\equiv V}{B\|\equiv B\overset{K}{\longleftrightarrow}D}$

The concerned following rules and notations of BAN logic are described in Tables 4 and 5 respectively, where X and Z are the general instances that participate in a protocol.

Following goals are required to be satisfied by aforesaid rules in order to prove the robustness of the ECC-CoAP under BAN logic.

Goals

Goal 1: $S| \equiv S \overset{SK}{\longleftrightarrow} C$
Goal 2: $S| \equiv C| \equiv_{SK} S \longleftrightarrow C$
Goal 3: $C| \equiv C \longleftrightarrow S$
Goal 4: $C| \equiv S| \equiv C \overset{SK}{\longleftrightarrow} S$

Idealized form of communicated messages

Message 1	$C \rightarrow S : ID_U, H_U, T_1 : \left\{\langle H_U\rangle_{(ID_U, PW_U, q_U)}\right\}_K$
Message 2	$S \rightarrow C : ID_S, DID_U, R_S, T_2 : \left\{\langle DID_U\rangle_{(ID_U, K, H_U)}\right\}_K$
Message 3	$C \rightarrow S : M_U, R_U, T_3 : \left\{\langle M_U\rangle_{(R_U, H_U)}, \langle R_U\rangle_{(r_U \cdot q_U \cdot q_S \cdot P)}\right\}_K$
Message 4	$S \rightarrow C : M_S, T_4 : \left\{\langle M_S\rangle_{(R_S, M_U)}\right\}_{SK}$

Following assumptions are required to authenticate the BAN logic for ECC-CoAP.

A1: $S|\equiv\#\, T_2, T_4$
A2: $C|\equiv\#\, T_1, T_3$
A3: $S|\equiv\#\, q_S$
A4: $S|\equiv\#\, r_S$
A5: $C|\equiv\#\, q_U$
A6: $C|\equiv\#\, r_U$
A7: $S|\equiv C\,\#\, Q_U, R_U$
A8: $C|\equiv S\,\#\, Q_S, R_S$
A9: $C|\equiv C \overset{K}{\leftrightarrow} S$
A10: $S|\equiv S \overset{K}{\leftrightarrow} C$

Proof of the Proposed Scheme using BAN Logic

Message 3

$$C \rightarrow S : M_U, R_U, DID_U, T_3 : \left\{ \langle M_U \rangle_{(R_U, H_U, DID_U)}, \langle R_U \rangle_{(r_U.q_U.P)}, \langle DID_U \rangle_{(ID_U, H_U, K)} \right\}_K$$

By applying seeing rule:

$$S1 : S \lhd \left\{ \langle M_U \rangle_{(R_U, H_U, DID_U)}, \langle R_U \rangle_{(r_U, q_U)} \right\}$$

By applying message meaning rule, S1, A10:

$$S2 : S| \equiv C \sim \left\{ \langle M_U \rangle_{(R_U, H_U, DID_U)}, \langle R_U \rangle_{(r_U.q_U.P)} \right\}$$

According to A5, A6, S2 and freshness rule

$$S3 : S| \equiv C| \equiv \#\left\{ \langle M_U \rangle_{(R_U, H_U, DID_U)}, \langle R_U \rangle_{(r_U.q_U.P)} \right\}$$

According to S3, S2 and nonce verification rule

$$S4 : S| \equiv C| \equiv \left\{ \langle M_U \rangle_{(R_U, H_U, DID_U)}, \langle R_U \rangle_{(r_U.q_U.P)} \right\}$$

According to A7, S4 and jurisdiction rule

$$S5 : S| \equiv \left\{ \langle M_U \rangle_{(R_U, H_U, DID_U)}, \langle R_U \rangle_{(r_U.q_U.P)} \right\}$$

As the session key is calculated as

$$SK = q_s.r_S.r_U.q_u.P$$

According to S5, S3 and session key rule

$$S6 : S| \equiv S| \equiv S \overset{K}{\leftrightarrow} C \qquad \text{(Goal 1)}$$

According to S6 and session key rule

$$S7 : S| \equiv C| \equiv S \equiv S \overset{K}{\leftrightarrow} C \qquad \text{(Goal 2)}$$

By applying seeing rule:

$$S8 : C \lhd R_S, M_S, T_4 \left\{ \langle R_S \rangle_{(rs, q_U, qs.P)}, \langle M_S \rangle_{(R_S, M_U)} \right\}_K$$

By applying message meaning rule, S8, A10:

$$S9 : C| \equiv S \sim R\left\{ {}_S, M_S, T_4 : \left\{ \langle M_S \rangle_{(R_S, M_U)}, \langle R_S \rangle_{(r_S.r_U.q_U.q_S.P)} \right\}_K \right\}$$

According to A3, A4,S9 and freshness rule

$$S10 : C| \equiv S| \equiv \#\left\{ R_S, M_S, T_4 : \left\{ \langle M_S \rangle_{(R_S, M_U)}, \langle R_S \rangle_{(r_S.r_U.q_U.q_S.P)} \right\}_K \right\}$$

According to S9, S10 and nonce verification rule

$$S11 : C| \equiv S| \equiv \left\{ R_S, M_S, T_4 : \left\{ \langle M_S \rangle_{(R_S, M_U)}, \langle R_S \rangle_{(r_S . r_U . q_U . q_S . P)} \right\}_K \right\}$$

According to A8, S11 and jurisdiction rule

$$S12 : C| \equiv \left\{ R_S, M_S, T_4 : \left\{ \langle M_S \rangle_{(R_S, M_U)}, \langle R_S \rangle_{(r_S . r_U . q_U . q_S . P)} \right\}_K \right\}$$

As the session key is calculated as

$$SK = q_s . r_S . r_U . q_u . P$$

According to S10, S12 and session key rule

$$S13 : C| \equiv C \overset{K}{\leftrightarrow} S \tag{Goal 3}$$

According to S13 and session key rule

$$S14 : C| \equiv S| \equiv C \overset{K}{\leftrightarrow} S. \tag{Goal 4}$$

5 Simulation for Formal Security Verification of ECC-CoAP Using AVISPA Tool

In this section, formal security verification by using the *Automated Verification Internet Security Protocol and Applications* (AVISPA) simulator tool is used for ECC-CoAP to ensure that ECC-CoAP is secure against all relevant attacks. AVISPA is a role-based simulator [45–49] that denotes that each participant plays a specific role and supports a language called *High Level Protocol Specification Language* (HLPSL).

5.1 Brief Discussion of the AVISPA Simulation Tool

The working procedure of AVISPA by using HLPSL is shown in the following Fig. 2. HLPSL specification is translated into an *Intermediate Format* (IF) by using a translator called HLPSL2IF, where IF is a lower level language compared to HLPSL [50]. It is used directly by the backends of the AVISPA tool to analyze whether the security goals are satisfied or violated. Based on this result, AVISPA tool produces the output either in SAFE or UNSAFE mode [48]. In current situation, AVISPA tool supports 4 (four) different types of the following back-ends [45, 46] that are (i) On-the-fly Model-Checker (OFMC) (ii) Constraint Logic based attack searcher (CL-At Se) (iii) State of the Art based Model checker (SATMC) (iv) Tree Automata based protocol for the security protocol analysis (TA4SP).

5.2 Brief Role Wise Specification of ECC-CoAP in AVISPA Simulation Tool

In this section, all the roles are developed in HLPSL language using AVISPA simulator to measure the scheme ECC-CoAP is secure or not. Both in Fig. 3, role U for user/IoT device and Fig. 4, role S for server are implemented in HLPSL. In Fig. 5, various roles regarding the session, goal and the environment in HLPSL language are presented. The current

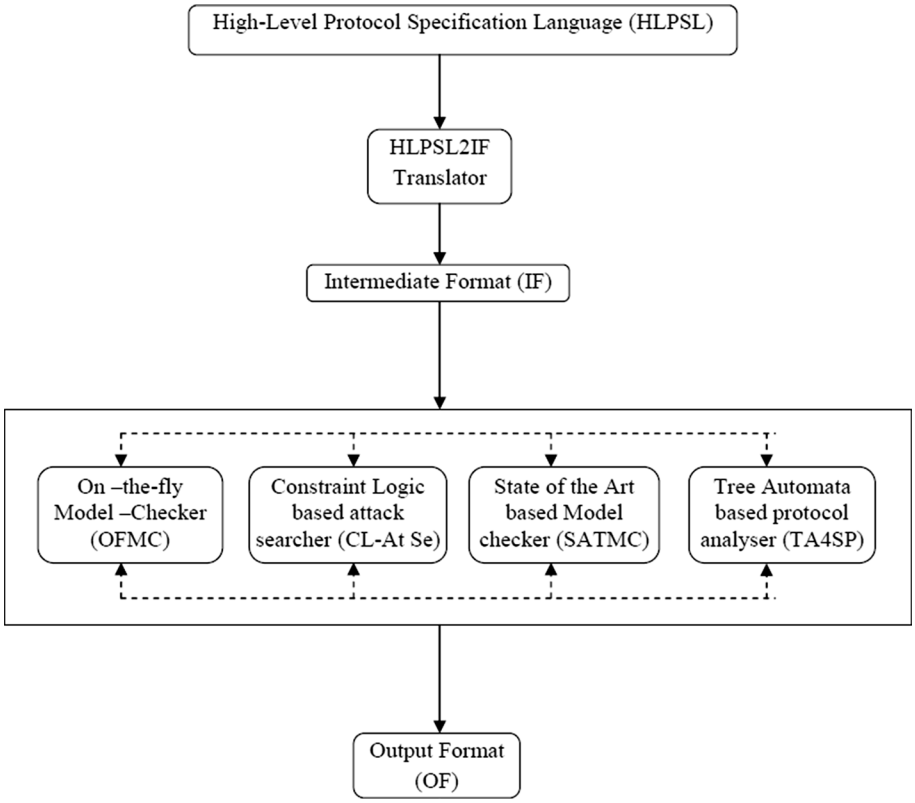

Fig. 2 Working structure of AVISPA using HLPSL

version (2006/02/2013) of HLPSL holds the standard authentication and secrecy goals. In ECC-COAP, the following five authentication procedures and nine secrecy goals are verified and shown in Table 5.

5.3 Simulation Result

In this section, the simulation results of ECC-CoAP for both the back-ends OFMC and CL-AtSe using AVISPA tool are showed in Figs. 6 and 7 respectively. From both the figures it is ensured that our proposed scheme is SAFE in two backend OFMC and CL-AtSe. Hence, ECC-CoAP is secured against all known active and passive attacks and achieves security goals (Table 6).

```
role role_U(U:agent,S:agent,K:symmetric_key,
            H:hash, Mul:function,P:text,
          SND,RCV:channel(dy))

played_by U def=

                local State:nat,
                UrU,MIDU,IDU,IDS,UqU:text,
                RU,RS,HU,HS,MU,MS,SK,DIDU,PWU,QU,QS,CAU,CAS,BU:message,
                Inc:function

                const

sec_1,sec_2,sec_3,sec_4,sec_5,sec_6,sec_7,sec_8,sec_9,sec_10,sec_11,sec_12,
                auth_1,auth_2,auth_3,auth_4,auth_5,auth_6:protocol_id

        init State:=0

       transition

              1. State=0 /\ RCV(start) =|>
                   State':=1 /\ UqU':= new()
                         /\ QU':= Mul(UqU'.P)
                             /\ secret(UqU',sec_2,{U})
                         /\ PWU':= new()
                         /\ IDU':= new()
                         /\ K':= Mul(UqU'.QS)
                             /\ HU':= H(IDU'.PWU'.UqU')
                             /\ CAU':= H(IDU'.QU')
                             %%/\ SND(IDU'.CAU')
                             /\ SND(IDU'.{CAU'.HU'}_K')
                             %%/\ secret(IDU',sec_3,{U,S})
                             /\ secret(QU',sec_3,{U,S})
                             /\ secret(PWU',sec_4,{U})
                             /\ secret(HU',sec_5,{U,S})
                             /\ secret(CAU',sec_6,{U,S})
                        %% U hopes that IDU will permit to authenticate him
                             /\ witness(U,S,auth_1,IDU')
                             /\ witness(U,S,auth_3,CAU')

                 2. State=1 /\ RCV({DIDU'.CAS'}_K')=|>
                State':=2 /\ K':= Mul(UqU.QS')
                             /\ DIDU':= H(IDU'.K'.HU')
                             /\ request(U,S,auth_2,DIDU')
                    %% U checks that he receives the same DIDU
                    %% that the server sent at step 1.
                         /\ UrU':= new()
                             /\ RU':= Mul(UrU'.K')
                             /\ MU':= H(HU'.RU')
                         /\ SND({RU'.DIDU'.MU'}_K')
                             /\ secret(UrU',sec_7,{U})
                             /\ secret(RU',sec_8,{U,S})
                             /\ secret(MU',sec_9,{U,S})
                             /\ witness(S,U,auth_3,MU')

              3. State=2 /\ RCV({RS'.MS'}_SK')=|>

                    %% U checks that he receives the same MS
                    %% that the server sent and based on server is authenticated
                   State':=3 /\ SK':= Mul(UrU'.RS')
                         /\ MS'= H(RS'.K')
                             /\ request(U,S,auth_4,MS')
                             %%/\ SK':= Mul(UrU'.RS')
                    %% Key negotiation is done.

end role
```

Fig. 3 Role specification for the user *U* in HLPSL

6 Performance Analysis of the Proposed Scheme

In this section, the overall performance of ECC-CoAP is discussed based on some primitive metrics like computation overhead, communication overhead, storage overhead and number of message communication. The evaluation is performed over a platform having an Intel Pentium Dual CPU E2200 2.20 GHz processor, 2048 MB of RAM and Ubuntu 17.04.1 LTS 32 bit operating system [51, 52]. To analyze performance of our scheme we

```
role role_S(S:agent,U:agent,K:symmetric_key,
            H:hash, Mul:function,P:text,
          SND,RCV:channel(dy))

played_by S def=

                local State:nat,
                SrS,MIDU,IDU,IDS,SqS,T1,T2,T3,DT:text,
                RS,RU,HU,MU,MS,HS,SK,DIDU,PWU,QS,QU,BU,CAS,CAU:message


       init State:=0

      transition

                  1.State=0 /\ RCV(IDU'.{CAU'.HU'}_K')=|>

              %% S receives IDU and QUfrom user certificate CAU
                    State':=1 /\ K':= Mul(SqS.QU')
                      /\ wrequest(S,U,auth_1,IDU')
                             %%/\ request(S,U,auth_3,CAU')
                             %%/\ SrS':=new()
                             %%/\ K':= Mul(SqS.QU')
                             %%/\ IDS':=new()
                             /\ DIDU':= H(IDU'.K'.HU')
                             /\ CAS':= H(IDS'.QS')
              %% S stores DIDU and HU for further communication
                             /\ SND({DIDU'.CAS'}_K')
                             %%/\ SND({DIDU'}_K)
                             /\ secret(DIDU',sec_8,{S,U})
                             /\ secret(QS',sec_10,{S,U})
                             /\ secret(CAS',sec_12,{S})
                        %% U hopes that DIDU will permit to authenticate
him
                             /\ witness(U,S,auth_2,DIDU')

            2. State=1 /\ RCV({RU'.DIDU'.MU'}_K')=|>

                        State':=2 /\ MU':= H(HU'.RU')
                             /\ request(S,U,auth_3,MU')
                           %% S hopes that  user will be authenticate
based on the values of MU
                             /\ SrS':=new()
                             /\ RS':= Mul(SrS'.K')
                             /\ MS':= H(RS'.K')
                             /\ SK':= Mul(SrS'.RU')
                             /\ SND({RS'.MS'}_SK')
                             /\ secret(SrS',sec_11,{S})
                          /\ witness(S,U,auth_4,MS')

                    end role
```

Fig. 4 Role specification for the server S in HLPSL

have compared the proposed scheme with the recently proposed Dey and Hossain scheme [27] discussed in literature review. The below subsections illustrates the performance analysis of our scheme in terms of aforementioned parameters separately.

```
role
session(U:agent,S:agent,M,SK,QS,QU,CAU,CAS:message,H:hash,Mul:function,P:te
xt,K:symmetric_key)
def=
local SND2,RCV2,SND1,RCV1:channel(dy)
      composition
            %%role_U(U,S,K,SK,QS,SND1,RCV1) /\
role_S(S,U,K,SK,QS,SND2,RCV2)
                  role_U(U,S,K,H,Mul,P,SND1,RCV1) /\
role_S(S,U,K,H,Mul,P,SND2,RCV2)
end role

role environment()
def=
      const
                u,s:agent,
                k:symmetric_key,
                srs,uru,idu,ids,sqs,uqu,p:text,
                rs,ru,sk,didu,pwu,qs,qu,m,cau,cas:message,
                h:hash,mul:function,

sec_1,sec_2,sec_3,sec_4,sec_5,sec_6,sec_7,sec_8,sec_9,sec_10,sec_11,sec_12,
                auth_1,auth_2,auth_3,auth_4,auth_5,auth_6:protocol_id

      intruder_knowledge = {u,s,idu,ids,m,p,qs,qu,cau,cas}

      composition
            session(u,s,m,sk,qs,qu,cau,cas,h,mul,p,k) /\
session(s,u,m,sk,qs,qu,cau,cas,h,mul,p,k)
end role
goal
      secrecy_of sec_1
        secrecy_of sec_2
        secrecy_of sec_3
        secrecy_of sec_4
        secrecy_of sec_5
        secrecy_of sec_6
        secrecy_of sec_7
        secrecy_of sec_8
        secrecy_of sec_9
        secrecy_of sec_10
        secrecy_of sec_11
        secrecy_of sec_12
        authentication_on auth_1
        authentication_on auth_2
        authentication_on auth_3
        authentication_on auth_4
        authentication_on auth_5
        authentication_on auth_6

end goal
environment()
```

Fig. 5 Role specification for the session and the environment in HLPSL

6.1 Computation Overhead

In the proposed scheme, ECC is incorporated for secure communication between low powered constraint IoT devices and remote server. The execution time for different cryptographic operations and computational overhead for the ECC-CoAP are discussed in Tables 7 and 8 respectively. Here, T_h is the time for cryptographic hash operation, $T_{E/D(S)}$ is the time for encryption/decryption with symmetric key, T_{ECPM} is the time required for elliptic curve point multiplication, T_{ECPA} is the time required for elliptic curve point

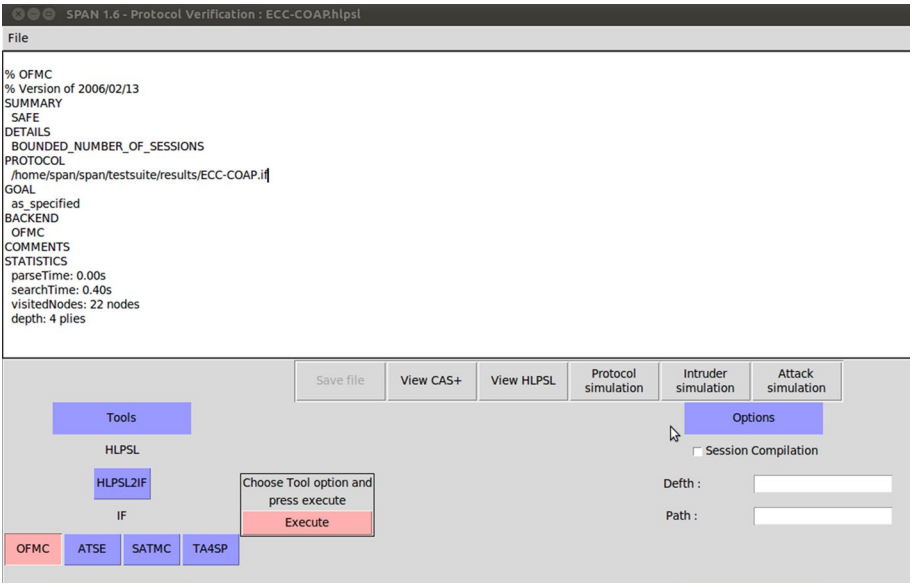

Fig. 6 Screen shot of AVISPA Simulation for the result of OFMC back end

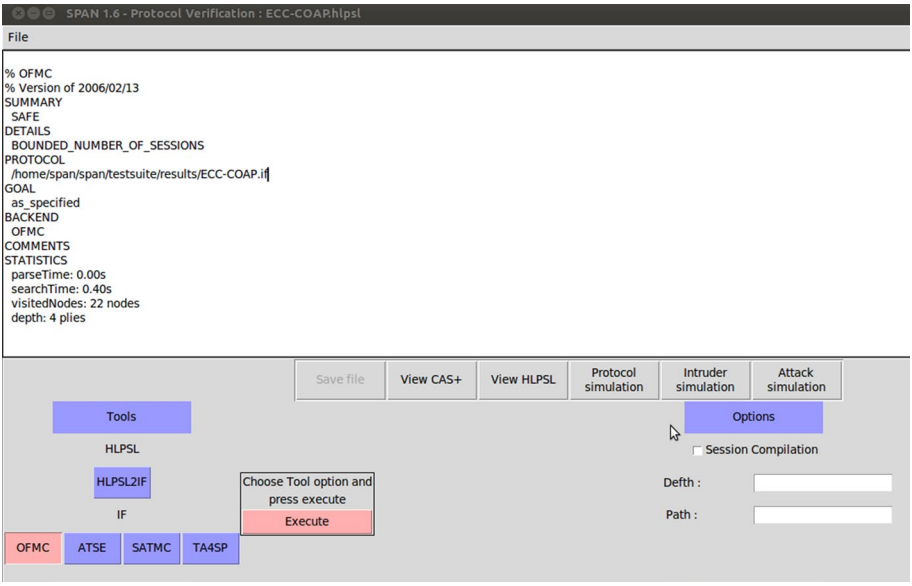

Fig. 7 Screen shot of AVISPA Simulation for the result of Cl–At Se back end

Table 6 Security goals of AVISPA

Sl. no	Secret/authentication type	Description
1	secrecy_of sec_1	sec_1, represents private key of user/IoT device and kept secret with U
2	secrecy_of sec_2	sec_2, represents user public key and kept secret between U and S
3	secrecy_of sec_3	sec_3, represents user password and kept secret with U
4	secrecy_of sec_4	sec_4, represents Hash random value of user and kept secret between U and S
5	secrecy_of sec_5	sec_5, represents, user selects a random number and kept secret with U
6	secrecy_of sec_6	sec_6, user selects a random number and kept secret between U and S
7	secrecy_of sec_7	sec_7,represents dynamic identity and kept secret between U and S
9	secrecy_of sec_8	sec_8,represents Message random value of user and remains between U and S
10	secrecy_of sec_9	sec_9,represents Public key of server and kept secret between U and S
11	secrecy_of sec_10	sec_10, represents random number selected by server and kept secret with S
12	authentication_on auth_1	represents IDU and is used authentication purpose between the U and S
13	authentication_on auth_2	represents $DIDU$ and is used for authentication purpose between the U and S
14	authentication_on auth_3	represents CAU and is used for authentication purpose between the U and S
15	authentication_on auth_4	represents $MS' = h(RS'.MU')$ for authentication purpose between the U and S
16	authentication_on auth_5	represents $MU = h(HU'.RU')$ for authentication purpose between the U and S

Table 7 Execution time in milliseconds (ms) for different cryptographic operation [52]

Notation	Description and execution time (ms)
T_{ECPM}	Time complexity for the execution of the elliptic curve (ECC) point multiplication Where $T_{ECPM} = 2.226$ ms
$T_{E/D(S)}$	Time complexity for the execution of the encryption/decryption with symmetric key Where $T_{E/D(S)} = 3.85$ ms
T_h	Time complexity for the execution of the hash function where $T_h = 0.0046$ ms
T_{ECPA}	Time complexity for the execution of the elliptic curve (ECC) point addition where $T_{ECPA} = 0.0288$ ms
T_{EMod}	Time complexity for the execution of the for modular exponential operation where $T_{EMod} = 3.85$ ms

Table 8 Comparison of computational overheads

Schemes	Overheads	Time (ms)
Proposed ECC-CoAP	$5T_h + 6T_{ECPM} + 4T_{E/D(S)}$	28.779
Dey and Hossain scheme [27]	$6T_h + 4T_{EMod} + 6T_{E/D(S)}$	38.5276

Table 9 Comparison of communication overheads

Schemes	Communication (bits)	Number of message communication
Proposed ECC-CoAP	1024	4
Dey and Hossain scheme [27]	1312	5

Table 10 Comparison of security strength

Security feature	[17]	[27]	[54]	Proposed ECC-CoAP
Man-in-the-middle attack	Yes	Yes	Yes	Yes
Denial-of-service (DoS) attack	No	No	No	Yes
Replay attack	Yes	Yes	Yes	Yes
Insider attack	Yes	Yes	Yes	Yes
User impersonation attack	Yes	Yes	Yes	Yes
Server impersonation attack	Yes	No	No	Yes
Offline password guessing attack	No	No	No	Yes
Known session specific temporary attack	Yes	Yes	Yes	Yes
Session key computation attack	Yes	Yes	Yes	Yes
Efficient mutual authentication	Yes	Yes	Yes	Yes
Non-repudiation	Yes	Yes	Yes	Yes
Perfect forward secrecy	Yes	Yes	Yes	Yes

addition and T_{EMod} is the time required for modular exponential operation. The computational overhead of our scheme is calculated in terms of execution time using Table 7 as *28.779* ms which is considerably less than Dey and Hossain scheme [27] due to the use of less expensive cryptographic operations.

6.2 Communication Overhead

The communication overhead between participating devices depends on the number of messages communicated as well as total number of bits transmitted during conversation as the network congestion also depends on the number of messages exchanged between two devices. The total number of messages communicated during the conversation in our protocol is 4 messages whereas for the LESS protocol by Bhattchariya et al. [17] and Dey and Hossain schemes [27] require 4 messages and 5 messages respectively. Thus, our protocol is efficient in terms of communication overhead than the schemes. In ECC-CoAP, identity of the communicating parties (ID_U, ID_S)is taken 64 bits long, time stamps(T_1, T_2, T_3,T_4)are taken as 32 bits long, random values generated (R_U) is taken as 128 bits and encryption done by symmetric keys (K,SK) are taken as 160 bits long [50–53]. After calculation the message total number of transmitted bits is 1024 bits which is less than Dey and Hossain scheme [27]. The communication overheads are shown in Table 9.

6.3 Security Strength

As stated in Sects. 4.1 and 4.2, ECC-CoAP is well protected against several relevant attacks. However, comparative security strength of the proposed scheme with other related schemes [17, 27, 54] is depicted in Table 10. It is found that ECC-CoAP is very much secured than other related schemes.

7 Conclusion

A flexible ECC based CoAP for communication between the user/IoT device and server for setting up secure session among constraints IoT devices is proposed. The proposed scheme will be used to solve the key management and related security issues of resource constraint IoT devices as well as securely operated in insecure channel. The proposed scheme is mathematically analyzed to show its strong resilience against relevant cryptographic attacks. Moreover, ECC-CoAP is formally verified using well accepted AVISPA simulator and BAN logic and found well secure. Finally, the performance study demonstrates that our scheme is more effective in terms of communication and computation overheads for resource constrained IoT devices. Thus ECC-CoAP becomes cost-effective solution for highly demanded client side IoT based CoAP applications.

Acknowledgement Authors are immensely grateful to the Editor-in-Chief and anonymous reviewers for their precious comments and beneficial suggestions. The research work is an outcome of the R&D project sanctioned to Dr. Sangram Ray under the Seed Grant funded by TEQIP III, NPIU, Ministry of Education, Government of India. Muhammad Khurram Khan is supported by Researchers Supporting Project number (RSP-2020/12), King Saud University, Riyadh, Saudi Arabia.

References

1. Vasseur, J. P., & Dunkels, A. (2010). *Interconnecting smart objects with IP: The next internet*. Burlington, MA: Morgan Kaufmann.
2. Mikami, S., Watanabe, D., Li, Y., & Sakiyama, K. (2015). Fully integrated passive UHF RFID tag for hash-based mutual authentication protocol. *The Scientific World Journal.* https://doi.org/10.1155/2015/498610.
3. Lopez, J., & Rubio, J. E. (2018). Access control for cyber-physical systems interconnected to the cloud. *Computer Networks, 134,* 46–54.
4. Keoh, S. L., Kumar, S. S., & Tschofenig, H. (2014). Securing the Internet of Things: A standardization perspective. *IEEE Internet of Things Journal, 1*(3), 265–275.
5. Capossele, A., Cervo, V., De Cicco, G., & Petrioli, C. (2015, June). Security as a CoAP resource: An optimized DTLS implementation for the IoT. In *2015 IEEE international conference on communications (ICC)* (pp. 549–554). IEEE.
6. Rahman, R. A., & Shah, B. (2016, March). Security analysis of IoT protocols: A focus in CoAP. In *2016 3rd MEC international conference on big data and smart city (ICBDSC)* (pp. 1–7). IEEE.
7. Nguyen, H. V., & Iacono, L. L. (2015, September). REST-ful CoAP message authentication. In *2015 international workshop on secure Internet of Things (SIoT)* (pp. 35–43). IEEE.
8. Brachmann, M., Garcia-Morchon, O., & Kirsche, M. (2011). *Security for practical CoAP applications: Issues and solution approaches*. Stuttgart: GI/ITG KuVS Fachgesprch Sensornetze (FGSN). Universitt Stuttgart.

9. Yassein, M. B., Shatnawi, M. Q., Aljwarneh, S., & Al-Hatmi, R. (2017, May). Internet of Things: Survey and open issues of MQTT protocol. In *2017 international conference on engineering & MIS (ICEMIS)* (pp. 1–6). IEEE.
10. Alliance, O. M. (2002). Generic content download over the air specification. v1. 0 December.
11. Palattella, M. R., Accettura, N., Vilajosana, X., Watteyne, T., Grieco, L. A., Boggia, G., et al. (2012). Standardized protocol stack for the internet of (important) things. *IEEE Communications Surveys & Tutorials, 15*(3), 1389–1406.
12. Alghamdi, T. A., Lasebae, A., & Aiash, M. (2013, November). Security analysis of the constrained application protocol in the Internet of Things. In *Second international conference on future generation communication technologies (FGCT 2013)* (pp. 163–168). IEEE.
13. Villaverde, B. C., Pesch, D., Alberola, R. D. P., Fedor, S., & Boubekeur, M. (2012, July). Constrained application protocol for low power embedded networks: A survey. In *2012 sixth international conference on innovative mobile and internet services in ubiquitous computing* (pp. 702–707). IEEE.
14. Moritz, G., Golatowski, F., & Timmermann, D. (2011, October). A lightweight SOAP over CoAP transport binding for resource constraint networks. In *2011 IEEE eighth international conference on mobile ad-hoc and sensor systems* (pp. 861–866). IEEE.
15. Schneider, J., Kamiya, T., Peintner, D., & Kyusakov, R. (2011). Efficient XML interchange (EXI) format 1.0. *W3C Proposed Recommendation, 20,* 32.
16. Khalique, A., Singh, K., & Sood, S. (2010). Implementation of elliptic curve digital signature algorithm. *International Journal of Computer Applications, 2*(2), 21–27.
17. Bhattacharyya, A., Bose, T., Bandyopadhyay, S., Ukil, A., & Pal, A. (2015, March). LESS: Lightweight establishment of secure session: A cross-layer approach using CoAP and DTLS-PSK channel encryption. In *2015 IEEE 29th international conference on advanced information networking and applications workshops* (pp. 682–687). IEEE.
18. Granjal, J., Monteiro, E., & Silva, J. S. (2015). Security for the Internet of Things: A survey of existing protocols and open research issues. *IEEE Communications Surveys & Tutorials, 17*(3), 1294–1312.
19. Ray, S., Biswas, G. P., & Dasgupta, M. (2016). Secure multi-purpose mobile-banking using elliptic curve cryptography. *Wireless Personal Communications, 90*(3), 1331–1354.
20. Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security, 1*(1), 36–63.
21. Levi, A., & Savas, E. (2003, July). Performance evaluation of public-key cryptosystem operations in WTLS protocol. In *Proceedings of the eighth IEEE symposium on computers and communications. ISCC 2003* (pp. 1245–1250). IEEE.
22. Raza, S., Helgason, T., Papadimitratos, P., & Voigt, T. (2017). SecureSense: End-to-end secure communication architecture for the cloud-connected Internet of Things. *Future Generation Computer Systems, 77,* 40–51.
23. Iglesias-Urkia, M., Orive, A., & Urbieta, A. (2017, January). Analysis of CoAP implementations for industrial Internet of Things: A survey. In *ANT/SEIT* (pp. 188–195).
24. Alaba, F. A., Othman, M., Hashem, I. A. T., & Alotaibi, F. (2017). Internet of Things security: A survey. *Journal of Network and Computer Applications, 88,* 10–28.
25. Albalas, F., Al-Soud, M., Almomani, O., & Almomani, A. (2018). Security-aware CoAP application layer protocol for the Internet of Things using elliptic-curve cryptography. *Power (mw), 1333,* 151.
26. Harish, M., Karthick, R., Rajan, R. M., & Vetriselvi, V. (2018). Securing CoAP through payload encryption: Using elliptic curve cryptography. *International Conference on Communications and Cyber Physical Engineering, 2018,* 497–511.
27. Dey, S., & Hossain, A. (2019). Session-key establishment and authentication in a smart home network using public key cryptography. *IEEE Sensors Letters, 3*(4), 1–4.
28. Yeh, H. L., Chen, T. H., Liu, P. C., Kim, T. H., & Wei, H. W. (2011). A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors, 11*(5), 4767–4779.
29. Koblitz, N. (Ed.). (2000). *Towards a quarter-century of public key cryptography*. New York, NY: Kluwer Academic.
30. Miller, V. S. (1985, August). Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques* (pp. 417–426). Berlin: Springer.
31. Paar, C., & Pelzl, J. (2009). *Understanding cryptography: A textbook for students and practitioners*. Berlin: Springer.
32. Islam, S. H., Amin, R., Biswas, G. P., Farash, M. S., Li, X., & Kumari, S. (2017). An improved three party authenticated key exchange protocol using hash function and elliptic curve cryptography for mobile-commerce environments. *Journal of King Saud University-Computer and Information Sciences, 29*(3), 311–324.

33. Ray, S., & Biswas, G. P. (2011, December). Design of mobile-PKI for using mobile phones in various applications. In *2011 international conference on recent trends in information systems* (pp. 297–302). IEEE.
34. Ray, S., & Biswas, G. P. (2012, October). An ECC based public key infrastructure usable for mobile applications. In *Proceedings of the second international conference on computational science, engineering and information technology* (pp. 562–568).
35. Sadhukhan, D., Ray, S., Biswas, G. P., Khan, M. K., & Dasgupta, M. (2020). A lightweight remote user authentication scheme for IoT communication using elliptic curve cryptography. *Journal of Supercomputing*. https://doi.org/10.1007/s11227-020-03318-7.
36. Tribedi, D., Sadhukhan, D., & Ray, S. (2018, July). Cryptanalysis of a secure and privacy preserving mobile wallet scheme with outsourced verification in cloud computing. In *International conference on computational intelligence, communications, and business analytics* (pp. 411–424). Singapore: Springer.
37. Sadhukhan, D., & Ray, S. (2018, March). Cryptanalysis of an elliptic curve cryptography based lightweight authentication scheme for smart grid communication. In *2018 4th international conference on recent advances in information technology (RAIT)* (pp. 1–6). IEEE.
38. Turkanović, M., Brumen, B., & Hölbl, M. (2014). A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Networks, 20,* 96–112.
39. Wang, D., Li, W., & Wang, P. (2018). Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics, 14*(9), 4081–4092.
40. Chatterjee, U., Sadhukhan, D., & Ray, S. (2020). An improved authentication and key agreement protocol for smart healthcare system in the context of internet of things using elliptic curve cryptography. In *Proceedings of international conference on IoT inclusive life (ICIIL 2019), NITTTR Chandigarh, India* (pp. 11–22). Singapore: Springer.
41. Das, A. K., Sharma, P., Chatterjee, S., & Sing, J. K. (2012). A dynamic password-based user authentication scheme for hierarchical wireless sensor networks. *Journal of Network and Computer Applications, 35*(5), 1646–1656.
42. Mishra, D., Das, A. K., & Mukhopadhyay, S. (2014). A secure user anonymity-preserving biometric-based multi-server authenticated key agreement scheme using smart cards. *Expert Systems with Applications, 41*(18), 8129–8143.
43. Burrows, M., Abadi, M., & Needham, R. M. (1989). A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 426*(1871), 233–271.
44. Mahmood, K., Chaudhry, S. A., Naqvi, H., Kumari, S., Li, X., & Sangaiah, A. K. (2018). An elliptic curve cryptography based lightweight authentication scheme for smart grid communication. *Future Generation Computer Systems, 81,* 557–565.
45. Adhikari, S., Ray, S., Obaidat, M. S., & Biswas, G. P. (2020). Efficient and secure content dissemination architecture for content centric network using ECC-based public key infrastructure. *Computer Communications, 157,* 187–203.
46. Challa, S., Wazid, M., Das, A. K., Kumar, N., Reddy, A. G., Yoon, E. J., et al. (2017). Secure signature-based authenticated key establishment scheme for future IoT applications. *IEEE Access, 5,* 3028–3043.
47. Amin, R., & Biswas, G. P. (2016). A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. *Ad Hoc Networks, 36,* 58–80.
48. Ali, R., Pal, A. K., Kumari, S., Karuppiah, M., & Conti, M. (2018). A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring. *Future Generation Computer Systems, 84,* 200–215.
49. Adhikari, S., Ray, S., Biswas, G. P., & Obaidat, M. S. (2019). Efficient and secure business model for content centric network using elliptic curve cryptography. *International Journal of Communication Systems, 32*(1), e3839.
50. Kumari, S., & Om, H. (2016). Authentication protocol for wireless sensor networks applications like safety monitoring in coal mines. *Computer Networks, 104,* 137–154.
51. Schneier, B. (2007). *Applied cryptography: Protocols, algorithms, and source code in C*. Hoboken, NJ: Wiley.
52. Kilinc, H. H., & Yanik, T. (2013). A survey of SIP authentication and key agreement schemes. *IEEE Communications Surveys & Tutorials, 16*(2), 1005–1023.
53. Vermesan, O., Friess, P., Guillemin, P., Sundmaeker, H., Eisenhauer, M., Moessner, K., et al. (2013). *Internet of Things strategic research and innovation agenda* (p. 7). Brighton: River Publishers Series in Communications.

54. Kumar, P., Gurtov, A., Iinatti, J., Ylianttila, M., & Sain, M. (2015). Lightweight and secure session-key establishment scheme in smart home environments. *IEEE Sensors Journal, 16*(1), 254–264.

Mr. Suman Majumder is a Ph.D. Scholar in the Department of Computer Science and Engineering, National Institute of Technology Sikkim, India under the Ministry of Human Resource Department, Govt. of India. He has obtained B.Tech. degree in Computer Science Engineering from West Bengal University of Technology (presently known as Maulana Abul Kalam Azad University of Technology) in 2006 and M.Tech. degree in Information Technology from Jadavpur University, in 2011 respectively. His research area includes Cryptography and Information Security, Elliptic Curve Cryptography, Internet of Things, Network Security etc.

Dr. Sangram Ray is an Assistant Professor in the Department of Computer Science and Engineering, National Institute of Technology Sikkim, India. He has obtained B.Sc. (Mathematics Honours, 2005) from University of Burdwan and M.Sc. (Mathematics and Computing, 2007), M.Tech. (Computer Application, 2009) and Ph.D. (Computer Science and Engineering, 2014) from Indian Institute of Technology (Indian School of Mines), Dhanbad respectively. His research area includes Cryptography and Information Security, Elliptic Curve Cryptography, Content Centric Network, Internet-of-Things etc.

Ms. Dipanwita Sadhukhan is an Institute PhD Scholar in the Department of Computer Science and Engineering, National Institute of Technology Sikkim, India under the Ministry of Human Resource Department, Govt. of India. She has obtained B.Tech. degree in Information Technology and M.Tech. degree in Computer Science and Engineering from West Bengal University of Technology, presently known as Maulana Abul Kalam Azad University of Technology in 2010 and 2013 respectively. Her research area includes Cryptography and Information Security, Elliptic Curve Cryptography, Internet of Things, Network Security etc.

Dr. Muhammad Khurram Khan is currently working as a Professor of Cybersecurity at the Center of Excellence in Information Assurance, King Saud University, Kingdom of Saudi Arabia. He is founder and CEO of the 'Global Foundation for Cyber Studies and Research', an independent, non-profit, and non-partisan cybersecurity think-tank in Washington D.C. USA. He is the Editor-in-Chief of Telecommunication Systems Journal published by Springer-Nature with 1.73 impact factor. He is on the editorial board of several journals published by IEEE, Elsevier, Springer, and Wiley. He has published over 375 publications and an inventor of 10 patents. He has secured several national and international awards and grants in the area of Cybersecurity. He is a distinguished lecturer of the IEEE. He is a fellow of the IET (UK) and a fellow of the BCS (UK). His detailed profile can be visited at http://www.professorkhurram.com.

Dr. Mou Dasgupta is an Assistant Professor in the Department of Computer Application, National Institute of Technology Raipur, India. She has obtained B.Sc. (Economics honours, 2004) from Calcutta University, Master of Computer Application (2007) from West Bengal University of Technology and Ph.D. (Computer Science, 2012) from Indian Institute of Technology (Indian School of Mines), Dhanbad respectively.

Affiliations

Suman Majumder[1] · Sangram Ray[1] · Dipanwita Sadhukhan[1] · Muhammad Khurram Khan[2] · Mou Dasgupta[3]

Suman Majumder
suman.majumder3014@gmail.com

Dipanwita Sadhukhan
dipanwitasadhukhan2012@gmail.com

Muhammad Khurram Khan
mkhurram@ksu.edu.sa

Mou Dasgupta
elle.est.mou@gmail.com

[1] Department of Computer Science and Engineering, National Institute of Technology Sikkim, Ravangla, Sikkim 737139, India

[2] Center of Excellence in Information Assurance, College of Computer & Information Sciences, King Saud University, Riyadh 11653, Kingdom of Saudi Arabia

[3] Department of Computer Application, National Institute of Technology Raipur, Raipur 492010, India