Arabian Journal for Science and Engineering (2021) 46:3865–3888
https://doi.org/10.1007/s13369-020-05276-x

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

A Secure User Authentication Protocol Based on ECC for Cloud Computing Environment

Diksha Rangwani[1] · Hari Om[1]

Received: 13 June 2020 / Accepted: 21 December 2020 / Published online: 29 January 2021

Abstract
Cloud computing relies on on-demand sharing of the computing resources and data without the user's direct involvement in resource management over the network, but it has major security threats. Recently, an it is Elliptic Curve Cryptography (ECC) based three-factor authentication and key negotiation protocol for fog computing has been discussed by Wazid et al. In this paper, we show that the Wazid et al.'s protocol requires high communication as well as storage cost, and also, it is susceptible to the denial-of-service attack, stolen smart card attack, and privileged insider attack. We further propose a new protocol that overcomes these problems. We carry out informal and formal security analysis and also simulate it using the it is Automated Validation of Internet Security Protocols and Applications tool (AVISPA) to prove its robustness against the security threats. Its performance analysis illustrates that it is efficient and lightweight in comparison with the existing schemes.

Keywords Authentication · Cloud computing · ECC · Privileged insider attack

1 Introduction

Cloud computing may be understood as simply providing the services and resources over the network. It has gained good popularity due to its efficiency, mobility, on-demand service model, broad network access, high computing capability, and scalability. Also the cloud computing is a blend of service-oriented and event-driven architecture. The cloud architecture generally consists of loosely coupled components, which may broadly be categorized as front end and back end. The front end is the client side, i.e., client infrastructure and the back end comprises of application, service, run-time cloud, storage, infrastructure, management, and security. The Internet works as the medium of contact between these two ends. The architecture of cloud computing is depicted in Fig. 1.

Li et al. [1] discuss an identity-based mutual authentication protocol that has been built on the hierarchical cloud architecture, claiming it to be lightweight and proficient than the secure socket layer (SSL) and professing its use in scalable environment. Sun et al. [2] show that the protocol [1] is vulnerable to the privileged insider attack, lack of user anonymity, lack of proper mutual authentication, session key leakage, and lack of perfect forward secrecy, and they discuss a scheme by overcoming these problems. Li et al. [3] report that the scheme [4] suffers from the stolen smart card attack and lacks user anonymity, and they discuss a key negotiation and authentication protocol by overcoming these flaws. Wazid et al. [5] analyze the scheme [3] and find that it is susceptible to the stolen smart card attack, user impersonation attack, replay attack, off-line password guessing attack and man-in-the-middle attack, and discuss a protocol for multi-server environment in cloud computing applications. Hu et al. [6] review the fog computing basic fundamentals, applications, model architecture, and related issues. Alrawais et al. [7] discuss an attribute-based key negotiation scheme for fog and cloud environment. Mukherjee et al. [8] discuss the security threats, vulnerabilities, research trends and their challenges in privacy preservation for fog computing environment. Koo et al. [9] discuss a method for data deduplication of multi-party owned data in fog computing and claim that their method incurs less communication overheads. Wang et al. [10] discuss a data aggregation scheme for fog-based cloud environment using the homomorphic encryption and pseudonyms to assure data secrecy and device

✉ Diksha Rangwani
diksharangwani@gmail.com

Hari Om
hariom4india@gmail.com

1 Indian Institute of Technology (ISM) Dhanbad, Dhanbad, India

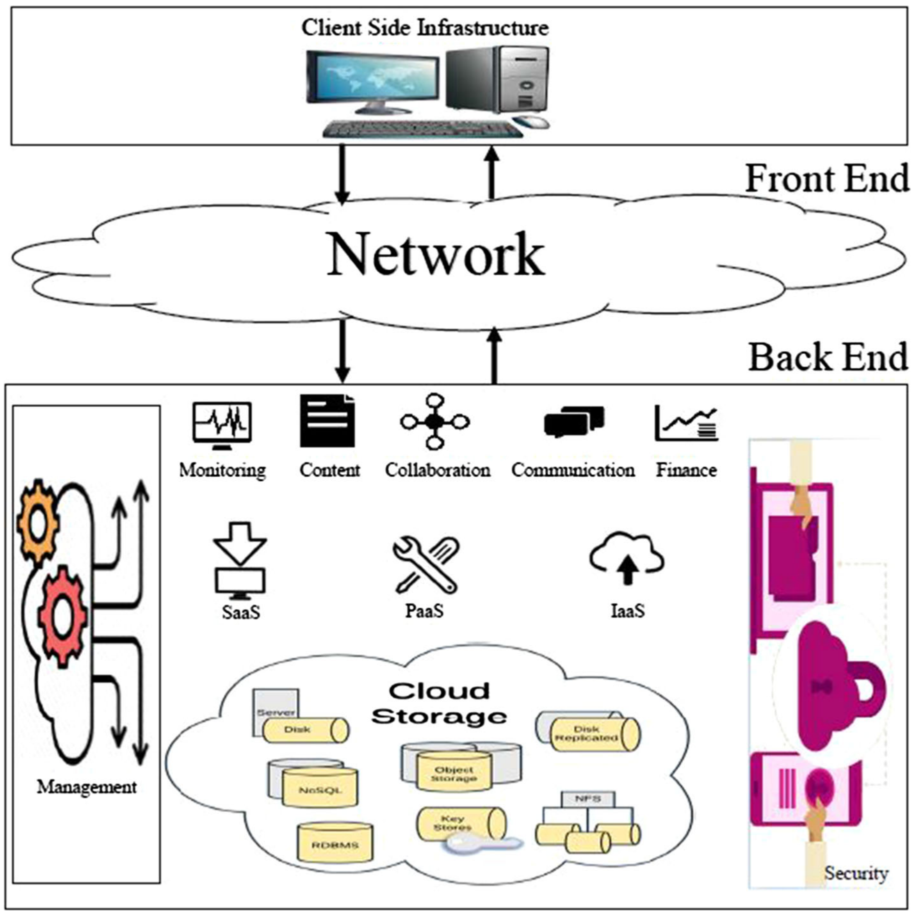


Fig. 1 Cloud Computing Architecture

anonymity. Wazid et al. [11] discuss an authentication protocol applying ECC to resolve the weakness of the protocols [2,3,6]. In this paper, we scrutinize the protocol [11] and find that it is susceptible to the denial-of-service attack, privileged insider attack, and stolen smart card attack, and we develop a protocol by overcoming these vulnerabilities. The novelties of our proposed protocol are summarized below:

1. We analyze the protocol [11] and show that it suffers from attacks like denial-of-service, stolen smart card and privileged insider.
2. Our protocol incorporates the lightweight cryptographic hash function and overcomes the flaws of the protocol [11], while maintaining less communication as well as computation overheads.
3. Its formal security analysis is carried out by using the Burrows–Abadi–Needham (BAN) logic [12] to prove its legitimacy.
4. Its informal security analysis is done to show its robustness to all the fore-known security attacks.
5. Its simulation is carried out using the AVISPA simulation tool [13] to show its resilience against various active and passive attacks.
6. Its performance is compared with the existing schemes to show its efficiency.

The remaining parts of the paper are arranged as mentioned. Section 2 introduces preliminaries, and Sect. 3 reviews the Wazid et al.'s protocol [11], followed by its cryptanalysis in Sect. 4. Section 5 introduces the proposed protocol, and its security analysis is provided in Sect. 6. Its performance analysis is done in Sect. 7, and finally, the paper is concluded in Sect. 8.

2 Preliminaries

Here, we introduce basics of ECC [14] and some discrete problems that will be used in our proposed protocol.

2.1 Elliptic Curve

The non-singular elliptic curve, denoted by E/F_q (E being the elliptic curve, q a prime) over a prime finite field represented by F_q, is given as: $y^2 = (x^3 + ax + b)\ mod\ q, a, b \in \mathbb{Z}_q^*$. The fundamental group operations of ECC are defined as follows:

1. *Elliptic Curve point addition* For two points A, B on an elliptic curve, their sum $A+B = C$, where the line joining A and B cuts the curve at -C, which is reflection of C with respect to x-axis [14].
2. *Elliptic Curve point of subtraction* Let A, B be two points on an elliptic curve such that $A = -B$, i.e., $A+B = A+(-A) = 0$. The points A and B along with the intersecting line of the curve join it at abstract point 0, called the point of infinity [14].
3. *Elliptic Curve point doubling* Adding a point A to itself on an elliptic curve gives a new point B on the curve such that $B = 2A$, which is the reflection of the intersection point with the tangent drawn at A with respect to x-axis [14].
4. *Elliptic Curve scalar point multiplication* For a point A on an elliptic curve such that $p.A = A + A + ... + A(p\,times)= \sum_1^p A$, where $p \in \mathbb{Z}_q^*$ is a scalar [14].
5. *Order of point* The order of an element (point) A in G_q is defined as n, where $n > 0$ is an integer such that $n.P = 0$ (point of infinity) [14].

The robustness of ECC depends on the non-existence of polynomial time solution to the computational problems of ECC. The computational problems are defined below.

2.2 Computational Hardness

The computational hardness of ECC is briefly discussed below:

1. *Discrete Logarithm Problem (DLP)* It is hard to find x from B such that $B = xP$ in polynomial time. The points P and B lie on the elliptic curve $E/F_q, \forall P, B \in G_q$ and $x \in \mathbb{Z}_q^*$ [15].
2. *Diffie–Hellman Problem (DHP)* Finding x, y is hard from xyP for random occurrences of x, y, and P, where $(P, xP, yP) \in E/F_q$ and $x, y \in \mathbb{Z}_q^*$ [16].
3. *Factorization Problem (FP)* Computation of xP and yP from B is not feasible in polynomial time, where $B = xP + yP$, $P, B \in G_q$ and $x, y \in \mathbb{Z}_q^*$ [17].

3 Review of the Wazid et al.'s Protocol [11]

The symbols used in the protocol [11] are given in Table 1. The protocol [11] has three phases as discussed below:

3.1 Registration Phase

The registration phase is carried out by trusted authority (TA) in which each communicating entity, i.e., mobile device, fog server, cloud server and user, needs to register individually.

Table 1 Notations used in Wazid et al.'s protocol [11]

Notation	Description
TA	Trusted authority
U_i, D_k, CS_l	ith user, kth mobile device, lth cloud server
ID_U	User's Identity
ID_k	Mobile device$'s$ Identity
ID_l	Identity of cloud server
PW_U	Password of user
BIO_U	Biometric of user
SK	Session key
h(.)	One-way hash function
E(.), D(.)	Cryptographic encryption, decryption functions
$\oplus$, ||	XOR function, concatenation
TID_K, TID_j, TID_l	Temporary identities of communicating parties
$F(.), G(.)$	Bivariate polynomial generating functions of degree t
(TID_K, y)	Input variables to function F(.) where y ∈ GF(p)
(TID_j, y)	Input variables to function F(.) where y ∈ GF(p)
(TID_K, y)	Input variables to function G(.) where y ∈ GF(p)
$RID_K, RID_i, RID_l, RID_j$	Pseudo identities of communicating entities
TC_K, TC_i, TC_j, TC_l	Temporal credentials of communicators of system
et	Error-tolerance threshold of fuzzy extractor
$Gen(.), Rep(.)$	Functions of fuzzy extractor
G_q	Prime cyclic group
$\mathbb{A}$	Adversary/attacker
GF(p)	Finite prime Galois field
$\mathbb{Z}_q^*$	Multiplicative prime cyclic group

3.1.1 Registration of Mobile Device

A mobile device D_k is registered by executing the steps as given below:

1. TA selects RID_K, TID_K, TC_K, $F(TID_K, y)$, where $F(TID_k, y)=\sum_{m,n=0}^t [a_{m,n}(TID_k)^m]y^n, a_{m,n} \in GF(p)$; y $\in GF(p)$ and TID_k are input parameters for bivariate polynomial generating function $F(.)$ that takes values from 0 to t (degree of polynomial being generated by the function); and sends a message consisting of $\{RID_K, TID_K, TC_K, F(TID_K, y)\}$ to mobile device D_k.
2. D_k stores RID_K, TID_K, TC_K and $F(TID_K, y)$ in its memory.

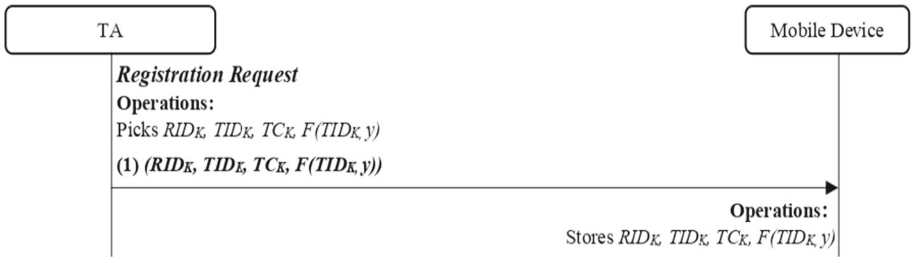

TA
Mobile Device
Registration Request
Operations:
Picks RIDK, TIDK, TCK, F(TIDK, y)
(1) (RIDK, TIDK, TCK, F(TIDK, y))
Operations:
Stores RIDK, TIDK, TCK, F(TIDK, y)

Fig. 2 Registration of Mobile Device

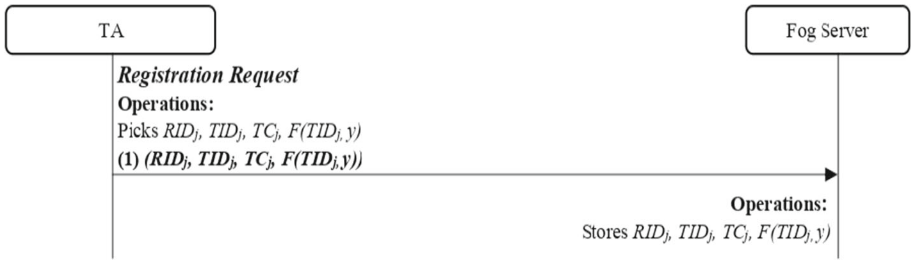

TA
Fog Server
Registration Request
Operations:
Picks RIDj, TIDj, TCj, F(TIDj, y)
(1) (RIDj, TIDj, TCj, F(TIDj, y))
Operations:
Stores RIDj, TIDj, TCj, F(TIDj, y)

Fig. 3 Registration of Fog Server

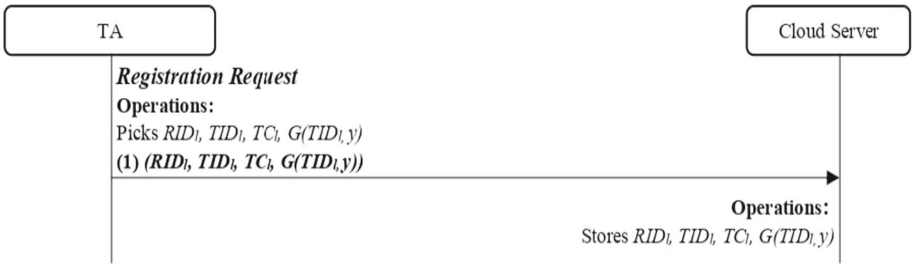

TA
Cloud Server
Registration Request
Operations:
Picks RIDl, TIDl, TCl, G(TIDl, y)
(1) (RIDl, TIDl, TCl, G(TIDl, y))
Operations:
Stores RIDl, TIDl, TCl, G(TIDl, y)

Fig. 4 Registration of Cloud Server

This procedure is diagrammatically depicted in Fig. 2.

3.1.2 Registration of Fog Server

The fog server is registered using the steps as given below:

1. TA picks RID_j, TID_j, TC_j, $F(TID_j, y)$, where $F(TID_j, y) = \sum_{m,n=0}^{t}[a_{m,n}(TID_j)^m]y^n$, $a_{m,n} \in GF(p)$; y $\in GF(p)$ and TID_j are input variate for $F(.)$ that takes values from 0 to t; and sends a message that comprises of $\{RID_j, TID_j, TC_j, F(TID_j, y)\}$ to fog server FS_j .
2. FS_j stores RID_j, TID_j, TC_j and $F(TID_j, y)$ in its database.

This procedure is shown in Fig. 3.

3.1.3 Registration of Cloud Server

The registration of cloud server is done using the following steps:

1. TA chooses RID_l, TID_l, TC_l, $G(TID_l, y)$ where $G(TID_l, y) = \sum_{m,n=0}^{t}[b_{m,n}(TID_l)^m]y^n$, $b_{m,n} \in GF(p)$; y $\in GF(p)$ and TID_l are input variable to the bivariate polynomial generating function $G(.)$ that takes values from 0 to t; and sends a message that consists of $\{RID_l, TID_l, TC_l, G(TID_l, y)\}$ to cloud server CS_l.
2. CS_l stores RID_l, TID_l, TC_l and $G(TID_l, y)$ in its database and $G(TID_l, y)$ in database of FS_j, for each CS_l.

Registration of cloud server is schematically shown in Fig. 4.

3.1.4 Registration of User

A user U_i is registered by executing the steps as given below:

1. User U_i selects a random number $d_i \in \mathbb{Z}_q^*$ and sends the registration request $\{RID_i, P_i\}$ to TA securely, where P_i is user's public key.
2. TA chooses TC_i, calculates h(stored TC_j), and sends $\{TC_i, (TID_j, h(TC_j)\}$ to U_i.

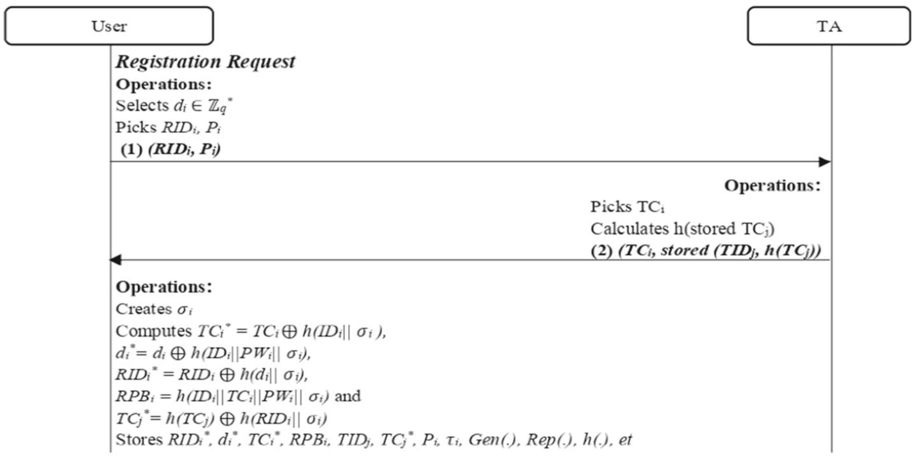

User
TA
Registration Request
Operations:
Selects $d_i \in \mathbb{Z}_q^*$
Picks RID_i, P_i
(1) (RID_i, P_i)
Operations:
Picks TC_i
Calculates h(stored TC_j)
(2) $(TC_i$, stored $(TID_j, h(TC_j))$
Operations:
Creates σ_i
Computes $TC_i^* = TC_i \oplus h(ID_i|| \sigma_i)$,
$d_i^* = d_i \oplus h(ID_i||PW_i|| \sigma_i)$,
$RID_i^* = RID_i \oplus h(d_i|| \sigma_i)$,
$RPB_i = h(ID_i||TC_i||PW_i|| \sigma_i)$ and
$TC_j^* = h(TC_j) \oplus h(RID_i|| \sigma_i)$
Stores RID_i^*, d_i^*, TC_i^*, RPB_i, TID_j, TC_j^*, P_i, τ_i, $Gen(.)$, $Rep(.)$, $h(.)$, et

Fig. 5 Registration of Remote User

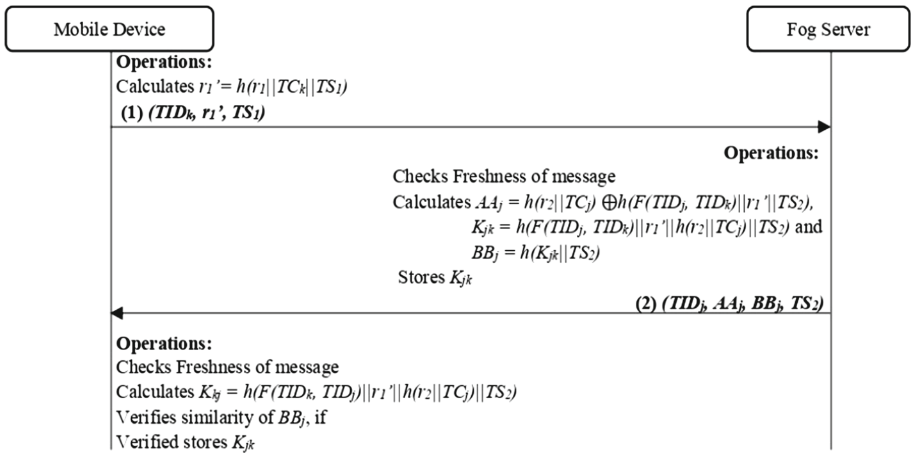

Mobile Device
Fog Server
Operations:
Calculates $r_1' = h(r_1||TC_k||TS_1)$
(1) (TID_k, r_1', TS_1)
Operations:
Checks Freshness of message
Calculates $AA_j = h(r_2||TC_j) \oplus h(F(TID_j, TID_k)||r_1'||TS_2)$,
$K_{jk} = h(F(TID_j, TID_k)||r_1'||h(r_2||TC_j)||TS_2)$ and
$BB_j = h(K_{jk}||TS_2)$
Stores K_{jk}
(2) $(TID_j, AA_j, BB_j, TS_2)$
Operations:
Checks Freshness of message
Calculates $K_{kj} = h(F(TID_k, TID_j)||r_1'||h(r_2||TC_j)||TS_2)$
Verifies similarity of BB_j, if
Verified stores K_{jk}

Fig. 6 Key negotiation among mobile devices and fog server

3. U_i creates a secret biometric key σ_i and computes $TC_i^* = TC_i \oplus h(ID_i||\sigma_i)$, $d_i^* = d_i \oplus h(ID_i||PW_i||\sigma_i)$, $RID_i^* = RID_i \oplus h(d_i||\sigma_i)$, $RPB_i = h(ID_i||TC_i||PW_i||\sigma_i)$ and $TC_j^* = h(TC_j) \oplus h(RID_i||\sigma_i)$. U_i stores $\{RID_i^*$, d_i^*, TC_i^*, RPB_i, TID_j, TC_j^*, P_i, τ_i, $Gen(.)$, $Rep(.)$, $h(.)$, $et\}$ in its memory, where τ_i is user's public reproduction parameter.

This process is depicted in Fig. 5.

3.2 Key Management Phase

In this phase, the keys are generated and exchanged among the communicating entities for data exchange and establishing the session key.

3.2.1 Key Negotiation Among Mobile Devices and a Fog Server

The key negotiation process amid a mobile device and a fog server is given below:

1. Mobile device D_K composes a request message $\{TID_k$, r_1', $TS_1\}$ to send it to FS_j, where $r_1' = h(r_1||TC_k||TS_1)$, r_1 and TS_1 are random nonce and current timestamp of the mobile device, respectively.
2. FS_j checks the freshness of received message and computes $AA_j = h(r_2||TC_j) \oplus h(F(TID_j, TID_k)||r_1'||TS_2)$, $K_{jk} = h(F(TID_j, TID_k)||r_1'||h(r_2||TC_j)||TS_2)$ and $BB_j = h(K_{jk}||TS_2)$, where r_2 and TS_2 are its random nonce and current timestamp, respectively. The fog server composes the reply message $\{TID_j, AA_j, BB_j, TS_2\}$ for D_k.
3. D_k verifies the timeliness of the message, calculates secret key $K_{kj} = h(F(TID_k, TID_j)||r_1'||h(r_2||TC_j))||TS_2)$, and performs verification over the similarity of BB_j. If the fog server is successfully verified, both D_k and FS_j store the similar secret key $K_{kj} (= K_{jk})$ for further communication.

This process is diagrammatically shown in Fig. 6.

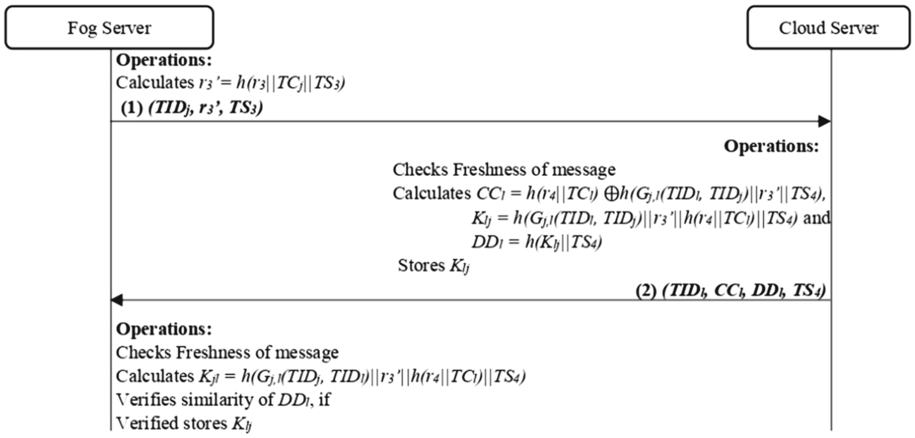

Fog Server
Cloud Server
Operations:
Calculates r3'= h(r3||TCj||TS3)
(1) (TIDj, r3', TS3)
Operations:
Checks Freshness of message
Calculates CCl = h(r4||TCl) ⊕h(Gj,l(TIDl, TIDj)||r3'||TS4),
Klj = h(Gj,l(TIDl, TIDj)||r3'||h(r4||TCl)||TS4) and
DDl = h(Klj||TS4)
Stores Klj
(2) (TIDl, CCl, DDl, TS4)
Operations:
Checks Freshness of message
Calculates Kjl = h(Gj,l(TIDj, TIDl)||r3'||h(r4||TCl)||TS4)
Verifies similarity of DDl, if
Verified stores Klj

Fig. 7 Key negotiation between fog server and cloud server

3.2.2 Key Negotiation Between a Fog Server and a Cloud Server

The process of key negotiation between a fog server and a cloud server is given below:

1. FS_j composes a request message $\{TID_j, r_3', TS_3\}$ for CS_l, where $r_3' = h(r_3||TC_j||TS_3)$, r_3 and TS_3 are its random nonce and current timestamp, respectively.
2. CS_l checks if the received message is fresh. If freshness is affirmed, the cloud server computes $CC_l = h(r_4||TC_l) \oplus h(G_{j,l}(TID_l, TID_j)||r_3'||TS_4)$, secret key $K_{lj} = h(G_{j,l}(TID_l, TID_j)||r_3'||h(r_4||TC_l)||TS_4)$ and $DD_l = h(K_{lj}||TS_4)$, where r_4 and TS_4 are its random nonce and current timestamp, respectively. The cloud server sends the reply message $\{TID_l, CC_l, DD_l, TS_4\}$ to FS_j.
3. FS_j verifies the received message as per the threshold of packet reception time, computes the secret key $K_{jl} = h(G_{j,l}(TID_j, TID_l)||r_3'||h(r_4||TC_l)||TS_4)$, and performs the verification over the similarity of DD_l. If the verification is affirmed, both CS_l and FS_j store the similar secret key $K_{jl}(= K_{lj})$ for further communication.

The process of key negotiation is depicted in Fig. 7.

3.3 Login and Authentication

The process of login and authentication among the communicating entities is given below:

1. When user U_i wants to access any information from the system, he needs to log into the system by entering his credentials. After logging into the system, user picks a random nonce r_u and computes $R_u = r_u.G$, $a_u = d_i + r_u(mod p)$, $RID_i' = RID_i \oplus h(h(TC_j)||TS_u)$, $E_u = h(TC_i||d_i||TS_u) \oplus h(h(TC_j)||RID_i)$ and $F_u = RID_k \oplus h(h(TC_j)||TS_u)$, where TS_u is its current timestamp and G is generator point. The user sends the authentication request message $\{RID_i', R_u, a_u, E_u, F_u, TS_u\}$ to FS_j .
2. FS_j validates the received message for freshness and checks the similarity of $a_u.G$ with $P_i + R_u$. If the similarity is confirmed, the fog server generates a random nonce r_f, to compute $K_{uf} = r_f.R_u = (r_u r_f).G$, $h(TC_i||d_i||TS_u) = E_u \oplus h(h(TC_j)||RID_i)$, $P_f = r_f.G$, $RID_k = F_u \oplus h(h(TC_j)||TS_u)$, $RID_i^* = h(RID_i \oplus h(K_{jk}||RID_k||TS_f)$, $RID_k^* = RID_k \oplus h(K_{jk}||TS_f)$, $G_j = h(K_{jk}||RID_k||TS_f) \oplus h(K_{uf}|| h(TC_i||d_i||TS_u)||h(RID_i))$ and $H_j = h(h(RID_i|| RID_k||G_j||P_f||TS_f)$, where TS_f is its current timestamp. FS_j composes the reply message $\{RID_i^*, RID_k^*, G_j, H_j, P_f, TS_f\}$ for D_k.
3. D_k verifies the timeliness of received data packet and checks the similarity of H_j. If the similarity is confirmed, the mobile device generates a random nonce r_k to compute $I_j = G_j \oplus h(K_{jk}||RID_k||TS_f) = h(K_{uf}||h(TC_i||d_i|| TS_u)||h(RID_i))$, $RID_k^{**} = RID_k \oplus h(h(RID_i)||TS_k)$, session key $SK_{ki} = h(I_j||h(TC_k||r_k)||TS_k)$, $M_k = h(TC_k||r_k) \oplus h(RID_k||h(RID_i)||TS_k)$ and $N_k = h(SK_{ki}||P_f||TS_k)$. Lastly, it composes the reply message $\{RID_k^{**}, M_k, N_k, P_f, TS_k\}$ for U_i, where TS_k is its current timestamp.
4. U_i verifies the timeliness of the received message. If the verification is successful, the user formulates the shared session-key $SK_{ik} = h(h(K_uf||h(TC_i||d_i||TS_u)|| h(RID_i))||h(TC_k||r_k)||TS_k)(= SK_{ki})$ and verifies the correctness of N_k. Upon successful verification, U_i authenticates D_k and the negotiated session key $SK_{ik}(= SK_{ki})$ is used for further secure communication.

Login and authentication phase is diagrammatically shown in Fig. 8.

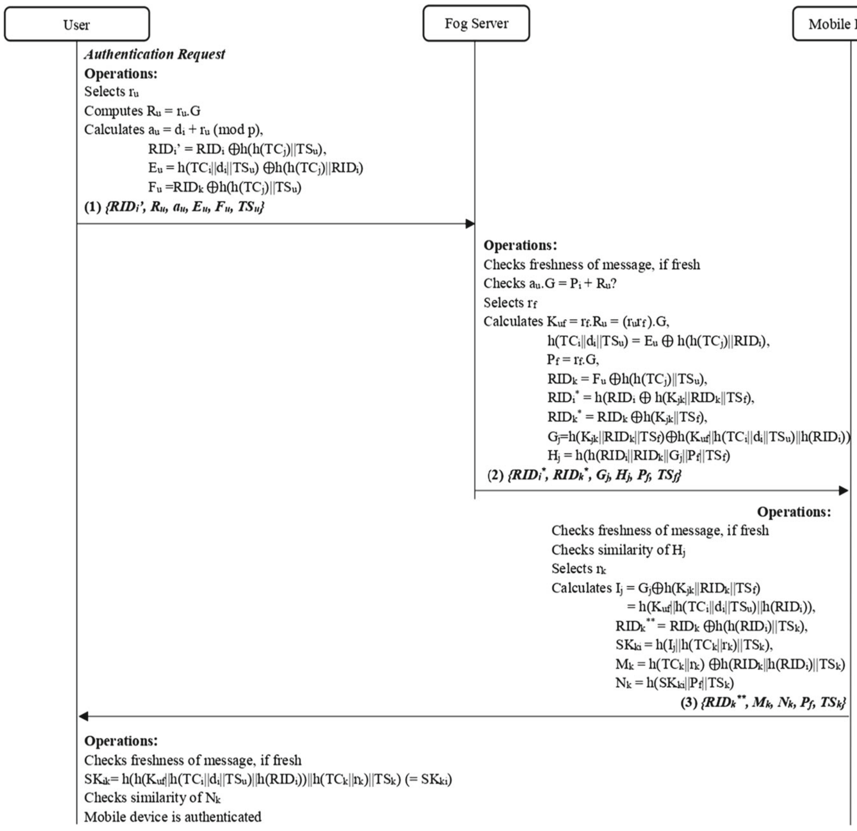

User
Fog Server
Mobile Device
Authentication Request
Operations:
Selects r_u
Computes R_u = r_u.G
Calculates a_u = d_i + r_u (mod p),
RID_i' = RID_i ⊕h(h(TC_j)||TS_u),
E_u = h(TC_i||d_i||TS_u) ⊕h(h(TC_j)||RID_i)
F_u =RID_k ⊕h(h(TC_j)||TS_u)
(1) {RID_i', R_u, a_u, E_u, F_u, TS_u}
Operations:
Checks freshness of message, if fresh
Checks a_u.G = P_i + R_u?
Selects r_f
Calculates K_uf = r_f.R_u = (r_u r_f).G,
h(TC_i||d_i||TS_u) = E_u ⊕ h(h(TC_j)||RID_i),
P_f = r_f.G,
RID_k = F_u ⊕h(h(TC_j)||TS_u),
RID_i* = h(RID_i ⊕ h(K_jk||RID_k||TS_f),
RID_k* = RID_k ⊕h(K_jk||TS_f),
G_j=h(K_jk||RID_k||TS_f)⊕h(K_uf||h(TC_i||d_i||TS_u)||h(RID_i))
H_j = h(h(RID_i||RID_k||G_j||P_f||TS_f)
(2) {RID_i*, RID_k*, G_j, H_j, P_f, TS_f}
Operations:
Checks freshness of message, if fresh
Checks similarity of H_j
Selects r_k
Calculates I_j = G_j⊕h(K_jk||RID_k||TS_f)
= h(K_uf||h(TC_i||d_i||TS_u)||h(RID_i)),
RID_k** = RID_k ⊕h(h(RID_i)||TS_k),
SK_ki = h(I_j||h(TC_k||r_k)||TS_k),
M_k = h(TC_k||r_k) ⊕h(RID_k||h(RID_i)||TS_k)
N_k = h(SK_ki||P_f||TS_k)
(3) {RID_k**, M_k, N_k, P_f, TS_k}
Operations:
Checks freshness of message, if fresh
SK_ik= h(h(K_uf||h(TC_i||d_i||TS_u)||h(RID_i))||h(TC_k||r_k)||TS_k) (= SK_ki)
Checks similarity of N_k
Mobile device is authenticated

Fig. 8 Login and authentication

4 Cryptanalysis of Wazid et al.'s Protocol [11]

In this section, we cryptanalyze the protocol [11] and show that it suffers from various vulnerabilities as discussed below.

4.1 Denial-of-Service Attack

The Wazid et al.'s protocol [11] is not resilient to denial-of-service attack as shown below:

1. D_k sends the message $\{RID_k^{**}, M_k, N_k, P_f, TS_k\}$ to U_i in step 3 of authentication phase, as discussed in Sect. 3.
2. Adversary $\mathbb{A}$ changes the timestamp TS_k to TS_k' such that it appears to be fresh. All the calculations carried out henceforth get morphed to illegitimate data from a legitimate mobile device.
3. User calculates $RID_{kM} = RID_k^{**} \oplus h(h(RID_i)||TS_k')$, $h(TC_k||r_k)_M = M_k \oplus h(RID_{kM}||h(RID_i)||TS_k')$, $K_{uf} = r_u.P_f$ and the shared session key $SK_{ikM}= h(h(K_{uf}|| h(TC_i||d_i||TS_u)||h(RID_i))||h(TC_k||r_k)_M||TS_k')$. Lastly, U_i computes $N_{kM}' = h(SK_{ikM}||P_f||TS_k')$ and checks if $N_{kM}' = N_k$.

Here, the condition for the equality of two parameters would fail after carrying out so many calculations. Since the mismatch has occurred, the legitimate entity is considered to be fraud, which means that the authentication request will be rejected. Thus, a legitimate entity of the system, mobile device cannot establish a connection. In this way, we have shown that the protocol [11] suffers from the denial-of-service attack. It is summarily shown in Fig. 9.

4.2 Stolen Smart-Card Attack

The protocol [11] suffers from the stolen smart-card attack as discussed below. If adversary $\mathbb{A}$ somehow gets the smart-card, it obtains the information $RID_k, TID_k, TC_k, K_{kj}$ stored in it using the power analysis.

1. Adversary $\mathbb{A}$ intercepts the message $\{RID_i^*, RID_k^*, G_j, H_j, P_f, TS_f\}$ sent by fog server to mobile device to calculate $I_j =$public $G_j \oplus h($stored $K_{jk}||$stored $RID_k||$public $TS_f)$ and $h(RID_i) =$public $RID_i^* \oplus h($stored $K_{jk}||$stored $RID_k||$public $TS_f)$.
2. $\mathbb{A}$ intercepts the message $\{RID_k^{**}, M_k, N_k, TS_k\}$ sent by mobile to user to calculate $h(TC_k||r_k)=$ public M_k $\oplus$ $h($stored $RID_k||$calculated $h(RID_i)||$public $TS_k)$. Lastly, $\mathbb{A}$ calculates SK_{ki} = h(calculated $I_j||$calculated $h(TC_k||r_k)||$public $TS_k)$.

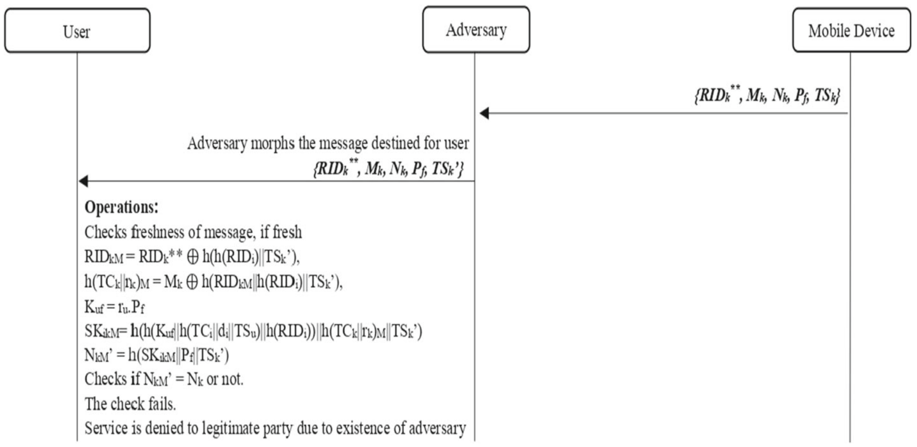

User
Adversary
Mobile Device
{RIDk**, Mk, Nk, Pf, TSk}
Adversary morphs the message destined for user
{RIDk**, Mk, Nk, Pf, TSk'}
Operations:
Checks freshness of message, if fresh
RIDkM = RIDk** ⊕ h(h(RIDi)||TSk'),
h(TCk||rk)M = Mk ⊕ h(RIDkM||h(RIDi)||TSk'),
Kuf = ru.Pf
SKikM= h(h(Kuf||h(TCi||di||TSu)||h(RIDi))||h(TCk||rk)M||TSk')
NkM' = h(SKikM||Pf||TSk')
Checks if NkM' = Nk or not.
The check fails.
Service is denied to legitimate party due to existence of adversary

Fig. 9 Denial-of-Service attack

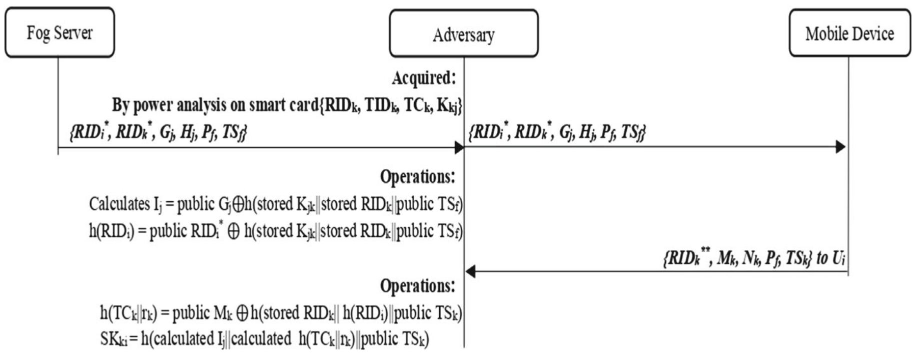

Fog Server
Adversary
Mobile Device
Acquired:
By power analysis on smart card{RIDk, TIDk, TCk, Kkj}
{RIDi*, RIDk*, Gj, Hj, Pf, TSf}
{RIDi*, RIDk*, Gj, Hj, Pf, TSf}
Operations:
Calculates Ij = public Gj⊕h(stored Kjk||stored RIDk||public TSf)
h(RIDi) = public RIDi* ⊕ h(stored Kjk||stored RIDk||public TSf)
{RIDk**, Mk, Nk, Pf, TSk} to Ui
Operations:
h(TCk||rk) = public Mk ⊕h(stored RIDk|| h(RIDi)||public TSk)
SKki = h(calculated Ij||calculated h(TCk||rk)||public TSk)

Fig. 10 Stolen smart-card attack

Likewise, $\mathbb{A}$ can generate a valid session key. Thus, the protocol [11] is not resilient to stolen smart-card attack. The attack is summarily shown in Fig. 10.

4.3 Privileged Insider Attack

The privileged insider attack is possible in authentication phase of the protocol [11] as discussed below.

1. Adversary/Insider $\mathbb{A}$ receives the message $\{RID_i', R_u, a_u, E_u, F_u, TS_u\}$ sent by user to fog server (i.e., insider). Then, it calculates RID_i = public $RID_i' \oplus h$(own $h(TC_j)||$public TS_u), $h(TC_i||d_i||TS_u)$ = public E_u $\oplus$ h(own $h(TC_j)||$calculated RID_i), $h(RID_i)$ = h(calculated RID_i), K_{uf} = own r_f.public R_u and RID_k = public $F_u \oplus h$(own $h(TC_j)||$public TS_u)
2. $\mathbb{A}$ intercepts the message $\{RID_k^{**}, M_k, N_k, P_f, TS_k\}$ being sent by mobile device to user to calculate $h(TC_k||r_k)$ = public $M_k \oplus h$(calculated $RID_k||\ h(RID_i)||$public TS_k), SK_{ik}= $h(h(K_{uf}||h(TC_i||d_i||TS_u)||h(RID_i))||\ h(TC_k||r_k)||$public $TS_k)$, I_j = public $G_j \oplus$ h(stored $K_{jk}||$ calculated $RID_k||$ public TS_f) and $SK_{ki} = h(I_j||h(TC_k||r_k)||$ public $TS_k)$.

Thus, $\mathbb{A}$ generates a valid session key at both the communicating ends, and hence, it is not robust against the privileged insider attack. Figure 11 summarily shows the privileged insider attack.

5 Proposed Protocol

Here, we introduce a user authentication protocol for cloud environment utilizing ECC by overcoming the drawbacks of the Wazid et al.'s protocol [11]. Our protocol basically has four phases: user registration, mobile device registration, user login, and mutual authentication and session-key negotiation which are elaborated subsequently. The notations or symbols used for our protocol are given in Table 2.

5.1 Initialization

Here, the system administrator (SA) selects the following system parameters:

1. A prime finite field F_q, where $q > 2^{160}$.
2. An elliptic curve E over F_q represented as (E/F_q).

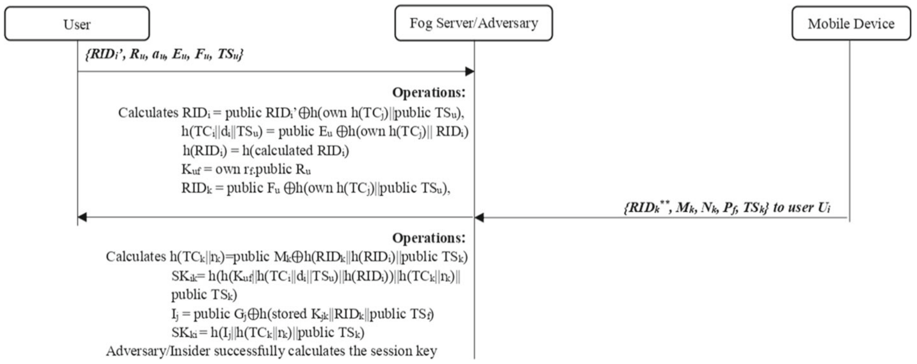

User
Fog Server/Adversary
Mobile Device
{RIDᵢ', Rᵤ, aᵤ, Eᵤ, Fᵤ, TSᵤ}
Operations:
Calculates RIDᵢ = public RIDᵢ'⊕h(own h(TCⱼ)||public TSᵤ),
h(TCᵢ||dᵢ||TSᵤ) = public Eᵤ ⊕h(own h(TCⱼ)|| RIDᵢ)
h(RIDᵢ) = h(calculated RIDᵢ)
K_uf = own r_f.public Rᵤ
RIDₖ = public Fᵤ ⊕h(own h(TCⱼ)||public TSᵤ),
{RIDₖ**, Mₖ, Nₖ, P_f, TSₖ} to user Uᵢ
Operations:
Calculates h(TCₖ||nₖ)=public Mₖ⊕h(RIDₖ||h(RIDᵢ)||public TSₖ)
SK_ik= h(h(K_uf||h(TCᵢ||dᵢ||TSᵤ)||h(RIDᵢ))||h(TCₖ||nₖ)||public TSₖ)
Iⱼ = public Gⱼ⊕h(stored K_jk||RIDₖ||public TS_f)
SK_ki = h(Iⱼ||h(TCₖ||nₖ)||public TSₖ)
Adversary/Insider successfully calculates the session key

Fig. 11 Privileged insider attack

Table 2 Notations used in our protocol

Notation	Description
SA	System administrator
U_U	Uth user
D_k	kth mobile device
CS_l	lth cloud server
UID_U	Uth user's identity
MID_k	kth mobile device$'s$ identity
CID_l	lth cloud server$'s$ identity
PW_U	Uth user$'s$ password
BIO_U	Uth user$'s$ biometric
M_U	Uth user$'s$ masked identity
Ur_U, UR_U	Uth user$'s$ random number and nonce
k	Shared symmetric pre-secret key among U_i and CS_l
β	Shared symmetric pre-secret key among D_k and CS_l
$Cr_l, Cr_{l1}, Cr_{l2}, Cr_{l3}$, $CR_l, CR_{l1}, CR_{l2}, CR_{l3}$	lth cloud server$'$s random number and nonce
$U\alpha_U, UV_U$	Uth user$'s$ private-public key pair
Mr_k, MR_k	kth mobile device$'s$ random number and nonce
$M\alpha_k, MV_k$	kth mobile device$'s$ private-public key pair
$C\alpha_l, CV_l$	lth cloud server$'s$ private-public key pair
$T_1, T_2, T_3, T_4, T_5, \Delta T$	Timestamp and threshold time
SK	Session Key
h(.)	One-way hash function
E(.), D(.)	Cryptographic encryption, decryption functions
||	Concatenation
G_q	Prime cyclic group
$\mathbb{Z}_q^*$	Multiplicative prime cyclic group

3. A generator P of order n over E/F_q.
4. A symmetric encryption/decryption algorithm agreed upon by the communicating parties.
5. An identity MID_k for mobile device.
6. A shared pre-secret symmetric key β for mobile device and cloud server that is stored into their memory.
7. A shared pre-secret symmetric key k for user and cloud server that is stored in smart device and database.

The cloud server in initialization phase selects a private key $C\alpha_l$ and calculates the public key as $CV_l = C\alpha_l.P$.

5.2 User Registration

Every user U_U has to register himself with the cloud server by executing the following steps.

1. User U_U randomly picks an identity UID_U, password PW_U and imprints biometric BIO_U as his secret credentials to calculate his masked identity $M_U = h(UID_U||PW_U||BIO_U)$. Further, he picks a random number $Ur_U \in \mathbb{Z}_q^*$ to compute $K = k.P$, $UR_U = Ur_U.P$, $A_U = h(M_U||UR_U||K)$ and sends the encrypted registration request $E_k(A_U, UR_U, M_U)$ to the cloud server CS_l, where confidentiality is ensured via the shared pre-secret k.
2. On reception of the request from U_U, CS_l computes $K = k.P$ and decrypts the message $D_{E_k}(A_U, UR_U, M_U)$ which gives A_U, UR_U, M_U to calculate $A'_U = h(\text{received } M_U||\text{received } UR_U||K)$. CS_l verifies the authenticity of message by checking if $A'_U =$ received A_U. If the condition is false, then the session is closed; otherwise, the cloud server calculates $HID_l = h(CID_l||K)$ and picks a random number $Cr_l \in \mathbb{Z}_q^*$ to compute $CR_l = Cr_l.P + K$, $A_l = h(\text{received } M_U||HID_l||K||CR_l)$. Lastly, CS_l stores M_U and sends the reply message $\{Cr_l, HID_l, A_l\}$ to U_U
3. Upon reception of the reply message, U_U calculates CR'_l =received $Cr_l.P + K$, $A'_l = h(M_U||\text{received } HID_l||K||CR'_l)$ and checks if $A'_l =$ received A_l for verifying that the message has come from a legitimate source. If it fails, the session is terminated; otherwise, U_U picks a random number $r_{Ul} \in \mathbb{Z}_q^*$ to compute the private-key

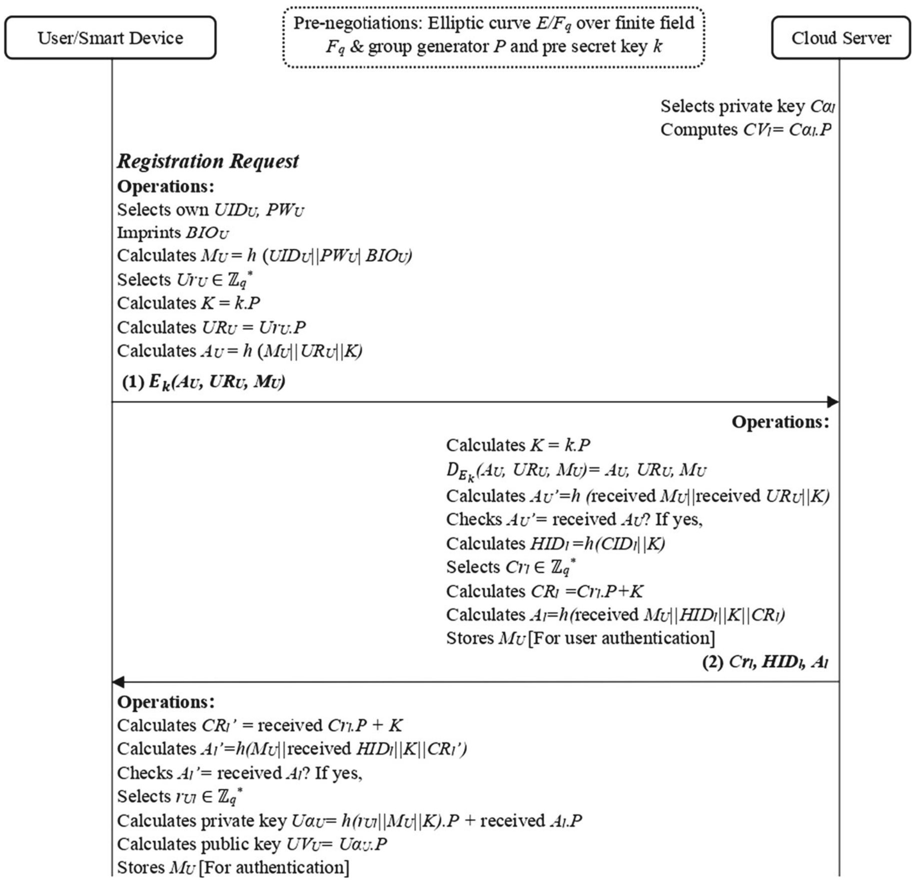

User/Smart Device
Pre-negotiations: Elliptic curve E/F_q over finite field F_q & group generator P and pre secret key k
Cloud Server
Selects private key $C\alpha_l$
Computes $CV_l = C\alpha_l.P$
Registration Request
Operations:
Selects own UID_U, PW_U
Imprints BIO_U
Calculates $M_U = h\ (UID_U||PW_U||BIO_U)$
Selects $Ur_U \in \mathbb{Z}_q^*$
Calculates $K = k.P$
Calculates $UR_U = Ur_U.P$
Calculates $A_U = h\ (M_U||UR_U||K)$
(1) $E_k(A_U, UR_U, M_U)$
Operations:
Calculates $K = k.P$
$D_{E_k}(A_U, UR_U, M_U) = A_U, UR_U, M_U$
Calculates $A_U' = h$ (received M_U||received $UR_U||K$)
Checks A_U' = received A_U? If yes,
Calculates $HID_l = h(CID_l||K)$
Selects $Cr_l \in \mathbb{Z}_q^*$
Calculates $CR_l = Cr_l.P+K$
Calculates $A_l = h$(received $M_U||HID_l||K||CR_l$)
Stores M_U [For user authentication]
(2) Cr_l, HID_l, A_l
Operations:
Calculates CR_l' = received $Cr_l.P + K$
Calculates $A_l' = h(M_U$||received $HID_l||K||CR_l')$
Checks A_l' = received A_l? If yes,
Selects $r_{Ul} \in \mathbb{Z}_q^*$
Calculates private key $U\alpha_U = h(r_{Ul}||M_U||K).P$ + received $A_l.P$
Calculates public key $UV_U = U\alpha_U.P$
Stores M_U [For authentication]

Fig. 12 User Registration Phase

$U\alpha_U = h(r_{Ul}||M_U||K).P+$received $A_l.P$ and public key $UV_U = U\alpha_U.P$. Finally, M_U is stored into the smart device.

Figure 12 summarizes the user registration process.

5.3 Mobile Device Registration

Every mobile device D_k needs to be registered with the cloud server by executing the following steps:

1. Mobile device D_k picks a random number $Mr_k \in \mathbb{Z}_q^*$ to compute $MR_k = Mr_k.P$, $A_k = h(MID_k||MR_k||\beta)$, $B_k = A_k.P + MR_k$ and transmits the registration request consisting of $\{MID_k, MR_k, B_k\}$ to CS_l.
2. On receiving the registration request from D_k, CS_l calculates A_k' = h(received MID_k||received $MR_k||\beta$) and B_k' = calculated $A_k'.P+$ received MR_k to check the legitimacy of the message by verifying if B_k' = received B_k. If it is false, the session is aborted; otherwise, CS_l calculates $HID_{l1} = h(CID_l||\beta)$ and picks an arbitrary number $Cr_{l1} \in \mathbb{Z}_q^*$ to compute $CR_{l1} = Cr_{l1}.P + \beta.P$ and $B_l = h$(received $B_k||HID_{l1}||\beta||CR_{l1}$). Lastly, CS_l stores A_k and B_l and sends the reply message $\{Cr_{l1}, B_l, HID_{l1}\}$ to D_k
3. On receiving the reply message, D_k calculates CR_{l1}' = received $Cr_{l1}.P + \beta.P$ and $B_l' = h(B_k$||received $HID_{l1}||\beta$||calculated CR_{l1}') to check whether B_l' = received B_l. If they do not agree, the session is terminated; otherwise, D_k computes private key $M\alpha_k = h(Mr_k||A_k||\beta).P+$ received $B_l.P$ and public key $MV_k = M\alpha_k.P$. Finally, A_k and B_l are stored in the mobile device's memory.

The mobile device registration process is summarized in Fig. 13.

5.4 User Login

For accessing any information from the system, user U_U needs to log into the system via a smart device. So, the user U_U feeds UID_U, PW_U, BIO_U into the smart device for creating a login request message in the form of masked identity M_U. When the smart device gets the login request, it computes $M_U' = h$(received UID_U||received PW_U||received BIO_U) and verifies if M_U' = stored M_U. If both match, then the user is considered to be genuine and is given access to

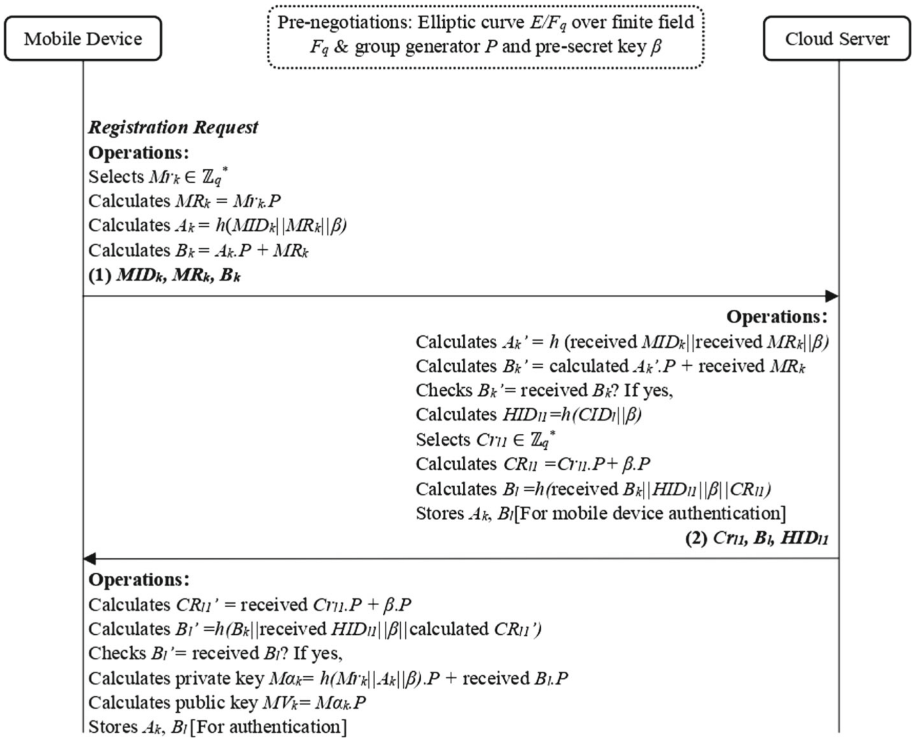

Mobile Device
Pre-negotiations: Elliptic curve E/F_q over finite field F_q & group generator P and pre-secret key β
Cloud Server
Registration Request
Operations:
Selects $Mr_k \in \mathbb{Z}_q^*$
Calculates $MR_k = Mr_k.P$
Calculates $A_k = h(MID_k||MR_k||\beta)$
Calculates $B_k = A_k.P + MR_k$
(1) MID_k, MR_k, B_k
Operations:
Calculates $A_k' = h$ (received MID_k||received $MR_k||\beta)$
Calculates B_k' = calculated $A_k'.P$ + received MR_k
Checks B_k'= received B_k? If yes,
Calculates $HID_{l1}=h(CID_l||\beta)$
Selects $Cr_{l1} \in \mathbb{Z}_q^*$
Calculates $CR_{l1} = Cr_{l1}.P+ \beta.P$
Calculates $B_l = h$(received $B_k||HID_{l1}||\beta||CR_{l1})$
Stores A_k, B_l[For mobile device authentication]
(2) Cr_{l1}, B_l, HID_{l1}
Operations:
Calculates CR_{l1}' = received $Cr_{l1}.P + \beta.P$
Calculates $B_l' = h(B_k||$received $HID_{l1}||\beta||$calculated $CR_{l1}')$
Checks B_l'= received B_l? If yes,
Calculates private key $M\alpha_k= h(Mr_k||A_k||\beta).P$ + received $B_l.P$
Calculates public key $MV_k= M\alpha_k.P$
Stores A_k, B_l [For authentication]

Fig. 13 Mobile Device Registration Phase

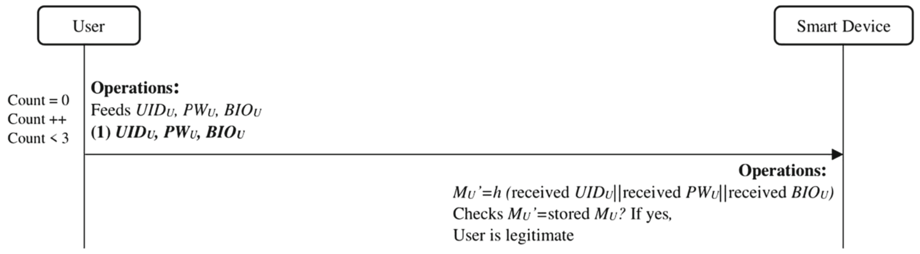

User
Smart Device
Count = 0
Count ++
Count < 3
Operations:
Feeds UID_U, PW_U, BIO_U
(1) UID_U, PW_U, BIO_U
Operations:
$M_U'=h$ (received UID_U||received PW_U||received $BIO_U)$
Checks M_U'=stored M_U? If yes,
User is legitimate

Fig. 14 User Login Phase

the system; otherwise, the session is terminated. The process to log into the system is summarized in Fig. 14.

5.5 Mutual Authentication and Session-Key Negotiation

In this part, authentication is done followed by session key generation by using the following steps:

1. User U_U records the current time as T_1 to compute $M_1 = h(\text{stored } M_U||UR_U||k)$ and $M_2 = h(M_1||T_1||(CV_l + UR_U))$. U_U sends the authentication request message comprising of $\{M_2, UR_U, T_1\}$ to CS_l
2. After receiving the authentication request, CS_l records the current time T_2 to check the freshness of the message as $|T_2 - T_1| \leq \Delta T$, where ΔT refers to the threshold time limit of receiving the message. If the condition is false, the session is dismissed; otherwise, CS_l computes $M_1' = h(\text{stored } M_U||\text{received } UR_U||k)$, $M_2' = h(M_1'||\text{received } T_1||(CV_l{+}\text{received } UR_U))$ to check whether M_2' = received M_2. If the condition holds, U_U is authenticated; otherwise, the session is terminated. After authenticating the user, CS_l picks a random number $Cr_{l2} \in \mathbb{Z}_q^*$ to calculate $CR_{l2} = Cr_{l2}.P$, $M_3 = h(\text{stored } A_k||\beta||\text{received } UR_U)$ and $M_4 = h(M_3||T_2||(MV_k + CR_{l2}))$. CS_l composes a fresh authentication request message containing $\{M_4, CR_{l2}, UR_U, T_2\}$ and sends it to D_k.
3. In reply of the request received from CS_l, D_k records the current time as T_3 to check the timeliness of the request as $|T_3 - T_2| \leq \Delta T$. If the inequality does not follows, session is terminated; otherwise, D_k computes $M_3' = h(\text{stored } A_k||\beta||\text{received } UR_U)$ and $M_4' = h(M_3'||\text{received } T_2||(MV_k{+}\text{received } CR_{l2}))$ to check if M_4' = received M_4. If they do not match, the session is

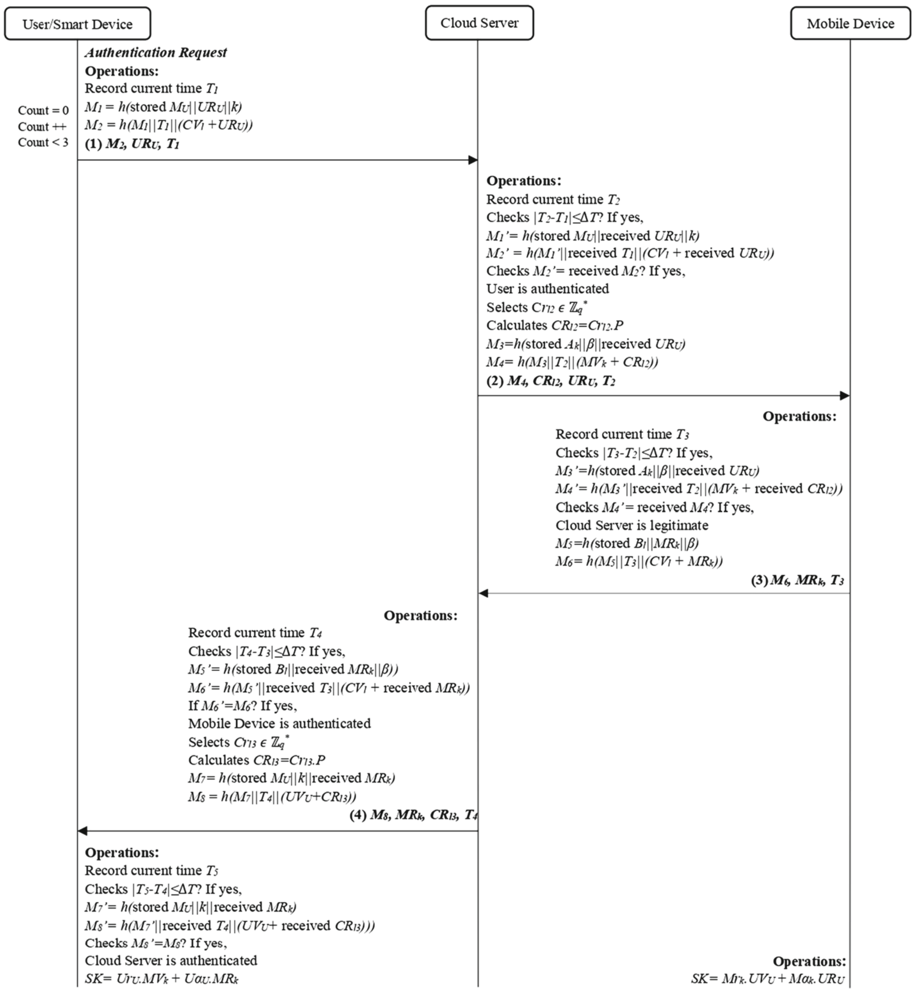

User/Smart Device
Cloud Server
Mobile Device
Authentication Request
Operations:
Record current time T_1
Count = 0
$M_1 = h$(stored $M_U||UR_U||k$)
Count ++
$M_2 = h(M_1||T_1||(CV_l + UR_U))$
Count < 3
(1) M_2, UR_U, T_1
Operations:
Record current time T_2
Checks $|T_2 - T_1| \leq \Delta T$? If yes,
$M_1' = h$(stored M_U||received $UR_U||k$)
$M_2' = h(M_1'$||received $T_1||(CV_l$ + received UR_U))
Checks M_2'= received M_2? If yes,
User is authenticated
Selects $Cr_{l2} \in \mathbb{Z}_q^*$
Calculates $CR_{l2}=Cr_{l2}.P$
$M_3=h$(stored $A_k||\beta||$received UR_U)
$M_4= h(M_3||T_2||(MV_k + CR_{l2}))$
(2) M_4, CR_{l2}, UR_U, T_2
Operations:
Record current time T_3
Checks $|T_3 - T_2| \leq \Delta T$? If yes,
$M_3'=h$(stored $A_k||\beta||$received UR_U)
$M_4'= h(M_3'$||received $T_2||(MV_k$ + received CR_{l2}))
Checks M_4'= received M_4? If yes,
Cloud Server is legitimate
$M_5=h$(stored $B_l||MR_k||\beta$)
$M_6= h(M_5||T_3||(CV_l + MR_k))$
(3) M_6, MR_k, T_3
Operations:
Record current time T_4
Checks $|T_4 - T_3| \leq \Delta T$? If yes,
$M_5'= h$(stored B_l||received $MR_k||\beta$))
$M_6'= h(M_5'$||received $T_3||(CV_l$ + received MR_k))
If $M_6'=M_6$? If yes,
Mobile Device is authenticated
Selects $Cr_{l3} \in \mathbb{Z}_q^*$
Calculates $CR_{l3}=Cr_{l3}.P$
$M_7= h$(stored $M_U||k||$received MR_k)
$M_8 = h(M_7||T_4||(UV_U+CR_{l3}))$
(4) M_8, MR_k, CR_{l3}, T_4
Operations:
Record current time T_5
Checks $|T_5 - T_4| \leq \Delta T$? If yes,
$M_7'= h$(stored $M_U||k||$received MR_k)
$M_8'= h(M_7'$||received $T_4||(UV_U$+ received CR_{l3})))
Checks $M_8'=M_8$? If yes,
Cloud Server is authenticated
$SK= Ur_U.MV_k + U\alpha_U.MR_k$
Operations:
$SK= Mr_k.UV_U + M\alpha_k.UR_U$

Fig. 15 Mutual authentication and session-key negotiation

closed; otherwise, the cloud server is considered as legitimate.

After verifying the legitimacy of cloud server, D_k computes M_5=h(stored $B_l||MR_k||\beta$), $M_6 = h(M_5||T_3||(CV_l + MR_k))$ to create a new mutual authentication request containing $\{M_6, MR_k, T_3\}$ for CS_l.

4. Cloud server CS_l records the current time as T_4 and verifies if $|T_4 - T_3| \leq \Delta T$. If the condition is not verified, the session is terminated; otherwise, CS_l computes $M_5' = h$(stored B_l||received $MR_k||\beta$)), $M_6' = h(M_5'$||received $T_3||(CV_l$+received MR_k)) to verify the authenticity of the message if M_6' = received M_6. If the condition is satisfied, the mobile device is authenticated.

   After authenticating the mobile device, CS_l picks a random number $Cr_{l3} \in \mathbb{Z}_q^*$ to calculate $CR_{l3} = Cr_{l3}.P$, $M_7 = h$(stored $M_U||k||$received MR_k) and $M_8 = h(M_7||T_4||(UV_U + CR_{l3}))$. It composes a fresh mutual authentication request message comprising of $\{M_8, MR_k, CR_{l3}, T_4\}$ for user.

5. User U_U records the current time as T_5 and verifies whether $|T_5 - T_4| \leq \Delta T$. If the inequality does not follows, session is terminated; else, U_U evaluates $M_7' = h$(stored $M_U||k||$received MR_k) and M_8' = h(M_7'|| received $T_4||(UV_U$+received CR_{l3}))) to verify M_8' = received M_8. If they match, the cloud server is authenticated.

   After validating the cloud server, U_U computes the session key as $SK = Ur_U.MV_k + U\alpha_U.MR_k$, where Ur_U is the user's random number, MV_k is the public key of mobile device, $U\alpha_U$ is user's private key, and MR_k is random nonce of mobile device.

   Similarly, the mobile device computes the session key as $SK = Mr_k.UV_U + M\alpha_k.UR_U$, where Mr_k is random number of the mobile device, UV_U is user's public key, $M\alpha_k$ represents the mobile device's private key, and UR_U is random nonce of user.

   The user and mobile device can directly converse with each other via the negotiated session key SK without the intervention of the cloud-server.

The mutual authentication process is summarized in Fig. 15.

6 Security Analysis

Here, we discuss different attacks and show that our protocol is invulnerable to them.

1. *Impersonation attack* When an adversary conducts itself as a genuine communicating authority of the system with the intention of fraudulence, then this outbreak is termed as impersonation attack [18].
2. *Replay attack* An authorized message communicated in a previous sessions is duplicated or illegally delayed in the system is termed as the replay attack [19].
3. *Denial of Service* Congestion of the server due to overflow of false service requests made by an adversary causing the exhaustion of resources that in turn leads to inaccessibility of services to the valid user is termed as the denial of service [20].
4. *Session key computation* Negotiation of session key for each session takes place amid communicating parties for secure data exchange. The session key should be formulated such that even if there is any leakage of the private/temporary parameter from the system, then also the hardness of session key should remain intact [21].
5. *Session-specific transient information attack* Accidental exposure of any of the transient parameters must not lead to compromise of the session key. This is known as session-specific transient information attack [22].
6. *Privileged insider attack* When a legitimate communicating party of the system acts as an adversary with the intention of fraudulence, then the attack is termed as privileged insider attack [23].
7. *Confidentiality* The property of secure communication where only the authorized users could identify the messages being communicated in the system is called confidentiality.
8. *Integrity* The characteristic of safe communication where the messages being passed over a public channel is preserved against any alterations is called integrity.
9. *Availability* It refers to the situation when all the authorized users of the system should be able to access the information as and when needed.
10. *Anonymity* The condition when a user or message of the system is not traceable through any activity of eavesdropping is called anonymity.

6.1 Informal Security Analysis

Here, the robustness of our proposed protocol toward various active and passive attacks is proved using different propositions.

Proposition 1 *Our protocol ensures secure three-factor mutual authentication.*

Proof It has three security factors (identity (UID_U), password (PW_U), and biometric (BIO_U)), for mutually authenticating any user of the system. Even if an attacker acquires any two of the above security factors, he still cannot pose as a legal user because generating an authenticated request message is not feasible. The request message M_2 is composed of M_1, UR_U, CV_l, T_1 and M_1, in turn comprises of M_U, UR_U, k, which are all in one-way hash format. So, doing reverse engineering on them is not possible. Here, M_U is the hashed masked identity of the user and without the knowledge of all three security features forging M_U is not feasible. Also user's biometric credential forgery is impossible; hence, M_U could not be computed.
Assuming that after acquiring two security factors, adversary somehow gets the third factor, he still cannot formulate a legit authentication request user and cloud server. Moreover, morphing any parameter leads to change in the hashed values which can be easily identified by the cloud server. Thus, our protocol ensures secure three-factor mutual authentication.

Proposition 2 *Our protocol is robust against the stolen/lost smart-card/device attack.*

Proof If the smart-card/device is acquired by an adversary, he can get M_U that is stored in smart device via the power analysis attack, where $M_U = h(UID_U||PW_U||BIO_U)$. During mutual authentication, the adversary needs to form a legal authentication request message consisting of M_2, UR_U, and T_1. However, M_2 cannot be computed as $M_2 = h(M_1||T_1||(CV_l+UR_U))$ where $M_1 = h(\text{stored } M_U||UR_U||k)$ and k is pre-shared symmetric key. Even if the attacker has the knowledge about M_U, he cannot form the valid M_1 as k is unknown to him, which leads to failed generation of M_2. So, it is not possible for an adversary to generate a valid authentication request message. Hence, our protocol is robust toward stolen smart-card attack.

Proposition 3 *Our protocol is safe from the user impersonation attack.*

Proof If an adversary eavesdrops the mutual authentication request over a public channel, then he can obtain the message $\{M_2, UR_U, T_1\}$. Further, if he tries to forge the request message to impersonate as the legal user, then there are the following two conditions:

Alteration of UR_U: As the random nonce UR_U is being sent over the public channel, it could be forged, but the forgery would be detected by the cloud server as M_2 is a hashed entity that contains UR_U.

Alteration of M_2: M_2 is composed of M_1, which in turn comprises of the masked identity M_U of the genuine user. M_U is formed by the user during registration that contains the user's biometric which is impossible to forge. Moreover, M_1 contains the secret pre-shared symmetric key k which is unknown to adversary. Hence, the adversary cannot generate a valid M_2 as any alterations made in M_U would be identified by the cloud server during message integrity check and would result in session termination. Therefore, our protocol is safe from the user impersonation attack.

Proposition 4 *Our protocol is resilient to the cloud-server impersonation attack.*

Proof When an adversary tries to impersonate as a cloud server during mutual authentication, he would try to morph $\{M_4, CR_{l2}, UR_U, T_2\}$ message, being sent to the mobile device. The parameters of the message are $M_4 = h(M_3||T_2||(MV_k + CR_{l2}))$ and $M_3 = h(A_k||\beta||UR_U)$. If the adversary attempts to compute M_3, he must have the prior knowledge of A_k and β, where $A_k = h(MID_k||MR_k||\beta)$ and β is the pre-shared secret symmetric key. However, if any of the parameters is forged by the adversary, it would reflect in the changed hash value of M_4 and would be detected by the mobile device.

Similarly, if the adversary tries to alter the message $\{M_8, MR_k, CR_{l3}, T_4\}$ being sent to the user during mutual authentication, where $M_8 = h(M_7||T_4||(UV_U+CR_{l3}))$ and $M_7 = h(M_U||k||MR_k)$, he would not be able to do so because the parameter M_8 is composed of M_7 that contains the pre-shared secret symmetric key k and M_U, which are known to user and server only. These alterations would lead to the failed integrity check at the user side. Hence, our protocol is safe from the cloud server impersonation attack.

Proposition 5 *Our protocol is resilient to the mobile device impersonation attack.*

Proof During authentication, the mobile device sends the message $\{M_6, MR_k, T_3\}$ to cloud server, where $M_6 = h(M_5||T_3||(CV_l + MR_k))$ and $M_5 = h(B_l||MR_k||\beta)$. To impersonate the mobile device, adversary requires valid B_l and pre-shared secret symmetric key β. Since adversary has no prior knowledge about these parameters and guessing two parameters in polynomial time is infeasible, morphing any value would be detected at the other communicating entity side during the integrity checks. Moreover, the messages contain the public parameters like timestamp and random nonce in the hashed format, thereafter making any forgery and alterations is infeasible. So, our protocol is resilient against the mobile device impersonation. □

Proposition 6 *Our protocol is robust toward the man-in-the-middle attack.*

Proof Let there be an adversary between a user and the server. To carry out the attack, he would capture the message $\{M_2, UR_U, T_1\}$ during mutual authentication, where $M_2 = h(M_1||T_1||(CV_l + UR_U))$ and $M_1 = h(\text{stored } M_U||UR_U||k)$. Then, he would try to generate $M_{1A} = h(\text{stored } M_{UA}||UR_{UA}||k_A)$ for composing M_2. He would need to form a valid $M_{UA} = h(UID_{UA}||PW_{UA}||BIO_{UA})$. Even if the adversary is successful in generating or guessing the parameter in polynomial time, forming the valid M_U is impractical as discussed in proposition 1. Further, he would not be able to form the valid M_1 as it contains pre-shared secret symmetric key k and user's random number UR_U is unknown to adversary. Moreover, guessing two parameters in polynomial time is not feasible. Further, any alteration in the value of M_1 leads to the changed hash digest of M_2, which would be easily detected by the cloud server during the integrity check as it stores the parameters M_U and k resulting in session termination on integrity check failure. Henceforth, our protocol is safe from man-in-the-middle attack.

Proposition 7 *Our protocol is safe from the replay attack.*

Proof In replay attack, adversary captures a message of the current session in mutual authentication phase and forwards it to the legal users in later sessions. In the proposed protocol, all the communications are carried out among the genuine parties, where the current recorded time is shared and used as a parameter of the hashed messages. So, even if the adversary tries to replay the messages just by morphing the timestamp, he would not be able to pass the integrity checks. Hence, our protocol stands strong against the replay attack.

Proposition 8 *Our protocol withstands the privileged insider attack.*

Proof Let us consider the insider be within the cloud server. He would have the knowledge about a few secrets of the communicating parties that could be used by him for posing as a legitimate entity. In our protocol, all the secret credentials of the user and mobile device like M_U and B_k are stored by the cloud server in a non-invertible hashed format. However, predicting more than one parameter simultaneously in polynomial time is not feasible. Hence, by using the stored value in the cloud server, any insider would not be able to form a legit mutual authentication requisition message. So, our protocol is free from privileged insider attack.

Proposition 9 *Our protocol is resilient from off-line user identity (UID_U) predicting attack.*

Proof If an adversary tries to predict the identity (UID_U) of a user from his smart device via power analysis or through eavesdropping messages in authentication phase, then it would be a failed attempt, as already discussed in proposition 1 that even after predicting the identity of user (UID_U), composing a valid masked identity $M_U = h(UID_U||PW_U||BIO_U)$ is not practical. However, M_U has user's other secret credentials like biometric, which cannot be forged. Therefore, our protocol ensures resilience against the off-line user identity (UID_U) predicting attack.

Proposition 10 *Our protocol is safe from off-line password predicting attack.*

Proof An adversary cannot predict the password of a user by acquiring the smart device and carrying out power analysis attack because it would require other authentication parameters like identity and biometric of the user. Moreover, in mutual authentication phase, the password is nowhere used; instead, the masked identity is used, which is in non-invertible hashed format. Hence, guessing the password is impossible for the adversary. Thus, our protocol is safe from the off-line password prediction.

Proposition 11 *Our protocol is resilient from the session-key computation attack.*

Proof A fresh session-key (SK) is generated among the mobile device and user for each new session. The session-key at user side is $SK = Ur_U.MV_k + U\alpha_U.MR_k$, where Ur_U and MR_k are the random numbers of user and nonce of device, respectively, selected fresh for each session. $U\alpha_U$ and MV_k are the user's private key and mobile device's public key, respectively. Similarly, the session key at mobile device side is $SK = Mr_k.UV_U + M\alpha_k.UR_U$, where Mr_k and UR_U are the random numbers of mobile device and nonce of user, respectively, selected fresh for each session. $M\alpha_k$ and UV_U are mobile device's private key and user's public key, respectively. Here, the security of session-key SK depends on the secret random numbers and private key of the user or mobile device. Hence, even if the random numbers are somehow exposed, the adversary still lacks the knowledge about user's or mobile device's private key. Thus, our protocol is resilient from session-key computation attack.

Proposition 12 *Our protocol is free from the known session-specific transient information attack.*

Proof Let somehow any of the temporary parameters (MR_k, Ur_U) are leaked to the adversary. Still he cannot compute the session-key because the user's private key ($U\alpha_U$) is also required. Moreover, guessing two parameters (MR_k, Ur_U) simultaneously in polynomial time is not practical. Hence, our protocol is free from the known session-specific transient information attack.

Proposition 13 *Our protocol ensures the user anonymity.*

Proof It ensures the user anonymity because the user's identity is never sent over a public channel in the raw form; rather, it is sent in the form of masked identity, which is non-invertible hash digest. Further, the masked identity used for login and authentication is stored at server side in the hash format, which ensures the user anonymity.

Proposition 14 *Our protocol is safe from the user and mobile device untraceability attacks.*

Proof While carrying out the attack, the adversary eavesdrops two authentication requests from diverse sessions and compares if both request messages are identical. If they are identical, then both the authentication request messages are sent by the same entity. Here, after eavesdropping the authentication request messages $\{M_2, UR_U, T_1\}$, where $M_2 = h(M_1||T_1||(CV_l + UR_U))$, $M_1 = h(storedM_U||UR_U||k)$ or $\{M_6, MR_k, T_3\}$, where $M_6 = h(M_5||T_3||(CV_l + MR_k))$, and $M_5 = h(B_l||MR_k||\beta)$. The adversary would not be able to trace the user or mobile device as the messages contain random nonce and timestamp which is changed after each session. Thus, the user and mobile device cannot be traced, and our protocol is resilient to the user and mobile device untraceability attacks.

Proposition 15 *The proposed protocol has strong perfect forward-secrecy.*

Proof Here, if any one of the long-term key k (between user and server) or β (between mobile device and server) is compromised, the negotiated session key SK is still safe. This is because the session-key SK is calculated by the user as $SK = Ur_U.MV_k + U\alpha_U.MR_k$, where Ur_U and $U\alpha_U$ are user's secrets. Similarly, the session key at mobile device is $SK = Mr_k.UV_U + M\alpha_k.UR_U$, where Mr_k and $M\alpha_k$ are the secrets of mobile device. Thus, by acquiring long terms key k or β, the adversary cannot calculate the session-key SK due to the unavailability of the private key and random number of the user/mobile device. Hence, the scheme has strong perfect forward secrecy attack.

In conclusion, using the state-of-the-art informal security analysis, it has been proved that our protocol is robust and provably safe from all the foreknown attacks.

6.2 Formal Security Analysis: BAN Logic

This subsection provides the proof of correctness of our protocol, which is formally verified using the BAN logic [24]. This evaluation model is based on few assumptions and claims, as mentioned in Tables 3, 4, 5, and 6. Table 4 highlights the rules of the BAN logic, wherein if the numerator is followed, then the denominator part is considered to follow. Table 5 gives the idealized form of messages as per authentication phase our proposed protocol which illustrates in details the message composition being sent from one party to another. For example, $U_U \xrightarrow{viaCS_l} D_k : T_1, UR_U : \langle UR_U\rangle_{Ur_U.P+K}, M_2 : \langle M_1\rangle_{(UR_U||M_U||k)}, T_1, \langle CV_l\rangle_{C\alpha_l.P}, \langle UR_U\rangle_{Ur_U.P+K}$ depicts that U_U sends the message $\{T_1, UR_U, M_2\}$ to D_k via sending it to CS_l in the first place and $UR_U : \langle UR_U\rangle_{Ur_U.P+K}$

Table 3 Notations of BAN

Notation	Description
$X\vert\equiv P$	X believes P to be true
$\#(P)$	P is considered fresh
$X\vert\sim P$	X once said P, i.e., X once sent a message pertaining P
P, Q_K	P/Q is encoded by symmetric key K
$\langle P\rangle_Q$	P contains Q
(P, Q)	P/Q is part of (P,Q)
$X \Rightarrow P$	X has authority over P
$\langle P\rangle_{K\mapsto X}$	P/Q is encoded by public key K of X
$(P, Q)_K$	P/Q is hashed using key K
$X \overset{K}{\longleftrightarrow} Y$	X and Y can securely communicate via shared key K
P/Q	If P holds then Q is followed
$Statement_i$	ith statement
$X \triangleleft P$	X sees P

Table 4 Basic inferences for BAN logic

Inference rules	Definitions
Message meaning rule (MMR)	$\frac{X\vert\equiv X\overset{K}{\longleftrightarrow}Y,\, X\triangleleft\langle M\rangle_N}{X\vert\equiv Y\vert\sim M}$
If X believes K is shared by Y and sees $\langle M\rangle_N$, then X believes, Y once said M	
Nonce verification rule (NVR)	$\frac{X\vert\equiv\#(M),\, X\vert\equiv Y\vert\sim M}{X\vert\equiv Y\vert\equiv M}$
If X believes, M to be fresh and Y once sent M, then X trusts, Y trusts M	
Jurisdiction rule (JR)	$\frac{X\vert\equiv Y\Rightarrow M,\, X\vert\equiv Y\vert\equiv M}{X\vert\equiv M}$
If X believes, Y has authority over M and Y believes M, then X believes M	
Freshness conjuncatenation rule (FCR)	$\frac{X\vert\equiv\#(M)}{X\vert\equiv\#(M,N)}$
If X believes, M is fresh, then X believes freshness of (M, N)	
Belief rule (BR)	$\frac{X\vert\equiv(M),\, X\vert\equiv(N)}{X\vert\equiv(M,N)}$
If X believes, M and N, then X believes (M, N)	
Session key rule (SKR)	$\frac{X\vert\equiv\#(M),\, X\vert\equiv Y\vert\equiv M}{X\vert\equiv X\overset{K}{\longleftrightarrow}Y}$
If X believes, M is fresh and Y believes M, an important factor of the session-key, then X believes, it shares the session-key K with Y	

shows that UR_U is composed of only $\langle UR_U\rangle$ which in turn is calculated as $Ur_U.P + K$ and likewise the composition of other parameters is shown in Table 5. In Table 6, the assumptions of form $X| \equiv \#\{P\}$ represent that X believes P to be fresh and $X| \equiv X \overset{A}{\longleftrightarrow} Y$ represent that X believes that A is shared amidst X *and* Y. Table 6 is formed as per our proposed protocol where each entity considers its own timestamp, all random numbers and all random nonce to be fresh. Moreover, user and mobile device believe that the pre-secret key and random number formulated as per the proposed protocol with cloud server are shared amid the respective pair.

6.2.1 Goals

For proving the security feature of the offered protocol below, mentioned goals should be satisfied.

Goal 1 $U_U| \equiv U_U \overset{SK}{\longleftrightarrow} D_k$
Goal 2 $U_U| \equiv D_k| \equiv D_k \overset{SK}{\longleftrightarrow} U_U$
Goal 3 $D_k| \equiv D_k \overset{SK}{\longleftrightarrow} U_U$
Goal 4 $D_k| \equiv U_U| \equiv U_U \overset{SK}{\longleftrightarrow} D_k$

Proof By using *Message 1*, we see that

$$Statement_1 : D_k \triangleleft UR_{U_{Ur_U}}$$

D_k receives UR_U from U_U that is composed of Ur_U and denoted as $UR_{U_{Ur_U}}$

From A_{11}, $Statement_1$ and applying the Message Meaning Rule (**MMR**)

$$\frac{D_k| \equiv D_k \overset{UR_U}{\longleftrightarrow} U_U,\, D_k \triangleleft \langle UR_U\rangle_{Ur_U}}{D_k| \equiv U_U| \sim UR_U}$$

D_k believes UR_U is shared by U_U and sees $\langle UR_U\rangle_{Ur_U}$ is true as per numerator, therefore the denominator part where D_k believes, U_U once said UR_U is true. Hence, $Statement_2$.

$$Statement_2 : D_k| \equiv U_U| \sim UR_U$$

Table 5 Idealized form of messages

Message 1	$U_U \overset{viaCS_l}{\longrightarrow} D_k : T_1, UR_U : \langle UR_U\rangle_{Ur_U.P+K}, M_2 : \langle M_1\rangle_{(UR_U\vert\vert M_U\vert\vert k)}, T_1, \langle CV_l\rangle_{C\alpha_l.P}, \langle UR_U\rangle_{Ur_U.P+K}$
Message 2	$CS_l \longrightarrow D_k : T_2, UR_U : \langle UR_U\rangle_{Ur_U.P+K}, CR_{l2} : \langle CR_{l2}\rangle_{Cr_{l2}.P}, M_4 : \langle M_3\rangle_{(A_k\vert\vert\beta\vert\vert UR_U)}, T_2, \langle MV_k\rangle_{M\alpha_k.P}, \langle CR_{l2}\rangle_{Cr_{l2}.P}$
Message 3	$D_k \overset{viaCS_l}{\longrightarrow} U_U : T_3, MR_k : \langle MR_k\rangle_{Mr_k.P}, M_6 : \langle M_5\rangle_{(B_l\vert\vert MR_k\vert\vert\beta)}, T_3, \langle CV_l\rangle_{C\alpha_l.P}, \langle MR_k\rangle_{Mr_k.P}$
Message 4	$CS_l \longrightarrow U_U : T_4, MR_k : \langle MR_k\rangle_{Mr_k.P}, CR_{l3} : \langle CR_{l3}\rangle_{Cr_{l3}.P}, M_8 : \langle M_7\rangle_{(M_U\vert\vert k\vert\vert MR_k)}, T_4, \langle UV_U\rangle_{U\alpha_U.P}, \langle CR_{l3}\rangle_{Cr_{l3}.P}$

Table 6 Assumptions

$A_1 : U_U \mid\equiv \#\{T_1\}$	$A_7 : U_U \mid\equiv U_U \xleftrightarrow{k} CS_l$	$A_{10} : D_k \mid\equiv D_k \xleftrightarrow{\beta} CS_l$
$A_2 : CS_l \mid\equiv \#\{T_2, T_4\}$	$A_8 : D_k \mid\equiv D_k \xleftrightarrow{k} U_U$	$A_{11} : D_k \mid\equiv D_k \xleftrightarrow{URU} U_U$
$A_3 : D_k \mid\equiv \#\{T_4\}$	$A_9 : CS_l \mid\equiv CS_l \xleftrightarrow{\beta} D_k$	$A_{12} : U_U \mid\equiv U_U \xleftrightarrow{MR_k} D_k$
$A_4 : U_U \mid\equiv \#\{Ur_U, Mr_k, r_{Ul}, Cr_l, Cr_{l1}, Cr_{l2}, Cr_{l3}, UR_U, MR_k, CR_l, CR_{l1}, CR_{l2}, CR_{l3}\}$		
$A_5 : CS_l \mid\equiv \#\{Ur_U, Mr_k, r_{Ul}, Cr_l, Cr_{l1}, Cr_{l2}, Cr_{l3}, UR_U, MR_k, CR_l, CR_{l1}, CR_{l2}, CR_{l3}\}$		
$A_6 : D_k \mid\equiv \#\{Ur_U, Mr_k, r_{Ul}, Cr_l, Cr_{l1}, Cr_{l2}, Cr_{l3}, UR_U, MR_k, CR_l, CR_{l1}, CR_{l2}, CR_{l3}\}$		

From $Statement_2$, A_6 and applying the Nonce Verification Rule (**NVR**)

$$\frac{D_k \mid\equiv \#(UR_U),\ D_k \mid\equiv U_U \mid\sim UR_U}{D_k \mid\equiv U_U \mid\equiv UR_U}$$

D_k believes, UR_U is fresh and U_U once said UR_U is true as per numerator; therefore, the denominator part where D_k believes, U_U believes UR_U is true. Hence, $Statement_3$.

$$Statement_3 : D_k \mid\equiv U_U \mid\equiv UR_U$$

From A_6, $Statement_3$ and applying the Session Key Rule (**SKR**)

$$\frac{D_k \mid\equiv \#(UR_U),\ D_k \mid\equiv U_U \mid\equiv UR_U}{D_k \mid\equiv D_k \xleftrightarrow{SK} U_U}$$

D_k believes, UR_U is fresh and U_U believes UR_U, an essential factor of the session key is true as per numerator; therefore the denominator part where D_k believes, it shares the session key SK with U_U is true. Hence, $Statement_4$.

$$Statement_4 : D_k \mid\equiv D_k \xleftrightarrow{SK} U_U$$

$SK = Mr_k.UV_U + M\alpha_k.UR_U$ *Goal 3*

By using *Message 3*, we see that

$$Statement_5 : U_U \triangleleft MR_{k_{Mr_k}}$$

U_U receives MR_k from D_k that is composed of Mr_k and denoted as $MR_{k_{Mr_k}}$

From A_{12}, $Statement_5$ and applying the Message Meaning Rule (**MMR**)

$$\frac{U_U \mid\equiv U_U \xleftrightarrow{MR_k} D_k,\ U_U \triangleleft \langle MR_k\rangle_{MR_k}}{U_U \mid\equiv D_k \mid\sim MR_k}$$

U_U believes MR_k is shared by D_k and sees $\langle MR_k\rangle_{Mr_k}$ is true as per numerator; therefore, the denominator part where U_U believes, D_k once said MR_k is true. Hence, $Statement_6$.

$$Statement_6 : U_U \mid\equiv D_k \mid\sim MR_k$$

From A_4, $Statement_6$ and applying the Nonce Verification Rule (**NVR**)

$$\frac{U_U \mid\equiv \#(MR_k),\ U_U \mid\equiv D_k \mid\sim MR_k}{U_U \mid\equiv D_k \mid\equiv MR_k}$$

U_U believes, MR_k is fresh and D_k once said MR_k is true as per numerator; therefore, the denominator part where U_U believes, D_k believes MR_k is true. Hence, $Statement_7$.

$$Statement_7 : U_U \mid\equiv D_k \mid\equiv MR_k$$

From A_4, $Statement_7$ and applying the Session Key Rule (**SKR**)

$$\frac{U_U \mid\equiv \#(MR_k),\ U_U \mid\equiv D_k \mid\equiv MR_k}{U_U \mid\equiv U_U \xleftrightarrow{SK} D_k}$$

U_U believes, MR_K is fresh and D_k believes MR_k, an essential factor of the session key is true as per numerator; therefore, the denominator part where U_U believes, it shares the session key SK with D_k is true. Hence, $Statement_8$.

$$Statement_8 : U_U \mid\equiv U_U \xleftrightarrow{SK} D_k$$

$SK = Ur_U.MV_k + U\alpha_U.MR_k$ *Goal 1*

From combining $Statement_7$: $U_U \mid\equiv D_k$ and $Statement_4$: $D_k \mid\equiv D_k \xleftrightarrow{SK} U_U$, we arrive at $Statement_9$ just like $P \mid\equiv Q$ and $Q \mid\equiv R$ results to $P \mid\equiv Q \mid\equiv R$ which is if P believes Q and Q believes R, then P would believe Q which in turn believes R (just like chain of trust).

$$Statement_9 : U_U \mid\equiv D_k \mid\equiv D_k \xleftrightarrow{SK} U_U$$

$SK = Mr_k.UV_U + M\alpha_k.UR_U$ *Goal 2*

Similarly, from combining $Statement_3$: $D_k \mid\equiv U_U$ and $Statement_8$: $U_U \mid\equiv U_U \xleftrightarrow{SK} D_k$ we arrive at $Statement_9$ just like $P \mid\equiv Q$ and $Q \mid\equiv R$ results to $P \mid\equiv Q \mid\equiv R$ which is if P believes Q and Q believes R, then P would believe Q which in turn believes R (just like chain of trust).

$$Statement_{10} : D_k \mid\equiv U_U \mid\equiv U_U \xleftrightarrow{SK} D_k$$

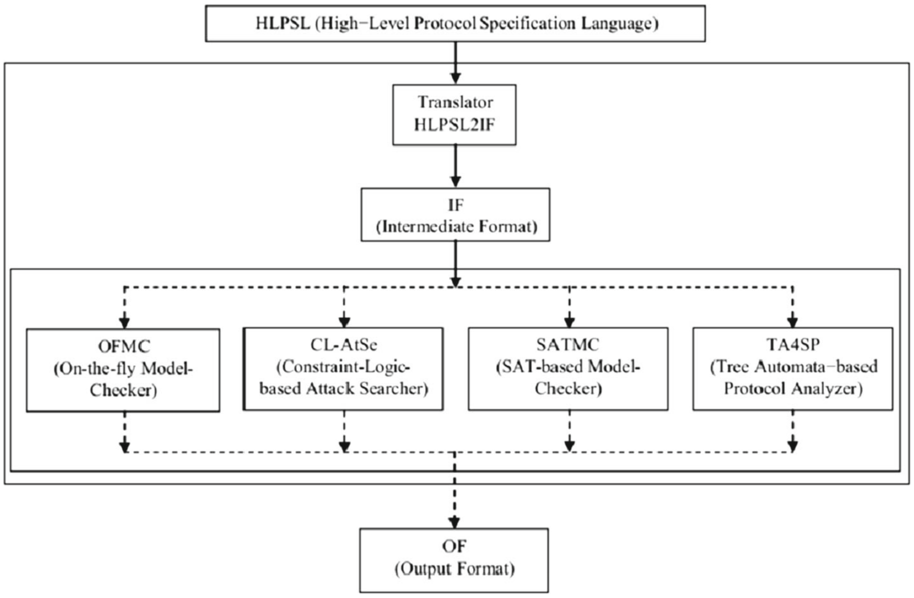

HLPSL (High-Level Protocol Specification Language)
Translator HLPSL2IF
IF (Intermediate Format)
OFMC (On-the-fly Model-Checker)
CL-AtSe (Constraint-Logic-based Attack Searcher)
SATMC (SAT-based Model-Checker)
TA4SP (Tree Automata-based Protocol Analyzer)
OF (Output Format)

Fig. 16 Architecture of AVISPA

$SK = Ur_U.MV_k + U\alpha_U.MR_k$ *Goal 4*

6.3 Formal Security Analysis: AVISPA simulation

AVISPA is the extensively used simulation tool for formally verifying the strength of the authentication protocols [13]. It checks whether the proposed protocol is SAFE or UNSAFE against various existing security threats. The architecture of AVISPA is shown in Fig. 16. High Level Protocol Specification Language (HLPSL) is used for simulating the protocol which is then converted via HLPSL2IF translator into a low-level code, called intermediate format (IF). Each communicating entity is executed simultaneously as a separate role where the role user, device, and server show the operations carried out by the corresponding communicating entities. Role session models them as in form of different unbounded sessions, and Role environment briefs about intruder knowledge while modeling the attacker parallelly with other communicating entities. The HLPSL code for communicating entities is given in Figs. 17, 18, 19 and 20.

Authentication properties:

1. authentication_on user_server_m2: CS_l receives M_2 from U_U and validates it on basis of M_2.
2. authentication_on server_device_m4: D_k receives M_4 from CS_l and validates it by using M_4.
3. authentication_on device_server_m6: CS_l receives M_6 from D_k and validates it on basis of M_6.
4. authentication_on server_user_m8: U_U receives M_8 from CS_l and validates it by using M_8.
5. authentication_on user_server_Au: CS_l receives A_U from U_U and verifies it on basis of A_U.
6. authentication_on server_user_Al: U_U receives A_l from CS_l and verifies it by using A_l.
7. authentication_on device_server_Bk: CS_l receives B_k from D_k and verifies it on basis of B_k.
8. authentication_on server_device_Bl: D_k receives B_l from CS_l and verifies it by using B_l.

Secrecy goals:

1. secrecy_of subs1: password PW_U and biometric BIO_U are only known to U_i.
2. secrecy_of subs2: the symmetric key k is known to U_U and CS_l.
3. secrecy_of subs3: the value of private key $U\alpha_U$ is only known to U_U.
4. secrecy_of subs4: the value of secret key $C\alpha_l$ is only known to CS_l.
5. secrecy_of subs5: the value of symmetric key β is known to CS_l and D_k.
6. secrecy_of subs6: the value of secret-key $M\alpha_k$ is known only to D_k.

Result of HLPSL code is shown in Fig. 21 that gives the results for the backends OFMC and CL-AtSe assuring that our protocol is secure and it withstands all fore-known attacks.

7 Performance Analysis

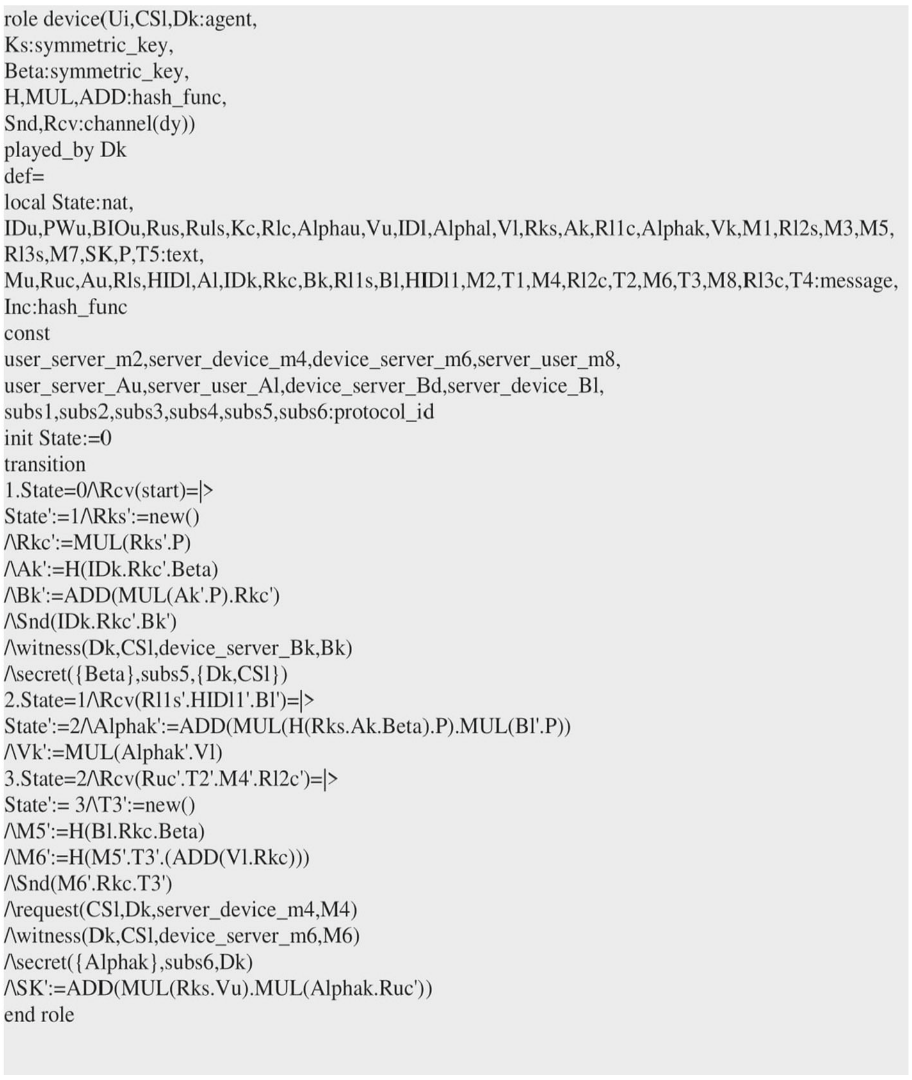
In this section, a comparative performance analysis of our protocol with the existing protocols [2,3,6,11] is presented.

```
role user(Ui,CSl,Dk:agent,
Ks:symmetric_key,
Beta:symmetric_key,
H,MUL,ADD:hash_func,
Snd,Rcv:channel(dy))
played_by Ui
def=
local State:nat,
IDu,PWu,BIOu,Rus,Ruls,Kc,Rlc,Alphau,Vu,IDl,Alphal,Vl,Rks,Ak,Rl1c,Alphak,Vk,M1,Rl2s,M3,M5,
Rl3s,M7,SK,P,T5:text,
Mu,Ruc,Au,Rls,HIDl,Al,IDk,Rkc,Bk,Rl1s,Bl,HIDl1,M2,T1,M4,Rl2c,T2,M6,T3,M8,Rl3c,T4:message,
Inc:hash_func
const
user_server_m2,server_device_m4,device_server_m6,server_user_m8,
user_server_Au,server_user_Al,device_server_Bk,server_device_Bl,
subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
init State:=0
transition
1.State=0/\Rcv(start)=|>
State':=1/\BIOu':=new()
/\PWu':=new()
/\Mu':=H(IDu.PWu'.BIOu')
/\Rus':=new()
/\Kc':=MUL(Ks.P)
/\Ruc':=ADD((MUL(Rus'.P)).Kc')
/\Au':=H(Mu'.Ruc'.Kc')
/\Snd({Ruc'.Au'.Mu'}_Ks)
/\witness(Ui,FSj,user_server_Au,Au)
/\secret({PWu',BIOu'},subs1,Ui)
/\secret({Ks},subs2,{Ui,CSl})
2.State=1/\Rcv(Rls'.HIDl'.Al')=|>
State':=2/\Ruls':=new()
/\Alphau':=ADD(MUL(H(Ruls'.Mu.Kc).P).MUL(Al'.P))
/\Vu':=MUL(Alphau'.Vl)
/\T1':=new()
/\M1':=H(Ruc.Ks.Mu)
/\M2':=H(M1'.T1'.(ADD(Vl.Ruc)))
/\Snd(M2'.Ruc.T1')
/\witness(Ui,CSl,user_server_m2,M2)
/\request(CSl,Ui,server_user_Al,Al)
/\secret({Alphau'},subs3,Ui)
3.State=2/\Rcv(M8'.Rl3c'.T4'.Rkc')=|>
State':=3/\T5':=new()
/\SK':=ADD(MUL(Rus.Vk).MUL(Alphau.Rkc'))
/\request(CSl,Ui,server_user_m8,M8)
end role
```

Fig. 17 Role User

```
role server(Ui,CSl,Dk:agent,
Ks:symmetric_key,
Beta:symmetric_key,
H,MUL,ADD:hash_func,
Snd,Rcv:channel(dy))
played_by CSl
def=
local State:nat,
IDu,PWu,BIOu,Rus,Ruls,Kc,Rlc,Alphau,Vu,IDl,Alphal,Vl,Rks,Ak,Rl1c,Alphak,Vk,M1,Rl2s,M3,M5,
Rl3s,M7,SK,P,T5:text,
Mu,Ruc,Au,Rls,HIDl,Al,IDk,Rkc,Bk,Rl1s,Bl,HIDl1,M2,T1,M4,Rl2c,T2,M6,T3,M8,Rl3c,T4:message,
Inc:hash_func
const
user_server_m2,server_device_m4,device_server_m6,server_user_m8,
user_server_Au,server_user_Al,device_server_Bk,server_device_Bl,
subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
init State:=0
transition
1.State=0/\Rcv({Rus'.Au'.Mu'}_Ks)=|>
State':=1/\Alphal':=new()
/\Vl':=MUL(Alphal'.P)
/\Kc':=MUL(Ks.P)
/\Rls':=new()
/\HIDl':=MUL(IDl.Kc')
/\Rlc':=ADD((MUL(Rls'.P)).Kc')
/\Al':=H(Mu'.HIDl'.Kc'.Rlc')
/\Snd(Rls'.HIDl'.Al')
/\secret({Alphal'},subs4,CSl)
/\request(Ui,CSl,user_server_Au,Au)
/\witness(CSl,Ui,server_user_Al,Al)
2.State=1/\Rcv(IDk.Rkc'.Bk')=|>
State':=2/\Rl1s':=new()
/\HIDl':=MUL(IDl.Beta)
/\Rl1c':=ADD(MUL(Rl1s'.P).MUL(Beta.P))
/\Bl':=H(Bk'.HIDl'.Beta.Rl1c')
/\request(Dk,CSl,device_server_Bk,Bk)
3.State=2/\Rcv(M2'.Ruc'.T1')=|>
State':=3/\T2':=new()
/\Rl2s':=new()
/\Rl2c':=MUL(Rl2s'.P)
/\M3':=H(Bk.Beta.Ruc')
/\M4':=H(M3'.T2'.(ADD(Vk.Rl2c')))
/\Snd(M4'.Rl2c'.Ruc'.T2')
/\request(Ui,CSl,user_server_m2,M2)
/\witness(CSl,Dk,server_device_m4,M4)
4.State=3/\Rcv(M6'.Rkc'.T3')=|>
State':=4/\T4':=new()
/\Rl3s':=new()
/\Rl3c':=MUL(Rl3s'.P)
/\M7':=H(Mu.Ks.Rkc')
/\M8':=H(M7'.T4'.(ADD(Vu.Rl3c')))
/\Snd(M8'.Rkc'.Rl3c'.T4')
/\request(Dk,CSl,device_server_m6,M6)
/\witness(CSl,Ui,server_user_m8,M8)
end role
```

Fig. 18 Role Server



```
role device(Ui,CSl,Dk:agent,
Ks:symmetric_key,
Beta:symmetric_key,
H,MUL,ADD:hash_func,
Snd,Rcv:channel(dy))
played_by Dk
def=
local State:nat,
IDu,PWu,BIOu,Rus,Ruls,Kc,Rlc,Alphau,Vu,IDl,Alphal,Vl,Rks,Ak,Rl1c,Alphak,Vk,M1,Rl2s,M3,M5,
Rl3s,M7,SK,P,T5:text,
Mu,Ruc,Au,Rls,HIDl,Al,IDk,Rkc,Bk,Rl1s,Bl,HIDl1,M2,T1,M4,Rl2c,T2,M6,T3,M8,Rl3c,T4:message,
Inc:hash_func
const
user_server_m2,server_device_m4,device_server_m6,server_user_m8,
user_server_Au,server_user_Al,device_server_Bd,server_device_Bl,
subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
init State:=0
transition
1.State=0/\Rcv(start)=|>
State':=1/\Rks':=new()
/\Rkc':=MUL(Rks'.P)
/\Ak':=H(IDk.Rkc'.Beta)
/\Bk':=ADD(MUL(Ak'.P).Rkc')
/\Snd(IDk.Rkc'.Bk')
/\witness(Dk,CSl,device_server_Bk,Bk)
/\secret({Beta},subs5,{Dk,CSl})
2.State=1/\Rcv(Rl1s'.HIDl1'.Bl')=|>
State':=2/\Alphak':=ADD(MUL(H(Rks.Ak.Beta).P).MUL(Bl'.P))
/\Vk':=MUL(Alphak'.Vl)
3.State=2/\Rcv(Ruc'.T2'.M4'.Rl2c')=|>
State':= 3/\T3':=new()
/\M5':=H(Bl.Rkc.Beta)
/\M6':=H(M5'.T3'.(ADD(Vl.Rkc)))
/\Snd(M6'.Rkc.T3')
/\request(CSl,Dk,server_device_m4,M4)
/\witness(Dk,CSl,device_server_m6,M6)
/\secret({Alphak},subs6,Dk)
/\SK':=ADD(MUL(Rks.Vu).MUL(Alphak.Ruc'))
end role
```

Fig. 19 Role Device

Here, we show that our protocol incurs less overheads as compared to the existing protocols. We consider log-in and authentication phases only for performance analysis as these phases are executed more often. The primitive factors for performance evaluation are communication, computation and storage overhead. All the comparative analysis is tabulated in Table 7, where Comp and Com represent Computational and Communication overheads, respectively.

7.1 Computation Overhead

Table 7 represents the comparison of computational overheads in context of completion time of the protocols in milliseconds (ms). The primitive operations required for protocol execution are symbolized as T_S, T_H, T_{FE}, T_{ECPA}, T_{ECPM}, T_{EXP}, T_{PKE} and T_{PKD}, which denote the execution time for symmetric-key encryption/decryption, hash function, fuzzy extractor, ECC-based point addition, ECC-based scalar point multiplication, exponential, public key encryption and public key decryption, respectively. These operations require 3.85ms, 0.0046ms, 2.226ms, 0.004ms, 2.226ms, 231ms, 385ms and 385ms to operate, respectively. Here, the time consumption of operations is taken from [25]. Though the existing protocols require less computational overheads in context of execution time, yet they suffer from numerous security flaws. Our protocol maintains a tradeoff between computational overhead and security strength.

7.2 Communication Overhead

Table 7 presents the comparative analysis of the communication overhead for every communicating party of the system, i.e., user, server and mobile device with respect to the length of transmitted and received messages in bits for

```
role session(Ui,CSl,Dk:agent,
Ks:symmetric_key,
Beta:symmetric_key,
H,MUL,ADD:hash_func)
def=
local SI,SJ,RI,RJ,TI,TJ:channel(dy)
composition
user(Ui,CSl,Dk,ks,beta,H,MUL,ADD,SI,RI)
/\server(Ui,CSl,Dk,ks,beta,H,MUL,ADD,SJ,RJ)
/\device(Ui,CSl,Dk,ks,beta,H,MUL,ADD,TI,TJ)
end role
role environment()
def=
const ui,csl,dk:agent,
ks:symmetric_key,
beta:symmetric_key,
h,mul,add:hash_func,
idu,pwu,biou,rus,ruls,kc,rlc,alphau,vu,idl,alphal,vl,rks,ak,rl1c,alphak,vk,ru1s,m1,rl2s,m3,m5,rl3s,m7,sk
,p,t5,m2,m4,m6,m8,ru1c,t1,rl2c,t2,t3,rl3c,ruc,rkc:text,
user_server_m2,server_device_m4,device_server_m6,server_user_m8,
user_server_Au,server_user_Al,device_server_Bk,server_device_Bl,
subs1,subs2,subs3,subs4,subs5,subs6:protocol_id
intruder_knowledge={ui,csl,dk,h,mul,add,m2,m4,m6,m8,ruc,t1,rl2c,t2,rkc,t3,rl3c}
composition
session(ui,csl,dk,ks,beta,h,mul,add)
/\session(ui,csl,dk,ks,beta,h,mul,add)
/\session(ui,csl,dk,ks,beta,h,mul,add)
end role
goal
secrecy_of subs1
secrecy_of subs2
secrecy_of subs3
secrecy_of subs4
secrecy_of subs5
secrecy_of subs6
authentication_on user_server_m2
authentication_on server_device_m4
authentication_on device_server_m6
authentication_on server_user_m8
authentication_on user_server_Au
authentication_on server_user_Al
authentication_on device_server_Bk
authentication_on server_device_Bl
end goal
environment()
```

Fig. 20 Role Session and Environment

```
% OFMC
% Version of 2006/20/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Fog_17_1_2020.if
GOAL
 as_specified
BACKEND
 OFMC
```

(a) OFMC Result

```
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL
PROTOCOL
 /home/span/span/testsuite/results/Fog_17_1_2020.if
GOAL
 As Specified
BACKEND
 CL-AtSe
```

(b) CL-AtSe Result

Fig. 21 Simulation results

Table 7 Performance Evaluation

Schemes			Ref[11]	Ref[2]	Ref[3]	Ref[6]	Ours
Communicating Entities	User	Comp	$16T_H + 2T_{ECPM} + T_{FE} = 6.7516ms$	$3T_H + 4T_{ECPM} = 8.9178ms$	$7T_H + 2T_{ECPM} = 4.4842ms$	-	$5T_H + 2T_{ECPM} + 3T_{ECPA} = 4.487ms$
		Com	1824	2688	4160	-	864
	Server	Comp	$10T_H + 3T_{ECPM} = 6.724ms$	$4T_H + 6T_{ECPM} + 4T_{ECPA} = 13.3904ms$	$7T_H + 4T_S + 3T_{ECPM} = 22.1102ms$	$4T_{EXP} + 3T_{PKE} + 3T_{PKD} = 3234ms$	$8T_H + 2T_{ECPM} + 4T_{ECPA} = 4.5048ms$
		Com	1984	2688	4160	14336	1728
	Device	Comp	$9T_H = 0.0414ms$	-	-	-	$4T_H + 2T_{ECPM} + 3T_{ECPA} = 4.4824ms$
		Com	1824	-	-	-	864
Total Execution Time (ms)			13.517	22.3082	26.5944	3234	13.4742
Storage Cost (bits)			1928	2720	1504	-	160

[2,3,6,11]. Here, we consider the length of password, identity (user, server) and random nonce as 128 bits each, whereas the timestamp and identity of mobile device are of 32 bits each. The symmetric-key encrypted message and value after cryptographic hash operation are 128 and 160 bits long, respectively, where SHA-1 algorithm has been used for hashing. Table 7 clearly shows that the proposed protocol has the lowest communication overhead, which is one of the main concerns for authentication protocols of a cloud computing network. Hence, this reduced overhead makes its applicable to the real-time applications.

7.3 Smart Device Storage Cost

The storage cost of our protocol is considerably less than that of the protocols [2,3,6,11], which could be clearly seen in Table 7. As the proposed protocol does not store much information about the communicated messages, the storage overhead is considerably less. Thus, the proposed protocol ensures the increased life time of the smart devices by reducing the storage utilization and making it useful in resource constrained environment.

In summary, our protocol is more proficient in the context of the communication, computation and storage costs.

8 Conclusion

In this research paper, we have scrutinized the Wazid et al.'s protocol and shown that it is vulnerable to denial of service attack, privileged insider attack, and stolen smart-card attack, and have discussed a robust protocol for authentication and key negotiation in cloud computing environment by overcoming the above-mentioned drawbacks. The proposed protocol is lightweight due to its use of ECC and irreversible hash functions. Further, it maintains a hard to achieve tradeoff between the security and performance. The state-of-the-art formal and informal security analysis using BAN logic and AVISPA show that our protocol is resilient to all malicious attacks. The comparative performance analysis depicts that our protocol outshines the other similar protocols. Further, its simplicity makes it easily implementable in practical scenarios.

References

1. Li, H.; Dai, Y.; Tian, L.; Yang, H.: Identity-based authentication for cloud computing. In: IEEE international conference on cloud computing, pp. 157–166, Springer, Berlin (2009)
2. Sun, H.; Wen, Q.; Zhang, H.; Jin, Z.: A novel remote user authentication and key agreement scheme for mobile client-server environment. Appl. Math. Inf. Sci. **7**(4), 1365 (2013)
3. Li, H.; Li, F.; Song, C.; Yan, Y.: Towards smart card based mutual authentication schemes in cloud computing. TIIS **9**(7), 2719–2735 (2015)
4. Chen, N.; Jiang, R.: Security analysis and improvement of user authentication framework for cloud computing. J. Netw. **9**(1), 198 (2014)
5. Wazid, M.; Das, A.K.; Kumari, S.; Li, X.; Wu, F.: Provably secure biometric-based user authentication and key agreement scheme in cloud computing. Secur. Commun. Netw. **9**(17), 4103–4119 (2016)
6. Hu, P.; Dhelim, S.; Ning, H.; Qiu, T.: Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl. **98**, 27–42 (2017)
7. Alrawais, A.; Alhothaily, A.; Hu, C.; Xing, X.; Cheng, X.: An attribute-based encryption scheme to secure fog communications. IEEE Access **5**, 9131–9138 (2017)
8. Mukherjee, M.; Matam, R.; Shu, L.; Maglaras, L.; Ferrag, M.A.; Choudhury, N.; Kumar, V.: Security and privacy in fog computing: Challenges. IEEE Access **5**, 19293–19304 (2017)
9. Koo, D.; Hur, J.: Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing. Future Gener. Comput. Syst. **78**, 739–752 (2018)

10. Wang, H.; Wang, Z.; Domingo-Ferrer, J.: Anonymous and secure aggregation scheme in fog-based public cloud computing. Future Gener. Comput. Syst. **78**, 712–719 (2018)
11. Wazid, M.; Das, A.K.; Kumar, N.; Vasilakos, A.V.: Design of secure key management and user authentication scheme for fog computing services. Future Gener. Comput. Syst. **91**, 475–492 (2019)
12. Chandrakar, P.; Om, H.: A secure and privacy preserving remote user authentication protocol for internet of things environment. In: International conference on computational intelligence, communications, and business analytics, pp. 537–551, Springer, Berlin (2017)
13. Armando, A.; Basin, D.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuéllar, J.; Drielsma, P. H.; Héam, P.-C.; Kouchnarenko, O.; Mantovani, J. *et al.*: The avispa tool for the automated validation of internet security protocols and applications. In: International conference on computer aided verification, pp. 281–285, Springer, Berlin (2005)
14. Kumar, A.; Om, H.: Lightweight, ecc based rfid authentication scheme for wlan. Int. J. Bus. Data Commun. Netw. (IJBDCN) **12**(2), 89–103 (2016)
15. Stallings, W.: Cryptogr. Netw. Secur. Pearson Education, India (2006)
16. Paar, C.; Pelzl, J.: Understanding cryptography: a textbook for students and practitioners. Springer Science and Business Media, Berlin (2009)
17. Ray, S.; Biswas, G.: Establishment of ecc-based initial secrecy usable for ike implementation. In: Proceedings of the world congress on engineering, vol. 1, (2012).
18. Ku, W.-C.; Chang, S.-T.: Impersonation attack on a dynamic id-based remote user authentication scheme using smart cards. IEICE Trans. Commun. **88**(5), 2165–2167 (2005)
19. Wu, Z.; Gao, S.; Cling, E. S.; Li, H.: A study on replay attack and anti-spoofing for text-dependent speaker verification. In: Signal and information processing association annual summit and conference (APSIPA), 2014 Asia-Pacific, pp. 1–5, IEEE, (2014)
20. Liu, H.: A new form of dos attack in a cloud and its avoidance mechanism. In: Proceedings of the 2010 ACM workshop on Cloud computing security workshop, pp. 65–76, (2010)
21. Kumar, V.; Kumar, R.; Pandey, S.: Polynomial based non-interactive session key computation protocol for secure communication in dynamic groups. Int. J. Inf. Technol. **12**(1), 283–288 (2020)
22. Sarvabhatla, M.; Reddy, M. C. M.; Vorugunti, C. S.: A robust remote user authentication scheme resistant to known session specific temporary information attack. In: 2015 Applications and innovations in mobile computing (AIMoC), pp. 164–169, IEEE, (2015)
23. Salem, M. B.; Hershkop, S.; Stolfo, S. J.: A survey of insider attack detection research. In: Insider attack and cyber security. pp. 69–90, Springer, Berlin (2008)
24. Alsalhi, I. N., Albermany, S. A.: Authentication of crns by using ban logic
25. Kilinc, H.H.; Yanik, T.: A survey of sip authentication and key agreement schemes. IEEE Commun. Surv. Tutor. **16**(2), 1005–1023 (2013)