Journal of Systems Architecture 123 (2022) 102370

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

BUAKA-CS: Blockchain-enabled user authentication and key agreement scheme for crowdsourcing system

Mohammad Wazid[a], Ashok Kumar Das[b,*], Rasheed Hussain[c], Neeraj Kumar[d], Sandip Roy[e]

[a] Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248002, India
[b] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India
[c] Networks and Blockchain Lab, Innopolis University, Innopolis 420500, Russia
[d] Department of Computer Science and Engineering, Thapar University, Patiala 147 004, India
[e] Department of Computer Science and Engineering, Asansol Engineering College, Asansol 713305, India

ARTICLE INFO

Keywords:
Crowdsourcing
Blockchain
Security
User authentication and key agreement
Simulation

ABSTRACT

Crowdsourcing is a practice of using collective intelligence of a group in order to achieve a common goal to solve complex problems in an innovative way. It involves obtaining information and opinions from a group of participants who submit their data (i.e., solutions) via the Internet using some applications. The application domains, where crowdsourcing can be used, include, but not limited to, healthcare, environment, public safety, disaster management, and transportation. Despite the unprecedented advantages of crowdsourcing, security and privacy are rising concerns that need to be addressed. Therefore, it is crucially important to provide effective solutions that address the security and privacy issues in crowdsourcing systems. To this end, the salient features of blockchain technology such as immutability, decentralization, transparency and resiliency can play a pivotal role to address the afore-mentioned security challenges. To fill these gaps, in this paper, we propose a new blockchain-based user authentication and key agreement scheme for crowdsourcing (BUAKA-CS) through lightweight cryptographic techniques. The security of the BUAKA-CS is proved through the formal method and also through other mathematical methods that depict the resilience of BUAKA-CS against various types of possible attacks. Moreover, the robustness of BUAKA-CS against possible attacks is proved through widely-recognized automated software validation tools. We also compare BUAKA-CS with other existing schemes and prove its out performance in terms of security, functionality, computation and communication costs. Finally, we conduct the extensive blockchain-based simulations to measure the impact of BUAKA-CS on the performance of the system.

1. Introduction

Crowdsourcing has gained a lot of interest and adoption since its inception in 2006 by Jeff Howe [1]. It is the practice of utilization of wisdom of a group to achieve a common goal. Crowdsourcing has proved its worth in solving complex problems in a distributed way where it leverages an open call for solutions. Thus, it allows the organizations to farm out work to the people anywhere geographically which enables them to utilize tremendous skills and expertise without increasing their expenditures [2]. To date, many firms choose crowdsourcing as a problem-solving technique in different domains such as web development, genealogy, energy science, geography, seismology, healthcare, environment, public safety, disaster management and transportation, and so on [3]. In real-world scenarios, there are some famous crowdsourcing campaigns such as McDonald's "My Burger", "Amazon Studio", Greenpeace's "Save The Arctic", "Airbnb Shorts", etc. [4]. A crowdsourcing system necessitates obtaining of work, information, and opinions from a group of participants who present their data (i.e., views) through the Internet using some applications (such as smartphone application). Participation in crowdsourcing can be either volunteering or paid. For instance, a traffic application may inspire a driver to report accidents and other roadside environmental conditions in order to provide a fresh real-time information to the other application users [2,5,6]. The use of latest computation and communication technologies enable the citizens to both contribute to and use the services provided by crowdsourcing.

* Corresponding author.
E-mail addresses: wazidkec2005@gmail.com (M. Wazid), ashok.das@iiit.ac.in (A.K. Das), r.hussain@innopolis.ru (R. Hussain), neeraj.kumar@thapar.edu (N. Kumar), sandiproy9500@gmail.com (S. Roy).

https://doi.org/10.1016/j.sysarc.2021.102370
Received 8 June 2021; Received in revised form 22 October 2021; Accepted 30 November 2021
Available online 20 December 2021
1383-7621/© 2021 Elsevier B.V. All rights reserved.

Some of the successful examples of crowdsourcing tools for disaster management include “Ushahidi”[1], and “OpenStreetMap”[2], that proved to be very effective tools for relief efforts during the disaster times. The real-time mapping of some environmental phenomena such as flood, the damage may be measured with the help of both crowdsourced information and satellite imagery. Furthermore, crowdsourcing can also be very useful in the field of healthcare and monitoring. Healthcare can greatly benefit from crowdsourcing because of the active and passive participation of different types of people. The information related to “health monitoring”, “disease protection”, “vaccination”, and “breakout of epidemic” can be made available to everybody easily. In November 2008, “Google Flu Trends” mapped the outbreak of influenza with the help of the Google search data of people and it could predict the spreading of the disease along with the response of the people in the real time. Apart from that, CrowdMed is an online platform that integrates different information such as the “ideas of patients”, “practitioners”, and “general crowd to diagnose” and treat different types of diseases [3,7]. Similarly crowdsourcing-based healthcare platform can also be useful to deal with the pandemic of “COVID’19”, which can integrate the views of patients, healthcare staff, and other people from society for the prediction of number of cases, their diagnosis, and treatment. Despite their advantages, crowdsourcing platforms have some security and privacy issues [8]. The crowdsourcing activities are carried out through Internet and all participating entities (i.e., requesters, workers, servers) of crowdsourcing are connected to the Internet. Therefore, it is crucially important to make sure that these activities are carried out in a secure and privacy-aware manner. In the absence of security measures, these activities can be easily manipulated and exploited by the cyber-attackers. Different types of attacks such as “replay”, “man-in-the-middle”, impersonation, privileged insider, illegal password guessing, unauthorized session key computation, etc., might be possible depending on the underlying communication paradigm. Such attacks will have dire consequences and can reveal the sensitive data of the system to the unauthorized parties as well as the attackers might be able to control the system and cause other damage to the system such as Denial of Service (DoS) [2,6,9,10].

1.1. Motivation

As the requester and workers of crowdsourcing system communicate through an insecure channel (i.e., Internet), there are the chances that the exchanged information may be leaked, altered or delayed by the attackers. Various information security related attacks (i.e., “replay, man-in-the-middle, impersonation, privileged insider, illegal credentials guessing, unauthorized session key computation, stolen verifier, denial of service (DoS), etc.”,) are possible during the communication in the crowdsourcing system.

Due to the existing vulnerabilities in a crowdsourcing system, the sensitive data may be leaked or it may cause other severe threats. Therefore, it becomes very essential to provide an effective solution to resolve the security and privacy issues of the crowdsourcing system. We can restrict such kind of incidents through the use of strong authentication mechanism. Moreover, blockchain can also play an important role as it is an tamper-proof technology which can prevent most of the severe attacks and make the system more robust and decentralized. Furthermore, there are no such approaches in the conducted literature survey for the crowdsourcing system. Hence, it is desirable to provide a robust blockchain based authentication mechanism to protect the communication in a crowdsourcing system.

[1] https://www.ushahidi.com/

[2] https://www.openstreetmap.org/#map=3/69.62/-74.90

1.2. Research contributions

The contributions of proposed work are summarized below.

- A Blockchain based User Authentication and Key Agreement scheme for CrowdSourcing system (BUAKA-CS) is proposed. BUAKA-CS utilizes lightweight cryptographic operations without introducing extra overhead on the system.
- The security of the proposed BUAKA-CS is done with the help of the formal security using the “Real-Or-Random (ROR) model” and also through other non-mathematical method in an informal way. Both the analysis shows that BUAKA-CS can resists various types of attacks needed in a crowdsourcing environment.
- The formal security verification is carried out to prove the security of BUAKA-CS through “Automated Validation of Internet Security Protocols and Applications (AVISPA)” that proves the robustness of BUAKA-CS against the existing attacks.
- BUAKA-CS is also compared with the other related existing schemes which proves its effectiveness over the other schemes as it provides more “security and functionality features” along with low computation and communication costs.
- At the end, a practical demonstration of BUAKA-CS through the blockchain implementation is also provided to observe its impact on the system performance on computational time.

1.3. Organization of the paper

Rest of the paper is organized as follows. A literature review of similar types of schemes is provided in Section 2. The different models i.e., network model and threat model used in the designing of BUAKA-CS are discussed in Section 3. The details of various phases of BUAKA-CS are provided in Section 4. The different types of security analysis of BUAKA-CS are provided in Section 5. The comparisons of BUAKA-CS with the other existing competing schemes are conducted in Section 6. The pragmatic study of BUAKA-CS is done in Section 7. Finally, the work is concluded in Section 8 with some concluding remarks and future work.

2. Literature review

In this section, we discuss the related works which are very close to the presented blockchain based authentication scheme in crowdsourcing.

Li et al. [2] suggested a blockchain-enabled framework for crowdsourcing, called CrowdBC. It is completely distributed in nature where the participants contribute to the requested task independently. It also provides privacy with a low transaction fee.

Wang et al. [11] proposed a distributed budget allocation method with an “agent-based dynamic grouping” technique to realize global differential privacy. They have proved that their proposed DADP could provide differential privacy for the real-time crowd-sourced statistical data published in the un-trusted environment. Moreover, the experiments were conducted on the real-world dataset to demonstrate the usefulness of their proposed DADP.

Kumar et al. [12] suggested a concept that effectively addressed the challenges of crowd-sourcing system. They provided the details of crowd-sourced blind authentication of co-channel transmitters (CBAT). Sun et al. [13] proposed the data collection method, which was based on a heterogeneous two-tier fog architecture. Their proposed scheme is applicable for crowdsensing on the Internet of Vehicles (IoV).

Wang et al. [14] made a survey work on the organization of blockchain networks. Their state-of-art review on consensus schemes was focused on distributed system and the design of incentive mechanism. The methods used for self-organization of nodes in the blockchain networks with the concept of game theory. A survey on various blockchain related applications was provided. In the end, some open issues for protocol designing in blockchain were highlighted.

Wang et al. [15] presented a “privacy-preserving blockchain-based incentive technique” for crowdsensing applications. In their proposed scheme, the authors used a cryptocurrency built on blockchains to provide the incentives in a secure way. Another similar work was carried out by Yang et al. [16] where they proposed a mechanism to preserve privacy in blockchain-enabled crowdsensing system. The proposed solution also increased the success rate of finishing the assigned task in addition to protecting the participants’ privacy. Xu et al. [17] designed a blockchain-enabled privacy protected crowdsourcing technique in mobile environment called as BPCM. The designed mechanism can preserve the participants’ privacy and also maintained the integrity of service provision and request. Dasgupta et al. [18] provided various security issues of blockchain-enabled methods and their applications, and the development trends of blockchain-enabled solutions. Similarly, Tiwari et al. [19] proposed an “authenticated controlled data access and sharing scheme for cloud storage”. They used biometric-based authentication mechanism for the secure data sharing and access.

Later on, Ghaffar et al. [20] identified several vulnerabilities in Tiwari et al.’s scheme [19], which include the “cloud server and mobile user impersonation attacks” as well as it did not provide “user anonymity”. Amin et al. [21] also designed a scheme for authentication and key establishment for the IoT-enabled devices in a “distributed cloud computing environment”. However, their scheme was insecure against mobile device stolen and password guessing attacks and did not provide user anonymity [20]. Furthermore, Mo et al. [22] proposed a provably secure user authentication and key establishment protocol for mobile cloud computing environment. However, it is vulnerable to attacks such as data leakage attack, mobile device stolen attack. Moreover, user anonymity property is not provided in this scheme [20]. Zhou et al. [23] also proposed a user authentication and key establishment protocol for cloud computing.

We note that most of the security protocols for cloud computing environment available in the literature do not provide sufficient security as they are prone to various security attacks. Moreover, so far there no blockchain based authentication scheme for crowdsourcing system. Therefore, to realize secure and viable crowdsourcing system, a blockchain-based authentication is essentially important. To achieve the goal, in this paper, a blockchain based user authentication scheme for the crowdsourcing system (BUAKA-CS) has been proposed.

3. System models

In the following, we explain the network and threat models which are required to explain the design of proposed BUAKA-CS.

3.1. Network model

The network model of blockchain-enabled secure communication environment for crowdsourcing is described in Fig. 1. Our network model consists of requesters who submit their problems along with the requirements to the crowdsourcing system. On the other hand, there are contributors who contribute to solving the problems sent to the crowdsourcing system. After performing the designated task, the contributors submit the results of the task to the crowdsourcing system. We note that the contributor and the requesting nodes are both users in our network model. Apart from these nodes, there are also some cloud servers located at the crowdsourcing platform that receive, process and store the information associated with the requesting and contributing nodes. Furthermore, there is a registration authority that acts as the trusted authority of the network and offers the registration services to other network entities i.e., users and cloud servers. Blockchain a tamper proof technology can help us to deploy more security in crowdsourcing system to prevent the happening of various types of attacks. Therefore, in this paper, we focus on the designing a blockchain-enabled secure

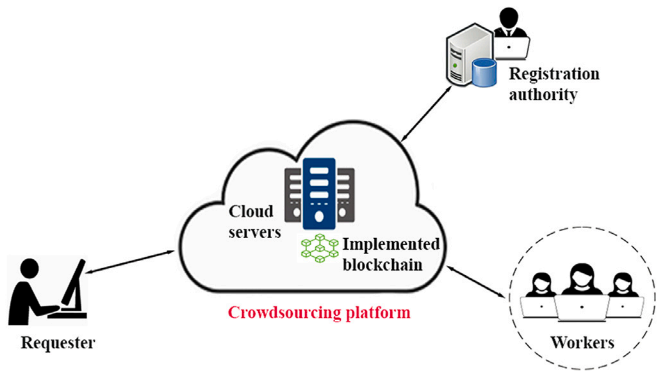


Fig. 1. Network model of blockchain enabled secure communication environment for crowdsourcing.
Source: Adapted from [2,24].

communication scheme for crowdsourcing system. It involves a secure remote user authentication and key agreement mechanism by leveraging blockchain technology. The remote user authentication and key agreement scheme helps to establish the secure session between the user (i.e., requesting node) and the cloud server. Through this secure session, a user can send his/her data to the cloud server in a secure way whereas the cloud servers carry out the processing and storing of the received data. Here, cloud servers also act as the miner nodes which maintain a blockchain of the data of the crowd sourcing system. Thus the data which is maintained through blockchain is secured and cannot be altered in any case.

From the network model discussed in Fig. 1, it is seen that the blockchain is run by the cloud server nodes instead of the distributed user nodes. On the one hand, if the number of cloud servers is too small, an adversary with super computing power is likely to forge a longer fork to tamper with data. On the other hand, if there are a lot of cloud servers, the construction cost of the crowdsourcing platform may be high. In order to make the model practical, we suggest to keep a better trade-off among the number of cloud servers in the system and the construction cost of the crowdsourcing platform. As a result, the right number of cloud servers needs to be chosen accordingly.

3.2. Threat model

The guidelines of widely known “Dolev Yao (DY) threat model” [25] are used in the designing of proposed scheme. According to this model, the communicating parties, i.e., Cloud Server (CS), Requester (U-user of system) and Contributor (U-user of system) communicate over an insecure open channel. In general, these communicating parties are not trustworthy as the exchanged information through the communication channel can be dropped, deleted or updated by the existing adversary $\mathcal{A}$. Moreover, the smartphone of the user can be physically stolen by the $\mathcal{A}$. $\mathcal{A}$ further tries to extract the information stored in the memory of smartphone of the user through sophisticated power analysis attack [26]. Then this extracted information is utilized in other malicious tasks such as illegal password guessing of the user and unauthorized session key computation.

Apart from that, we also consider the guidelines of the “Canetti and Krawczyk’s adversary model, also known as the CK-adversary model” [27] which is the present *de facto* model applied in designing an “authenticated key-agreement security protocol”. According to the CK-adversary model, $\mathcal{A}$ can have all capabilities as in the DY model along with another capability of compromising the secret credentials along with session states (i.e., session keys) corresponding to established sessions. Finally, the registration authority (RA), that offers the registration service to other communicating parties, is assumed to be trusted component involved in the network. Similarly, cloud servers that perform the task of blockchain mining are also considered to be trusted.

Table 1
Syllabaries of BUAKA-CS.

Notation	Its meaning
$\mathcal{A}$	A passive or an active adversary
U_i, SP_{U_i}	ith user and his/her smart phone
ID_{U_i}, RID_{U_i}	U_i's real and pseudo identities, respectively
PW_{U_i}, BIO_{U_i}	U_i's password and biometrics, respectively
RA	Trusted registration authority
ID_{RA}, RID_{RA}	RA's real and pseudo identities, respectively
CS_j	jth cloud server (miner node) in blockchain
ID_{CS_j}, RID_{CS_j}	CS_j's real and pseudo identities, respectively
k_{ra}	160-bit secret key of RA
N	1024-bit random secret of RA
T_1, T_2, T_3	Various timestamps
ΔT	"Maximum transmission delay"
$Gen(\cdot)$	Fuzzy extractor (probabilistic) generation function
$Rep(\cdot)$	Fuzzy extractor (deterministic) reproduction function
σ_{U_i}	U_i's biometric secret key obtained through BIO_{U_i} using $Gen(\cdot)$
τ_{U_i}	U_i's public reproduction parameter obtained through BIO_{U_i} using $Gen(\cdot)$
t	Fuzzy extractor error tolerance threshold value used in deriving σ_{U_i} from noisy biometric BIO'_{U_i} such that the Hamming distance between original BIO_{U_i} and noisy BIO'_{U_i} is less than or equal to t
$h(\cdot)$	"Collision-resistant cryptographic one-way hash function"
$SK_{A,B}$	Session key between two communicating entities A and B
$\|\|$	Data concatenation operator
$\oplus$	Bitwise exclusive-OR (XOR) operator
G	A base point in a non-singular elliptic curve
KR_{CS_j}	Elliptic curve cryptography (ECC)-based private key of CS_j
KU_{CS_j}	ECC-based public key of CS_j; $KU_{CS_j} = KR_{CS_j}.G$
$ECDSA.Sig(\cdot)$	Digital signature generation using Elliptic Curve Digital Signature Algorithm (ECDSA)
$ECDSA.Ver(\cdot)$	Digital signature verification using Elliptic Curve Digital Signature Algorithm (ECDSA)
K_{RA,CS_j}	Symmetric key between RA and CS_j
TC_{CS_j}	Temporal credential of CS_j
RTS_{CS_j}	CS_j's registration timestamp
s	Random secret of U_i
TC_{U_i}	Temporal credential of U_i
RTS_{U_i}	U_i's registration timestamp
TID_{U_i}	Temporary identity of U_i

4. The proposed scheme

In this section, we provide the details of our proposed "Blockchain-enabled User Authentication and Key Agreement Scheme for Crowdsourcing System (BUAKA-CS)". BUAKA-CS contains the following important phases: (a) "pre-deployment and registration phase", (b) "login phase", (c) "authentication and key agreement phase", (d) "blockchain implementation phase", (e) "dynamic device addition phase", (f) "password and biometric update phase", and (g) "smartphone revocation phase". The details of the notations used in this paper are provided in Table 1. Further, all the heterogeneous devices such users (smartphones) and cloud servers are assumed to be synchronized with their clocks, and (mutually) agree on a maximum transmission delay (ΔT) to prevent replay attacks in BUAKA-CS [9]. The detailed descriptions of these phases are provided below.

4.1. *Pre-deployment and registration*

In this phase, the cryptographic parameters and algorithms are decided which will be utilized to perform different tasks in the rest of the phases of BUAKA-CS. The registration of cloud servers and user (i.e., requester and contributor, respectively) can be performed as follows.

4.1.1. *Cloud servers registration*

The registration of cloud servers are performed by the "Registration Authority (RA)" using the following steps.

- **CSR1:** RA picks a long-term secret key k_{ra} and a random secret nonce N and compute the random identity for itself and cloud server CS_j as $RID_{RA} = h(ID_{RA} \| k_{ra})$ and $RID_{CS_j} = h(ID_{CS_j} \| k_{ra})$ where ID_{RA} and ID_{CS_j} are the identities of RA and CS_j.
- **CSR2:** RA further computes temporal credential of CS_j as $TC_{CS_j} = h(N \| RTS_{CS_j} \| k_{ra} \| RID_{RA} \| ID_{CS_j})$ where RTS_{CS_j} is the registration timestamp of CS_j.
- **CSR3:** Finally RA stores $\{RID_{CS_j}, TC_{CS_j}\}$ in secure memory of CS_j. Here it is important to notice that secured memory is tamper-resistant memory which means that adversary $\mathcal{A}$ cannot deduce the values of RID_{CS_j} and TC_{CS_j} using traditional technique.

4.1.2. *User registration*

The registration of users (i.e., requesting node or contributor) are performed by the (RA) using the following steps.

- **UR1:** User U_i picks his/her identity ID_{U_i}, password PW_{U_i} and random secret s to compute pseudo-password as $RPW_{U_i} = h(s \| PW_{U_i})$. Then U_i sends $\{ID_{U_i}, RPW_{U_i}\}$ to RA through a secure channel.
- **UR2:** RA calculates U_i's pseudo identity as $RID_{U_i} = h(ID_{U_i} \| k_{ra})$ and temporal credential as $TC_{U_i} = h(N \| RPW_{U_i} \| RTS_{U_i} \| k_{ra} \| RID_{RA} \| ID_{U_i})$ where RTS_{U_i} denotes U_i's registration timestamp.
- **UR3:** RA again computes $\alpha = RPW_{U_i} \oplus h(ID_{U_i} \| N)$ and a temporary identity of U_i as TID_{U_i}. Then RA sends $\{TID_{U_i}, RID_{U_i}, RID_{RA}, TC_{U_i}, \alpha\}$ to U_i via a secure channel.
- **UR4:** After receiving $\{TID_{U_i}, RID_{U_i}, RID_{RA}, TC_{U_i}, \alpha\}$ from RA, U_i inputs biometric data BIO_{U_i} to the biometric sensor of the smartphone SP_{U_i}. SP_{U_i} then computes $(\sigma_{U_i}, \tau_{U_i}) = Gen(BIO_{U_i})$, where σ_{U_i} and τ_{U_i} are the "biometric secret key of l bits" and "public reproduction parameter". $Gen(\cdot)$ is the fuzzy extractor's probabilistic generation procedure.
- **UR5:** SP_{U_i} of U_i computes $h(ID_{U_i} \| N) = \alpha \oplus RPW_{U_i}$, $\beta = h(h(ID_{U_i} \| N) \| RPW_{U_i})$, $tc_{U_i} = h(TC_{U_i} \| s \| \sigma_{U_i})$ and $\gamma = h(\beta \| tc_{U_i} \| \sigma_{U_i} \| RID_{U_i} \| RID_{RA})$.
- **UR6:** Then SP_{U_i} computes $tc^*_{U_i} = tc_{U_i} \oplus h(ID_{U_i} \| PW_{U_i} \| \sigma_{U_i})$, $s^* = s \oplus h(ID_{U_i} \| \sigma_{U_i} \| PW_{U_i})$, $A_{U_i} = h(ID_{U_i} \| N) \oplus RPW_{U_i}$, $RID^*_{U_i} = RID_{U_i} \oplus h(ID_{U_i} \| \sigma_{U_i} \| PW_{U_i})$, $TID^*_{U_i} = TID_{U_i} \oplus h(ID_{U_i} \| \sigma_{U_i})$ and $RID^*_{RA} = RID_{RA} \oplus h(\sigma_{U_i} \| PW_{U_i})$. Further information such as $\{TID^*_{U_i}, RID^*_{U_i}, RID^*_{RA}, tc^*_{U_i}, A_{U_i}, \gamma, \tau_{U_i}, s^*, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$ are stored in the memory of SP_{U_i}. After the completion of above registration steps of U_i, the sensitive information TC_{U_i}, tc_{U_i}, s, α, β, TID_{U_i}, RID_{U_i}, RID_{RA}, RPW_{U_i}, $h(ID_{U_i} \| N)$ are deleted from the memory of SP_{U_i} to protect against active attacks such as privileged insider attack, unauthorized session key computation, and illegal user's password guessing.
- **UR7:** After the completion of above steps, RA computes $X_{U_i} = h(RID_{RA} \| h(ID_{U_i} \| N))$ and sends the encrypted registration information of U_i, $\{TID_{U_i}, RID_{U_i}, X_{U_i}\}$, where $i = 1, 2, \ldots, n_U$ to CS_j by using the established symmetric key K_{RA,CS_j}. After receiving this information from RA, CS_j decrypts U_i's information with K_{RA,CS_j} and stores the following information $\{RID_{CS_j}, TC_{CS_j}, TID_{U_i}, RID_{U_i}, X_{U_i}\}$ in a secure tamper-resistant database where it will be very hard for the adversary $\mathcal{A}$ to deduce the stored confidential information using active attacks such as stolen verifier.

4.2. Login phase

A user U_i first has to login into the system for accessing the services belonging to the cloud server CS_j. The steps executed to accomplish this phase are provided below.

- **LG1:** U_i supplies his/her identity ID_{U_i} along with password PW'_{U_i}. U_i also imprints biometrics BIO'_{U_i} at the SP_{U_i}'s sensor. SP_{U_i} calculates biometric secret key $\sigma'_{U_i} = Rep(BIO'_{U_i}, \tau'_{U_i})$ with the constraint that the "Hamming distance between the real biometrics BIO_{U_i} provided during the user registration phase and current given BIO'_{U_i} is less than or equal to the error tolerance threshold t".
- **LG2:** SP_{U_i} of U_i again computes $s = s^* \oplus h(ID_{U_i} \| \sigma'_{U_i} \| PW'_{U_i})$, $RPW'_{U_i} = h(s \| PW'_{U_i})$, $tc'_{U_i} = tc^*_{U_i} \oplus h(ID_{U_i} \| PW'_{U_i} \| \sigma'_{U_i})$, $h(ID_{U_i} \| N) = A_{U_i} \oplus RPW'_{U_i}$, $\beta' = h(h(ID_{U_i} \| N) \| RPW'_{U_i})$, $RID'_{U_i} = RID^*_{U_i} \oplus h(ID_{U_i} \| \sigma'_{U_i} \| PW'_{U_i})$ and $RID'_{RA} = RID^*_{RA} \oplus h(\sigma'_{U_i} \| PW'_{U_i})$ and $\gamma' = h(\beta' \| tc'_{U_i} \| \sigma'_{U_i} \| RID'_{U_i} \| RID'_{RA})$. SP_{U_i} checks $\gamma' \overset{?}{=} \gamma$. If it matches then U_i is a genuine user and it will proceed for the "authentication and key agreement phase"; otherwise, the login process will be halted immediately.
- **LG3:** SP_{U_i} further computes $TID_{U_i} = TID^*_{U_i} \oplus h(ID_{U_i} \| \sigma_{U_i})$. Then SP_{U_i} picks the current timestamp T_1 along with a random secret r_1 and computes $M_1 = h(r_1 \| tc_{U_i}) \oplus h(RID_{RA} \| h(ID_{U_i} \| N))$ and $M_2 = h(h(RID_{RA} \| h(ID_{U_i} \| N)) \| T_1 \| h(r_1 \| tc_{U_i}))$. Finally, SP_{U_i} sends the message $Msg_1 = \langle TID_{U_i}, M_1, M_2, T_1\rangle$ as authentication request to the CS_j through open (insecure) channel.

4.3. Authentication and key agreement phase

This phase is required for the "mutual authentication and key agreement between the user (either requesting node or the contributor) U_i and cloud server CS_j". For this purpose, both parties execute the following steps.

- **AKM1:** After receiving $Msg_1 = \langle TID_{U_i}, M_1, M_2, T_1\rangle$ from SP_{U_i}, CS_j first verifies the timeliness of T_1 by checking the condition $|T_1 - T_1^*| \le \Delta T$, where the "maximum transmission delay is represented by ΔT" and T_1^* is reception time of the message Msg_1. If the condition holds, CS_j searches for the same RID_{U_i}, corresponding to received TID_{U_i} in its database and then proceeds for the other steps involved in authentication and key agreement phase.
- **AKM2:** CS_j also obtains the values of $X_{U_i} = h(RID_{RA} \| h(ID_i \| N))$ corresponding to TID_{U_i} from its database. Then CS_j computes $h(r_1 \| tc_{U_i}) = M_1 \oplus X_{U_i}$ and $M'_2 = h(X_{U_i} \| T_1 \| h(r_1 \| tc_{U_i}))$. It then checks the conditions $M'_2 \overset{?}{=} M_2$. If both match, U_i is authenticated with CS_j, otherwise CS_j halts the session with U_i.
- **AKM3:** CS_j generates the current timestamp T_2 along with a random secret r_2 and computes $z_{CS_j} = h(r_2 \| TC_{CS_j})$, $M_3 = z_{CS_j} \oplus h(RID_{U_i} \| h(r_1 \| tc_{U_i}) \| T_1)$, $M_{CS_j} = RID_{CS_j} \oplus h(z_{CS_j} \| T_1 \| T_2)$ and session key $SK_{CS_j,U_i} = h(RID_{U_i} \| h(r_1 \| tc_{U_i}) \| RID_{CS_j} \| X_{U_i} \| z_{CS_j} \| T_1 \| T_2)$, $M_4 = h(SK_{CS_j,U_i} \| T_2)$. CS_j again chooses a new temporary identity of U_i as $TID^{new}_{U_i}$ and computes $M_T = TID^{new}_{U_i} \oplus h(SK_{CS_j,U_i} \| z_{CS_j} \| X_{U_i} \| T_1 \| T_2)$. CS_j sends $Msg_2 = \langle M_3, M_{CS_j}, M_4, M_T, T_2\rangle$ to U_i through open channel.
- **AKM4:** After receiving $Msg_2 = \langle M_3, M_{CS_j}, M_4, M_T, T_2\rangle$ from CS_j, U_i first verifies the timeliness of T_2 through the condition checking $|T_2 - T_2^*| \le \Delta T$, where T_2^* is reception time of the message Msg_2. After that U_i computes $z_{CS_j} = M_3 \oplus h(RID_{U_i} \| h(r_1 \| tc_{U_i}) \| T_1)$, $RID_{CS_j} = M_{CS_j} \oplus h(z_{CS_j} \| T_1 \| T_2)$ and session key $SK_{U_i,CS_j} = h(RID_{U_i} \| h(r_1 \| tc_{U_i}) \| RID_{CS_j} \| h(RID_{RA} \| h(ID_i \| N)) \| z_{CS_j} \| T_1 \| T_2)$ and $M'_4 = h(SK_{U_i,CS_j} \| T_2)$. U_i checks the condition $M'_4 \overset{?}{=} M_4$. If it matches, then CS_j is authenticated with U_i and the computed session key is correct. Furthermore, U_i computes $TID^{new}_{U_i} = M_T \oplus h(SK_{CS_j,U_i} \| z_{CS_j} \| h(RID_{RA} \| h(ID_{U_i} \| N)) \| T_1 \| T_2)$. Then both U_i and CS_j use the temporary identity $TID^{new}_{U_i}$ for their future communication.
- **AKM5:** U_i further picks the current timestamp T_3 and computes $M_5 = h(SK_{U_i,CS_j} \| TID^{new}_{U_i} \| T_3)$ and sends $Msg_3 = \langle M_5, T_3\rangle$ to CS_j through open channel. After receiving $Msg_3 = \langle M_5, T_3\rangle$ from U_i, CS_j, first verifies the timeliness of T_3 through checking the condition $|T_3 - T_3^*| \le \Delta T$, where T_3^* is reception time of the message Msg_3. Then CS_j computes $M'_5 = h(SK_{CS_j,U_i} \| TID^{new}_{U_i} \| T_3)$ and checks $M'_5 \overset{?}{=} M_5$. If it matches then CS_j identifies that the computed session key by U_i is correct and user's temporary identity is also updated. After that both U_i and CS_j store and establish session key SK_{U_i,CS_j} $(= SK_{CS_j,U_i})$ for their future secure communication. The summary of the "login and authentication and key agreement phases" is also provided in Fig. 2.

4.4. Blockchain implementation phase

This section contains the details of block creation, i.e., block BLK_j created by CS_j. Moreover, the details of verification process along with its addition into the blockchain (i.e., BC_{CS}) are explained. Whenever a cloud server CS_j receives information from a requesting node or a contributor, it converts that information to a transactions and adds it to a block BLK_j. Then block BLK_j is verified by the "P2P CS network". After consensus is reached among the cloud servers in the "P2P CS network", the block is added to the blockchain BC_{CS}. In the BUAKA-CS, cloud server CS_j receives information from the users (either requesting node or contributor). When a user U_i wishes to exchange some information then he/she can send that to CS_j through the established session key SK_{U_i,CS_j}. Suppose $Tx_1, Tx_2, \cdots, Tx_{n_t}$ are transactions made in a specific session by CS_j. The formation of a block BLK_j consists of transactions $Tx_1, Tx_2, \cdots, Tx_{n_t}$ and is described in Fig. 3. After the completion of all steps of consensus procedure, BLK_j will be added to the blockchain BC_{CS} [28,29].

Since all legitimate cloud servers maintain a distributed ledger in the from of blockchain BC_{CS} that contains the information of all the submitted tasks by the requesting nodes and the result of the completed task (submitted by the contributors) in the connected blocks of BC_{CS}. Therefore, any legitimate cloud server can access this information and also make it available to other legitimate users in a secure way by arranging an established session key SK_{CS_j,U_i}. We utilize a private blockchain in BUAKA-CS because of the confidentiality and secrecy requirements [30]. The block BLK_j can be formed as follows:

- The transactions is encrypted with ECC cryptosystem using the public key of EC, KU_{CS_j}.
- Further the encrypted transactions $E_{KU_{CS_j}}(Tx_x)$, $(i = 1, 2, \ldots, n_t)$, are utilized for the calculation of "Merkle tree root (MTR)".
- The hash of the information $\{VR_{BLK_j}, H(BLK_{j-1}), MTR, TS_{BLK_j}, OB_{BLK_j}, KU_{CS_j}, E_{KU_{CS_j}}(Tx_x)\}$ is calculated for the value of hash of the block $H(BLK_j)$.
- Further CS_j calculates digital signature using "elliptic curve digital signature (ECDSA) algorithm" for the block, i.e., $Sig_{KR_{CS_j}(BLK_j)}$.

After the formation of BLK_j, it is sent to the other nodes (miners) in "P2P CS network" to execute the consensus procedure. When a block BLK_j is received by a cloud server, i.e., CS_i from CS_j, then a leader L (a cloud server) will be selected from the all existing cloud servers in "P2P CS network". To achieve this, a leader selection algorithm, similar to [31] is adopted. Next, the "Ripple Protocol Consensus Algorithm (RPCA)" [32] is used for consensus through voting mechanism. For this, a leader selection algorithm by Zhang et al. [31] is used. Furthermore,

User (U_i/SP_{U_i})	Cloud server (CS_j)
Input ID_{U_i}, PW'_{U_i} and BIO'_{U_i}. Calculate $\sigma'_{U_i} = Rep(BIO'_{U_i}, \tau'_{U_i})$, $s = s^* \oplus h(ID_{U_i}\|\|\sigma'_{U_i}\|\|PW'_{U_i})$, $RPW'_{U_i} = h(s\|\|PW'_{U_i})$, $tc'_{U_i} = tc^*_{U_i} \oplus h(ID_{U_i}\|\| PW'_{U_i}\|\|\sigma'_{U_i})$, $h(ID_{U_i}\|\|N) = A_{U_i} \oplus RPW'_{U_i}, \beta' = h(h(ID_{U_i}\|\|N)\|\|RPW'_{U_i})$, $RID'_{U_i} = RID^*_{U_i} \oplus h(ID_{U_i}\|\|\sigma'_{U_i}\|\|PW'_{U_i})$, $RID'_{RA} = RID^*_{RA} \oplus h(\sigma'_{U_i}\|\|PW_{U'_i})$. $\gamma' = h(\beta'\|\|tc'_{U_i}\|\|\sigma'_{U_i}\|\|RID'_{U_i}\|\|RID'_{RA})$. Check $\gamma' = \gamma$? If so, compute $TID_{U_i} = TID^*_{U_i} \oplus h(ID_{U_i}\|\|\sigma_{U_i})$. Generate T_1 and r_1, and compute $M_1 = h(r_1\|\|tc_{U_i}) \oplus h(RID_{RA}\|\| h(ID_{U_i}\|\|N))$, $M_2 = h(h(RID_{RA}\|\|h(ID_{U_i}\|\|N))\|\| T_1\|\|h(r_1\|\|tc_{U_i}))$. $\xrightarrow{< Msg_1 = TID_{U_i}, M_1, M_2, T_1 >}$ (via open channel)	
	Check if $\|T_1 - T_1^*\| \leq \Delta T$? If so, fetch RID_{U_i} and X_{U_i}. Calculate $h(r_1\|\|tc_{U_i}) = M_1 \oplus X_{U_i}$, $M'_2 = h(X_{U_i}\|\| T_1\|\|h(r_1\|\|tc_{U_i}))$. Check $M'_2 = M_2$? If so, generate T_2 & r_2, and compute $z_{CS_j} = h(r_2\|\|TC_{CS_j})$, $M_3 = z_{CS_j} \oplus h(RID_{U_i}\|\|h(r_1\|\|tc_{U_i})\|\|T_1)$, $M_{CS_j} = RID_{CS_j} \oplus h(z_{CS_j}\|\|T_1\|\|T_2)$, $SK_{CS_j,U_i} = h(RID_{U_i}\|\|h(r_1\|\|tc_{U_i})\|\|RID_{CS_j}$ $\|\|X_{U_i}\|\| z_{CS_j}\|\|T_1\|\|T_2)$, $M_4 = h(SK_{CS_j,U_i}\|\|T_2)$, $M_T = TID^{new}_{U_i} \oplus h(SK_{CS_j,U_i}\|\|z_{CS_j}\|\| X_{U_i}\|\|T_1\|\|T_2)$. $\xleftarrow{< Msg_2 = M_3, M_{CS_j}, M_4, M_T, T_2 >}$ (via open channel)
Check if $\|T_2 - T_2^*\| \leq \Delta T$? If so compute $z_{CS_j} = M_3 \oplus h(RID_{U_i}\|\|h(r_1\|\|tc_{U_i})\|\|T_1)$, $RID_{CS_j} = M_{CS_j} \oplus h(z_{CS_j}\|\|T_1\|\|T_2)$ $SK_{U_i,CS_j} = h(RID_{U_i}\|\|h(r_1\|\|tc_{U_i})\|\|RID_{CS_j}\|\|$ $h(RID_{RA}\|\|h(ID_i\|\|N))\|\|z_{CS_j}\|\|T_1\|\|T_2)$, $M'_4 = h(SK_{U_i,CS_j}\|\|T_2)$. Check $M'_4 = M_4$? If so, compute $TID^{new}_{U_i} = M_T \oplus h(SK_{CS_j,U_i}\|\|z_{CS_j}\|\|$ $h(RID_{RA}\|\|h(ID_{U_i}\|\|N))\|\|T_1\|\|T_2)$. Generate T_3 and compute $M_5 = h(SK_{U_i,CS_j}\|\| TID^{new}_{U_i}\|\|T_3)$. $\xrightarrow{< Msg_3 = M_5, T_3 >}$ (via open channel)	
	Check $\|T_3 - T_3^*\| \leq \Delta T$. If so, compute $M'_5 = h(SK_{CS_j,U_i}\|\| TID^{new}_{U_i}\|\|T_3)$. Check if $M'_5 = M_5$? If so, session key is correct.
Both U_i and CS_j store the common session key SK_{U_i,CS_j} (= SK_{CS_j,U_i})	

Fig. 2. Login and authentication phases.

a cloud server in the P2P CS network will have an "ECC-based private-public key pair, i.e., (KR_{CS}, KU_{CS}), where $KU_{CS} = KR_{CS} \cdot G$". The cloud server knows the public keys of other cloud servers (miners).

The aforementioned mechanism is explained in Algorithms 1 and 2.

4.5. Dynamic device addition

This phase is required in case we need to deploy a new cloud server CS_j^{new} in the network. For that purpose, RA has to register CS_j^{new}, then CS_j^{new} will be deployed in the network after storing the

Block Header	
Version of block	VR_{BLK_j}
Hash of previous block	$H(BLK_{j-1})$
Merkle Tree Root	MTR
Block's timestamp value	TS_{BLK_j}
Block's owner	OB_{BLK_j}
Owner's public key	KU_{CS_j}
Block Payload (Encrypted Transactions)	
Encrypted Transactions	$E_{KU_{CS_j}}(Tx_x)$ where $x = 1, 2, \cdots n_t$
Hash of this block	$H(BLK_j)$
ECDSA signature on this block	$ECDSA.Sig_{KR_{CS_j}(BLK_j)}$

Fig. 3. Structure of block BLK_j proposed by CS_j.

required information in its database. For this purpose, following steps are executed:

- **DD1:** For this process, RA uses previously selected and computed values such as k_{ra}, N, ID_{RA} and $RID_{RA} = h(ID_{RA} \parallel k_{ra})$. RA again selects new identity of CS_j^{new} as $ID_{CS_j}^{new}$ and computes its pseudo identity $RID_{CS_j}^{new} = h(ID_{CS_j}^{new} \parallel k_{ra})$.
- **DD2:** RA further computes temporal credential of CS_j^{new} as $TC_{CS_j}^{new} = h(N \parallel RTS_{CS_j}^{new} \parallel k_{ra} \parallel RID_{RA} \parallel ID_{CS_j}^{new})$ where $RTS_{CS_j}^{new}$ is the registration timestamp of CS_j^{new}.
- **DD3:** RA stores $\{RID_{CS_j}^{new}, TC_{CS_j}^{new}\}$ securely in CS_j^{new}'s memory. Apart from that, RA also stores the information of the registered users, i.e., TID_{U_i}, RID_{U_i}, X_{U_i}, where $i = 1, 2 \cdots n_U$ in its database. Finally, CS_j^{new} stores values $\{RID_{CS_j}^{new}, TC_{CS_j}^{new}, TID_{U_i}, RID_{U_i}, X_{U_i}\}$ in its secure database.

4.6. Password and biometric update

It is always desirable that the legitimate registered users are able to update their password and biometric information anytime without involving the RA. In this regard, to update such information, following steps are executed:

- **PBU1:** User U_i inputs his/her identity ID_{U_i} and password $PW_{U_i}^{old}$ and also marks biometrics information $BIO_{U_i}^{old}$ at the sensor of the SP_{U_i}. SP_{U_i} calculates biometric secret key $\sigma_{U_i}^{old} = Rep(BIO_{U_i}^{old}, \tau_{U_i}^{old})$.
- **PBU2:** SP_{U_i} of U_i calculates $s = s^* \oplus h(ID_{U_i} \parallel \sigma_{U_i}^{old} \parallel PW_{U_i}^{old})$, $RPW_{U_i}^{old} = h(s \parallel PW_{U_i}^{old})$, $tc_{U_i} = tc_{U_i}^* \oplus h(ID_{U_i} \parallel PW_{U_i}^{old} \parallel \sigma_{U_i}^{old})$, $(ID_{U_i} \parallel N) = A_{U_i} \oplus RPW_{U_i}^{old}$, $\beta^{old} = h(h(ID_{U_i} \parallel N) \parallel RPW_{U_i}^{old})$, $RID_{U_i} = RID_{U_i}^* \oplus h(ID_{U_i} \parallel \sigma_{U_i}^{old} \parallel PW_{U_i}^{old})$, $RID_{RA} = RID_{RA}^* \oplus h(as\sigma_{U_i}^{old} \parallel PW_{U_i^{old}})$ and $\gamma^{old} = h(\beta^{old} \parallel tc_{U_i} \parallel \sigma_{U_i}^{old} \parallel RID_{U_i} \parallel RID_{RA})$. SP_{U_i} checks the condition $\gamma^{old} \overset{?}{=} \gamma$. If the match is successful, U_i is genuine user and can update the password and biometric information, otherwise, the update process is halted immediately. In case of successful match, U_i computes $TID_{U_i} = TID_{U_i}^* \oplus h(ID_{U_i} \parallel \sigma_{U_i}^{old})$.
- **PBU3:** User U_i again selects his/her new password $PW_{U_i}^{new}$ and computes new random password $RPW_{U_i}^{new} = h(s \parallel PW_{U_i}^{new})$. U_i inputs new biometric data $BIO_{U_i}^{new}$ to the biometric sensor of the smartphone SP_{U_i}. SP_{U_i} then computes $(\sigma_{U_i}^{new}, \tau_{U_i}^{new}) = Gen(BIO_{U_i}^{new})$, where $\sigma_{U_i}^{new}$ and $\tau_{U_i}^{new}$ are the "biometric secret key of size l bits and public reproduction parameter".
- **PBU4:** SP_{U_i} again computes $\beta^{new} = h(h(ID_{U_i} \parallel N) \parallel RPW_{U_i}^{new})$ and $\gamma^{New} = h(\beta^{new} \parallel tc_{U_i} \parallel \sigma_{U_i}^{new} \parallel RID_{U_i} \parallel RID_{RA})$. Then SP_{U_i} computes $s^{**} = s \oplus h(ID_{U_i} \parallel \sigma_{U_i}^{new} \parallel PW_{U_i}^{new})$, $tc_{U_i}^{**} = tc_{U_i} \oplus h(ID_{U_i} \parallel PW_{U_i}^{new} \parallel \sigma_{U_i}^{new})$, $A_{U_i}^{new} = h(ID_{U_i} \parallel N) \oplus RPW_i^{new}$, $RID_{U_i}^{**} = RID_{U_i} \oplus h(ID_{U_i} \parallel \sigma_{U_i}^{new} \parallel PW_{U_i}^{new})$, $RID_{RA}^{**} = RID_{RA} \oplus h(\sigma_{U_i}^{new} \parallel PW_{U_i}^{new})$ and $TID_{U_i}^{**} = TID_{U_i} \oplus h(ID_{U_i} \parallel \sigma_{U_i}^{new})$. Further values of $TID_{U_i}^*$, $RID_{U_i}^*$, RID_{RA}^*, $tc_{U_i}^*$, A_{U_i}, γ, τ_{U_i} and s^* are replaced with $TID_{U_i}^{**}$, $RID_{U_i}^{**}$, RID_{RA}^{**}, $tc_{U_i}^{**}$, $A_{U_i}^{new}$, γ^{new}, $\tau_{U_i}^{new}$ and s^{**}. Information such as $\{TID_{U_i}^{**}, RID_{U_i}^{**}, RID_{RA}^{**}, TC_{U_i}^{**}, A_{U_i}^{new}, \gamma^{new}, \tau_{U_i}^{new}, s^{**}, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$ is stored in the memory of SP_{U_i}. After the completion of these steps, the sensitive information tc_{U_i}, s, β^{old}, β^{new}, TID_{U_i}, RID_{U_i}, RID_{RA}, $RPW_{U_i}^{old}$, $RPW_{U_i}^{new}$, $h(ID_i \parallel N)$ is deleted

Algorithm 1 Block verification and addition in blockchain

```
Input: Provide block $BLK_j$ and key pairs $(KR_{CS_l}, KU_{CS_l} = KR_{CS_l} \cdot G)$ for cloud servers $CS_l$ in P2P CS network.
Output: Block $BLK_j$'s verification and addition in blockchain.
1: First select a leader node $(L)$ from peer nodes (cloud server-miner) in P2P CS network with the help of leader selection technique as discussed in [31]. Suppose $L$ earns a block $BLK_j$.
2: $L$ initializes $\theta_{cn} \leftarrow 0$, where $\theta_{cn}$ represents counter of votes.
   $L$ again initializes $flag_{CS_l} = 0, \forall\{l = 1, 2, \cdots, \phi_{n_{cs}}, L \neq CS_l\}$, where $\phi_{n_{cs}}$ represents the count of total number of cloud servers in P2P CS network.
3: $L$ then calculates "mathematical puzzle $\Re_l$" along with a current timestamp value $TS_l$ for the other cloud servers.
4: $L$ again encrypts that puzzle $\Re_l$ with its public key say $KU_{CS_l}$ for other cloud server such as $E_{KU_{CS_l}}(\Re_l, TS_l)$.
5: $L$ then broadcasts the message $SMSG_l = \{BLK_j, E_{KU_{CS_l}}(\Re_l, TS_l), TS_l\}$ to all other cloud servers $CS_l$, $(l = 1, 2, \cdots, \phi_{n_{cs}}, L \neq CS_l)$ in P2P CS network.
6: Suppose each $CS_l$ in P2P CS network receives $SMSG_l$ from $L$ at time $TS_l^*$.
7: for each cloud server $CS_l$ do
8:   if $(|TS_l - TS_l^*| < \Delta T)$ then
9:     Calculate "Merkle tree root", $MTR^*$ on encrypted transactions $\{E_{KU_{CS_l}}(Tx_x), x = 1, 2, \cdots, n_t\}$.
10:    if $(MTR^* \neq MTR)$ then
11:      Immediately halt the consensus procedure.
12:    else
13:      Compute block hash $H(BLK_j)^*$ of collected block $Block_j$ as $H(BLK_j)^* = h(VR_{BLK_j} || H(BLK_{j-1}) || MTR^* || TS_{BLK_j} || OB_{BLK_j} || KU_{CS_l} || E_{KU_{CS_l}}(Tx_1) || E_{KU_{CS_l}}(Tx_2) || \cdots || E_{KU_{CS_l}}(Tx_{n_t}))$.
14:      if $(H(BLK_j)^* = H(BLK_j))$ then
15:        Perform the verification of ECDSA signature $ECDSA.Sig_{KR_{CS_l}(BLK_j)}$ of $BLK_j$ on message $MSG = H(BLK_j)^*$ through ECDSA signature verification mechanism.
16:        if If verification of signature happens successfully then
17:          Perform the decryption of encrypted puzzle $\Re_l$ via defined private key $KR_{CS_l}$ as $(\Re_l^*, TS_l'') = D_{KR_{CS_l}}[E_{KU_{CS_l}}(\Re_l, TS_l)]$ via ECC decryption mechanism.
18:          if $(TS_l'' = TS_l)$ then
19:            Send verification message for this block which consists of verification status $(VF_{st})$ and puzzle's solution such as $RMSG_l = \{E_{KU_L}(\Re_l^*, VF_{st})\}$ to the leader $L$.
20:          end if
21:        end if
22:      end if
23:    end if
24:  end if
25: end for
```

Algorithm 2 Block verification and addition in blockchain (continued...)

```
26: for each received $RMSG_l$ from the other miners (responders) $CS_l$ do
27:     $L$ enumerates $(\Re_l', VF_{st}) = D_{r_L}[E_{KU_L}(\Re_l^*, VF_{st})]$.
28:     if (($\Re_l' = \Re_l$) and ($VF_{st} = valid$) and ($flag_{CS_l} = 0$)) then
29:         $L$ settles $\theta_{cn} = \theta_{cn} + 1$ and $flag_{CS_l} = 1$.
30:     end if
31: end for
32: if ($\theta_{cn}$ is more than 50% of the votes) then
33:     Transaction will proceed to the next round.
34:     if ($\theta_{cn}$ less than contemplated threshold value, for instance, 80% of the votes) then
35:         Continue from Step 26.
36:     else
37:         Send response of this commitment to all other peer nodes $CS_l$ of P2P CS network.
            Add block $BLK_j$ in blockchain $BC_{CS}$ and dissolves the "consensus procedure".
38:     end if
39: end if
```

from the memory of SP_{U_i} to protect against active attacks such as privileged insider attack, unauthorized session key computation and illegal user password guessing, user impersonation, etc.

4.7. Smartphone revocation

Suppose smartphone of legitimate user is lost or stolen and the user still wants to access the services of CS_j. In such situation, we need to execute the following steps for the revocation of smartphone.

- **SRP1:** User U_i picks his/her identity ID_{U_i}, password PW_{U_i} and a new random secret number s^n. U_i computes a random password $RPW^n_{U_i} = h(s^n \| PW_{U_i})$ and sends $\{ID_{U_i}, RPW^n_{U_i}\}$ to RA through a secure channel.
- **SRP2:** In this process, RA uses previously generated and computed values such as N, ID_{RA}, k_{ra} and $RID_{RA} = h(ID_{RA} \| k_{ra})$. Furthermore, RA computes pseudo identity of U_i as $RID_{U_i} = h(ID_i \| k_{ra})$ and his/her new temporal credential as $TC^n_{U_i} = h(N \| RTS^n_{U_i} \| k_{ra} \| RID_{RA} \| ID_{U_i})$ where $RTS^n_{U_i}$ is the new registration timestamp of U_i.
- **SRP3:** RA again computes $\alpha^n = RPW^n_{U_i} \oplus h(ID_{U_i} \| N)$ and generates a fresh temporary identity $TID^n_{U_i}$ for U_i. Then RA sends $\{TID^n_{U_i}, RID_{U_i}, RID_{RA}, TC^n_{U_i}, \alpha^n\}$ to U_i through a secure channel.
- **SRP4:** After receiving $\{TID^n_{U_i}, RID_{U_i}, RID_{RA}, TC^n_{U_i}, \alpha^n\}$ from RA, U_i inputs biometric data BIO_{U_i} to the smartphone SP_{U_i}. SP_{U_i} then computes $(\sigma_{U_i}, \tau_{U_i}) = Gen(BIO_{U_i})$, where σ_{U_i} and τ_{U_i} are the "biometric secret key of size l bits, and public reproduction parameter".
- **SRP5:** SP_{U_i} of U_i again computes $h(ID_{U_i} \| N) = \alpha^n \oplus RPW^n_{U_i}$, $tc^n_{U_i} = h(TC^n_{U_i} \| s^n \| \sigma_{U_i})$, $\beta^n = h(h(ID_{U_i} \| N) \| RPW^n_{U_i})$ and $\gamma^n = h(\beta^n \| tc^n_{U_i} \| \sigma_{U_i} \| RID_{U_i} \| RID_{RA})$.
- **SRP6:** Then SP_{U_i} computes $s^{n*} = s^n \oplus h(ID_{U_i} \| \sigma_{U_i} \| PW_{U_i})$, $tc^{n*}_{U_i} = tc^n_{U_i} \oplus h(ID_{U_i} \| PW_{U_i} \| \sigma_{U_i})$, $A^n_{U_i} = h(ID_{U_i} \| N) \oplus RPW^n_{U_i}$, $RID^{n*}_{U_i} = RID_{U_i} \oplus h(ID_{U_i} \| \sigma_{U_i} \| PW_{U_i})$, $TID^{n*}_{U_i} = TID^n_{U_i} \oplus h(ID_{U_i} \| \sigma_{U_i})$ and $RID^{n*}_{RA} = RID_{RA} \oplus h(\sigma_{U_i} \| PW_{U_i})$. Further information such as $\{TID^{n*}_{U_i}, RID^{n*}_{U_i}, RID^{n*}_{RA}, tc^{n*}_{U_i}, A^n_{U_i}, \gamma^n, \tau_{U_i}, s^{n*}, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$ is stored in the memory of SP_{U_i}. After the completion of above registration steps by U_i, the sensitive information $tc^n_{U_i}$, s^n, α^n, β^n, $TID^n_{U_i}$, $RPW^n_{U_i}$, RID_{U_i}, RID_{RA}, $h(ID_i \| N)$ are deleted from the memory of SP_{U_i} to protect against the active attacks such as "privileged insider and user impersonation attacks, unauthorized session key computation, and illegal password guessing attacks". After the completion of above steps, RA computes $X_{U_i} = h(RID_{RA} \| h(ID_{U_i} \| N))$ and sends the encrypted registration information of U_i, $\{TID^n_{U_i}, RID_{U_i}, X_{U_i}\}$ (through the established symmetric key K_{RA,CS_j}) to CS_j. After receiving this information, CS_j decrypts the information with K_{RA,CS_j} and stores following information, $\{RID_{CS_j}, TC_{CS_j}, TID^n_{U_i}, RID_{U_i}, X_{U_i}\}$ in the secure database.

5. Security analysis of BUAKA-CS

In this section, we formally analyze the proposed BUAKA-CS to prove its resilience against various possible attacks. We carry out the formal security analysis and verification through two well-known tools, such as "Real-Or-Random (ROR) random oracle model" and "AVISPA software validation tool".

5.1. Formal security analysis through Real-Or-Random (ROR) model

ROR model [33] is standard model which is applied to prove the security of Session Key (SK) in BUAKA-CS. Moreover, this model is recently used in various authentication protocols to prove the security of SK [9].

There are two participants in our system, i.e., SP_{U_i} of U_i and CS_j during the "login and authentication phases of BUAKA-CS". We represent $\mathcal{O}^t_{SP_{U_i}}$ and $\mathcal{O}^v_{CS_j}$ as the instances t and v of SP_{U_i} and CS_j, respectively. These instances are also considered as the *oracles*.

The ROR contains the following:

- **Participants.** We represent $\mathcal{O}^t_{SP_{U_i}}$ and $\mathcal{O}^v_{CS_j}$ as the instances t and v of SP_{U_i} and CS_j, respectively. These instances are also considered as the *oracles*.
- **Accepted state.** Upon the receiving of last protocol message, an instance $\mathcal{O}^t$ is in "accept state" then we assume that $\mathcal{O}^t$ enters in the "accepted state". If all communicated messages (i.e., send and receive messages by $\mathcal{O}^t$) are concatenated in the order. (sid) of $\mathcal{O}^t$ is considered as the session identification for a specific session.
- **Partnering.** Instance $\mathcal{O}^{t_1}$ and $\mathcal{O}^{t_2}$ are assumed as the partner of each other, in case if following 3 conditions are assured: (1) "both instances are in the accepted states", (2) "both instances mutually authenticate each other and share the same session id sid", and (3) "both instances are mutual partners".
- **Freshness.** Let say session key SK_{ij} does not disclose through reveal query $\mathcal{R}$. In such situation, we can say $\mathcal{O}^t_{SP_{U_i}}$ or $\mathcal{O}^v_{CS_j}$ are fresh and up-to date.
- **Adversary.** The adversary $\mathcal{A}$ is implemented via Dolev-Yao (DY) model in which it is considered as $\mathcal{A}$ is controlling the communication happening via public channel and can eavesdrop (reading in unauthorized way), delete, update, or inject fake messages with the help of following queries.
  - $\mathcal{E}$ $(\mathcal{O}^t, \mathcal{O}^v)$: The eavesdropping attack a kind of passive attack is built with this *execute* query. With this query $\mathcal{A}$ can eavesdrop the exchanged messages between entities U_i and CS_j.
  - $\mathcal{S}$ $(\mathcal{O}^t, msg)$: It is built like an active attack. $\mathcal{A}$ sends/ receives messages to/ from $\mathcal{O}^t$ through the *send* query.
  - $\mathcal{R}$ $(\mathcal{O}^t)$: The *reveal* query tries to disclose session key SK_{ij} estimated by $\mathcal{O}^t$ and the other pair entity to the attacker $\mathcal{A}$ for a specific session.
  - $\mathcal{CSP}$ $(\mathcal{O}^t_{SP_{U_i}})$: It is built like an active attack. Under the *corrupt smartphone* query, $\mathcal{A}$ tries to obtain all secret information stored in the legitimate user's stolen or lost smartphone SP_{U_i} via the "sophisticated power analysis attack" [26,34].

– $\mathcal{T}$ ($\mathcal{O}^t$): Before the starting of the game, an unbiased coin c is flipped. It is built through *test* query. On the basis of the outcome of coin flipping, following decision are taken. If the result of coin flipping is, $c = 1$ then $\mathcal{O}^t$ returns SK_{ij}. Otherwise, in case of $c = 0$, a null value ($\perp$) is returned. In the formal security analysis, we apply a restriction that $\mathcal{A}$ can have access to limited number of $\mathcal{CSP}$ ($\mathcal{O}^t_{SP_{U_i}}$) and $\mathcal{CSP}$ ($\mathcal{O}^v_{CS_j}$) queries. However, $\mathcal{A}$ can execute unlimited number of $\mathcal{T}$ ($\mathcal{O}^t$) queries.

- **Semantic security:** The semantic security of session key SK_{ij} (established between U_i and CS_j) using RoR model depends on the adversary $\mathcal{A}$'s ability to find out the difference between real session key and a random number. The result of $\mathcal{T}$ ($\mathcal{O}^t$) query should be compatible with random bit c. Let say $\mathcal{A}$ guesses c' bit. If *Succ* is the expression of "winning probability", the "advantage of $\mathcal{A}$ in breaking the semantic security of SK_{ij} of BUAKA-CS is implied by $Adv_{\mathcal{A}}^{AKM} = 2|Pr[Succ] - 1|$, where the probability of an event X is $Pr[X]$".
- **Random oracle:** The cryptographic one-way hash function $h(\cdot)$ in the proposed scheme is used. it is built like a random oracle, i.e., $Hash$ value of $h(\cdot)$ is public where every participant including $\mathcal{A}$ can have ability of accessing the $Hash$ oracle.

Wang et al. [35] identified that the "passwords chosen by users follow the Zipf's law and the distribution of passwords is verily different from the uniform distribution". Moreover, the "usage size of password dictionary is generally much smaller". For example, users usually do not utilize the "whole space of passwords" and place of that they use small space of characters [35]. The Zipf's law is useful in formal security analysis to prove "session key security" of a presented scheme. The "semantic security of the session key" in BUAKA-CS is provided in Theorem 1. Similar proof [9] is also adhered in Theorem 1.

Theorem 1. *Let an adversary $\mathcal{A}$ can have breaking ability of SK-security of proposed BUAKA-CS via advantage function $Adv_{\mathcal{A}}^{AKM}$. Again q_h, q_s, and $|Hash|$ are "the number of $Hash$ queries, number of $Send$ queries, and the range space of the hash function $h(\cdot)$", respectively. If l_b is the number of bits present in the U_i's biometric secret key σ_{U_i}, and C', and s' express the Zipf's parameters [35], then*

$$Adv_{\mathcal{A}}^{BUAKA-CS} \leq \frac{q_h^2}{|Hash|} + 2\max\left\{C' \cdot q_s^{s'}, \frac{q_s}{2^{l_b}}\right\}.$$

Proof. For this formal proof, we have taken four games Gm_j, $j \in [0, 3]$. An event is also defined wherein $\mathcal{A}$ can guess the random bit c in the Gm_j correctly and its "success probability" can be express as $Succ_{\mathcal{A}}^{Gm_j}$. The advantage of $\mathcal{A}$ in winning the game Gm_j is expressed by $Adv_{\mathcal{A},Gm_j}^{BUAKA-CS} = Pr[Succ_{\mathcal{A}}^{Gm_j}]$.

- *Game Gm_0*: It is the first game and is executed as the identical game with actual scheme running under RoR model. Through such assumptions we can write down:

$$Adv_{\mathcal{A}}^{BUAKA-CS} = |2.Adv_{\mathcal{A},Gm_0}^{BUAKA-CS} - 1|. \tag{1}$$

- *Game Gm_1*: This game is the simulation of an eavesdropping attack. In this game $\mathcal{E}$ query is executed. $\mathcal{A}$ executes $\mathcal{T}$ query at the end of the game.
Then, $\mathcal{A}$ has to find out the difference between session key SK_{ij} and a random number after the output of $\mathcal{T}$ query is obtained. In proposed scheme, session key estimated by both parties U_i and CS_j is $SK_{U_i,CS_j} = h(RID_{U_i} \| h(r_1 \| tc_{U_i}) \| RID_{CS_j} \| h(RID_{RA} \| h(ID_i \| N)) \| z_{CS_j} \| T_1 \| T_2)$. The computation of session key contains both "long term secrets, i.e., secret keys and identities as well as the short term secrets, i.e., freshly generated timestamp values and random secret values". Hence through the eavesdropping of messages Msg_1, Msg_2 and Msg_3, winning probability of the game Gm_1 is un-changed and is not further helpful for the calculation of session key SK_{ij}. As per the "indistinguishability of Gm_0 and Gm_1" we can write down:

$$Adv_{\mathcal{A},Gm_1}^{BUAKA-CS} = Adv_{\mathcal{A},Gm_0}^{BUAKA-CS} \tag{2}$$

Game Gm_2: Another active attack is simulated through this game. In this game $\mathcal{A}$ simulates the "$\mathcal{S}$ and $Hash$ queries" for the misleading of a communicating party to obtain the fake messages. The $\mathcal{A}$ goes through a competition through some $Hash$ queries to find out the collision in hash outcomes for Msg_1, Msg_2, and Msg_3 messages. In the calculations of these messages, we use "both long term secrets, i.e., secret keys and identities as well as short term secrets, i.e., freshly generated timestamp values and random secret values", which produces different messages in different sessions. However, if $\mathcal{A}$ executes several $\mathcal{S}$ queries, the adversary does not have the ability to find out any collision in hash outputs. Hence through the birthday paradox following result can be obtained:

$$|Adv_{\mathcal{A},Gm_1}^{BUAKA-CS} - Adv_{\mathcal{A},Gm_2}^{BUAKA-CS}| \leq \frac{q_h^2}{2|Hash|}. \tag{3}$$

- *Game Gm_3*: In this last game $\mathcal{A}$ simulates the $\mathcal{CSP}$ query. via the stored information in U_i's smartphone SP_{U_i} and also through $\mathcal{CSP}$ query, $\mathcal{A}$ can try to obtain the biometric secret key σ_{U_i} of U_i by using stored values, i.e., $TID^*_{U_i}$, $RID^*_{U_i}$, RID^*_{RA}, $tc^*_{U_i}$, A_{U_i} and s^*. Without having the knowledge of secret number s, of U_i, it becomes computationally difficult problem for $\mathcal{A}$ to guess password PW_{U_i} of U_i correctly via *Send* query. Again, proposed BUAKA-CS uses the "fuzzy extractor method", which can extract at most nearly l_b random bits. Thus "probability of guessing σ_{U_i}" can be bordering on $1/2^{l_b}$ [36]. Further, games Gm_2 and Gm_3 are identical in the absence of the "password/biometrics guessing attacks". Hence, using the Zipf's law on passwords [35], the following result can be obtained.

$$|Adv_{\mathcal{A},Gm_2}^{BUAKA-CS} - Adv_{\mathcal{A},Gm_3}^{BUAKA-CS}| \leq \max\left\{C' \cdot q_s^{s'}, \frac{q_s}{2^{l_b}}\right\} \tag{4}$$

where "C' and s' are the Zipf's parameters" [35].

In the execution of all games, $\mathcal{A}$ needs to guess the correct bit c, which is expressed as:

$$Adv_{\mathcal{A},Gm_3}^{AKM} = \frac{1}{2}. \tag{5}$$

Using Eqs. (1) and (2), the following result is obtained.

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{BUAKA-CS} &= |Adv_{\mathcal{A},Gm_0}^{BUAKA-CS} - \frac{1}{2} \\ &= |Adv_{\mathcal{A},Gm_1}^{BUAKA-CS} - \frac{1}{2}| \\ &= |Adv_{\mathcal{A},Gm_1}^{BUAKA-CS} - Adv_{\mathcal{A},Gm_3}^{BUAKA-CS}| \end{aligned} \tag{6}$$

The "triangular inequality" and Eqs. (4) and (6) proceed to following results:

$$\begin{aligned} \frac{1}{2}.Adv_{\mathcal{A}}^{BUAKA-CS} &= |Adv_{\mathcal{A},Gm_1}^{BUAKA-CS} - Adv_{\mathcal{A},Gm_3}^{BUAKA-CS}| \\ &\leq |Adv_{\mathcal{A},Gm_1}^{BUAKA-CS} - Adv_{\mathcal{A},Gm_2}^{BUAKA-CS}| \\ &\quad + |Adv_{\mathcal{A},Gm_2}^{BUAKA-CS} - Adv_{\mathcal{A},Gm_3}^{BUAKA-CS}| \\ &\leq \frac{q_h^2}{2|Hash|} + \max\left\{C' \cdot q_s^{s'}, \frac{q_s}{2^{l_b}}\right\} \end{aligned} \tag{7}$$

Finally, if we multiplying "both sides of Eq. (7) with a factor of 2", we get the expected result:

$$Adv_{\mathcal{A}}^{BUAKA-CS} \leq \frac{q_h^2}{|Hash|} + 2\max\left\{C' \cdot q_s^{s'}, \frac{q_s}{2^{l_b}}\right\}. \quad \square$$

5.2. Informal security analysis of BUAKA-CS

in this section, we mathematically prove the security of BUAKA-CS using appropriate assumptions. We observe that BUAKA-CS is able to prevent most of the existing attacks.

5.2.1. Resilience against replay attack

In BUAKA-CS, we use freshly generated timestamp values in all of the exchanged messages, i.e., T_1, T_2 and T_3 in messages Msg_1, Msg_2 and Msg_3. These values are also verified when the messages are received at the receiver end using condition like $|T_x - T_x^*| \leq \Delta T$, where $x = 1, 2, 3$; If it holds then it is identified that message is original. Hence we can say that BUAKA-CS is able to prevent the replay attack.

5.2.2. Prevention of Man-in-The-Middle (MiTM) attack

Suppose, an active adversary $\mathcal{A}$, eavesdrops some of the exchanged messages and later on tries to modify them and sends to other communicating party to launch MiTM on BUAKA-CS. Let $\mathcal{A}$ generates a temporary identity $TID_{U_i}^a$, a timestamp T_1^a, and a random secret r_1^a, and tries to compute $M_1^a = h(r_1^a \| tc_{U_i}) \oplus h(RID_{RA} \| h(ID_{U_i} \| N))$ and $M_2^a = h(h(RID_{RA} \| h(ID_{U_i} \| N)) \| T_1^a \| h(r_1^a \| tc_{U_i}))$, where $TC_{U_i} = h(N \| RPW_{U_i} \| RTS_{U_i} \| k_{ra} \| RID_{RA} \| ID_{U_i})$, and $tc_{U_i} = h(TC_{U_i} \| s \| \sigma_{U_i})$. However, such kind of message creation $Msg_1^a = \langle TID_{U_i}^a, M_1^a, M_2^a, T_1^a \rangle$ is not possible for $\mathcal{A}$ because the adversary does not have the knowledge of secret values k_{ra}, N, s, RTS_{U_i}, $RPW_{U_i} = h(s \| PW_{U_i})$ and identities $RID_{RA} = h(ID_{RA} \| k_{ra})$. Furthermore, $\mathcal{A}$ cannot modify the value of Msg_1 or the other remaining messages. Therefore, it will be infeasible for $\mathcal{A}$ to launch MiTM attack. Hence BUAKA-CS is able to prevent MiTM attack.

5.2.3. Prevention of impersonation attack

Suppose, an active adversary $\mathcal{A}$ tries to impersonate a genuine user with the help of some generated parameters. Let $\mathcal{A}$ generates a temporary identity $TID_{U_i}^{ai}$, a timestamp T_1^{ai}, and a random secret r_1^{ai}, and tries to compute $M_1^{ai} = h(r_1^{ai} \| tc_{U_i}) \oplus h(RID_{RA} \| h(ID_{U_i} \| N))$ and $M_2^{ai} = h(h(RID_{RA} \| h(ID_{U_i} \| N)) \| T_1^{ai} \| h(r_1^{ai} \| tc_{U_i}))$, where $TC_{U_i} = h(N \| RPW_{U_i} \| RTS_{U_i} \| k_{ra} \| RID_{RA} \| ID_{U_i})$ and $tc_{U_i} = h(TC_{U_i} \| s \| \sigma_{U_i})$. However, creation of message $Msg_1^{ai} = \langle TID_{U_i}^{ai}, M_1^{ai}, M_2^{ai}, T_1^{ai} \rangle$ is not possible for $\mathcal{A}$ because the adversary does not have the knowledge of secret values k_{ra}, N, s, RTS_{U_i}, $RPW_{U_i} = h(s \| PW_{U_i})$ and identities $RID_{RA} = h(ID_{RA} \| k_{ra})$. Therefore, $\mathcal{A}$ cannot create the correct value of Msg_1 or the other remaining messages. Thus, $\mathcal{A}$ cannot impersonate a genuine user. In the similar way, we can prove that $\mathcal{A}$ cannot act like the original cloud server. Hence BUAKA-CS is able to prevent the impersonation attacks.

5.2.4. Traceability and anonymity

In all computed and exchanged messages, we use "freshly generated timestamp values and random secret values" that help us to get distinct messages in each session. Therefore, $\mathcal{A}$ cannot trace the exchanged messages. Hence BUAKA-CS supports traceability. Furthermore, we do not exchange any identity in the plaintext format in any exchanged message. Instead, we use temporary identity which is updated in each session according to the deployed mechanism. Hence BUAKA-CS also supports anonymity.

5.2.5. Mitigation of synchronization problem and associated attack

In BUAKA-CS, following messages are exchanged in each session, $\{Msg_1 = TID_{U_i}, M_1, M_2, T_1\}$, $\{Msg_2 = M_3, M_{CS_j}, M_4, M_T, T_2\}$ and $\{Msg_3 = M_5, T_3\}$. In this exchange, the last message Msg_3 contains $M_5 = h(SK_{U_i,CS_j} \| TID_{U_i}^{new} \| T_3)$. This mechanism restricts cloud server CS_j to cross check the updated value of temporary identity $TID_{U_i}^{new}$. If the verification of $M_5' = M_5$ happens successfully at CS_j' end, then it is considered that SP_{U_i} of U_i updates the value of user's temporary identity correctly. Otherwise CS_j, immediately terminates the session with U_i. Moreover, it is also assumed that all the heterogeneous devices such as users (smartphones) and cloud servers are synchronized with their clocks, and mutually agree on a maximum transmission delay (ΔT) to prevent the synchronization problem.

5.2.6. Protection against stolen smartphone attack

In BUAKA-CS, the genuine user can use his/her smartphone SP_{U_i} to communicate with the cloud server. The SP_{U_i} of genuine user U_i stores $\{TID_{U_i}^*, RID_{U_i}^*, RID_{RA}^*, tc_{U_i}^*, A_{U_i}, \gamma, \tau_i, s^*, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$. Moreover, we do not store any secret information such as TC_{CS_j}, RID_{CS_j}, X_{U_i}, TC_{U_i}, tc_{U_i}, s, α, β, RID_{U_i}, RID_{RA}, PW_{U_i}, RPW_{U_i}, $h(ID_{U_i} \| N)$ in the memory of SP_{U_i}. If $\mathcal{A}$ steals SP_{U_i} and then tries to deduce the stored information using power analysis attack [26], it will not help the adversary because the adversary still does not have the required information such as user password and session key or other useful information to launch the impersonation attack. Hence BUAKA-CS protects against the stolen smartphone attack.

5.2.7. Prevention of ephemeral secret leakage (ESL) attack under CK-adversary model

In BUAKA-CS, session key between U_i and CS_j is estimated as $SK_{CS_j,U_i} = h(RID_{U_i} \| h(r_1 \| tc_{U_i}) \| RID_{CS_j} \| X_{U_i} \| z_{CS_j} \| T_1 \| T_2)$, where $z_{CS_j} = h(r_2 \| TC_{CS_j})$, $TC_{U_i} = h(N \| RPW_{U_i} \| RTS_{U_i} \| k_{ra} \| RID_{RA} \| ID_{U_i})$, $RID_{U_i} = h(ID_{U_i} \| k_{ra})$, $RID_{CS_j} = h(ID_{CS_j} \| k_{ra})$, $X_{U_i} = h(RID_{RA} \| h(ID_{U_i} \| N))$, $TC_{CS_j} = h(N \| RTS_{CS_j} \| k_{ra} \| RID_{RA} \| ID_{CS_j})$ and $tc_{U_i} = h(TC_{U_i} \| s \| \sigma_{U_i})$. Therefore, the computation of "session key contains both long term secrets, i.e., secret keys and identities as well as the short term secrets, i.e., freshly generated timestamp and random secret". In each session, a new key is generated and established. $\mathcal{A}$ does not have the knowledge of these secret values which are mandatory to compute the key. Therefore, $\mathcal{A}$ does not have the ability to compute the session key. Hence BUAKA-CS prevents the ESL attack.

5.2.8. Prevention of stolen verifier attack

In BUAKA-CS, registration information of U_i and CS_j are stored in either the secure memory of SP_{U_i} or CS_j. Moreover, we do not store any of the sensitive information in their memory directly. For instance, SP_{U_i} stores $\{TID_{U_i}^*, RID_{U_i}^*, RID_{RA}^*, tc_{U_i}^*, A_{U_i}, \gamma, \tau_i, s^*, h(\cdot), Gen(\cdot), Rep(\cdot), t\}$ and CS_j stores $\{RID_{CS_j}, TC_{CS_j}, TID_{U_i}, RID_{U_i}, X_{U_i}\}$ in the secure database. Therefore, useful information are not available to $\mathcal{A}$ for launching further attacks, i.e., password guessing, impersonation, and unauthorized session key computation. Hence, BUAKA-CS prevents stolen verifier attack.

5.2.9. Prevention of privileged insider attack

In BUAKA-CS, a "privileged insider user of the trusted authority" may have the knowledge of registration information of user and cloud server. However, the insider will not be able to launch other attacks such as password guessing, impersonations and unauthorized session key computation as some of the secrets such as PW_{U_i}, σ_{U_i}, s and tc_{U_i} are not known to every legitimate user (except for the one taking part in communication). Hence, BUAKA-CS also prevents privileged insider attack.

5.2.10. Preservation of perfect forward secrecy

Perfect forward secrecy in an authenticated key agreement scheme is a security property which ensures that all the session keys established in different sessions over the public channels are secure. In the proposed BUAKA-CS, a unique session key for each session is created between two communicating entities instead of relying on the sessions to keep connections open. Thus, an adversary cannot gain access to information in the network from more than a single communication between the communicating entities. Moreover, each session key is generated using the random short-term secrets along with the current timestamps. This makes each established session key in every session is unique and distinct. Even if the session key in a particular session is compromised, the session keys established in other sessions are still secure as they are distinct. As a result, BUAKA-CS maintains the perfect forward secrecy property.

```
SUMMARY
  SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS

PROTOCOL
 /home/wazid/Desktop/span
 /results/auth.if

GOAL  as specified

BACKEND  OFMC

STATISTICS
  TIME 2395 ms
  parseTime 0 ms
  visitedNodes: 1990 nodes
  depth: 12 plies
```

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/wazid/Desktop/span
  /results/auth.if
GOAL
  As specified

BACKEND
  CL–AtSe

STATISTICS
  Analysed   : 1512 states
  Reachable  : 165 states
  Translation: 0.17 seconds
  Computation: 0.01 seconds
```

Fig. 4. AVISPA simulation results under OFMC and CL-AtSe backends.

5.3. Formal security verification of BUAKA-CS with AVISPA

Here, we provide the details of formal security verification of BUAKA-CS using the “Automated Validation of Internet Security Protocols and Applications (AVISPA)” [37]. We first implement BUAKA-CS using the “High Level Protocol Specification Language (HLPSL)” [37]. HLPSL is a role-based language which contains following types of roles:

- Basic roles: These roles represent various participating entities, i.e., communicating parties considered in the protocol.
- Composition roles: Under these roles, various scenarios including basic roles are presented.

It is necessary to consider intruder in the protocol which is one of basic adversarial role and is denoted as i. HLPSL2IF translator converts HLPSL specification of BUAKA-CS into Intermediate Format (IF). Then IF is provided as an input to the back-end in the AVISPA tool. AVISPA consists of four back-ends: (a) “On-the-Fly Model-Checker (OFMC)”, (b) “Constraint Logic based Attack Searcher (CL-AtSe)”, (c) “SAT-based Model-Checker (SATMC)”, and (d) “Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)”. The readers are referred to [37] for more details on these backends.

The implementation of various phases of BUAKA-CS such as registration, login and authentication is carried out through HLPSL specification. There are basic roles corresponding to registration authority RA, user U_i, and cloud server CS_j. In the simulation under the AVISPA back-ends, three verifications are performed that are: (a) “executability checking on non-trivial HLPSL specifications”, (b) “checking for replay attack”, and (c) “checking Dolev-Yao threat model (DY model)” [25]. It is necessary to perform executability check which provides assurance that the designed protocol will reach to a possible attack happening state during the running of the protocol. To discover the existence of replay attack on BUAKA-CS, both OFMC and CL-AtSe checks are necessary which identify that the legitimate agents can execute the designed protocol via a search of a passive intruder. Furthermore, both OFMC and CL-AtSe back-ends also discover the possibility of man-in-the-middle attack under DY model [25]. The obtained results of simulations are provided in Fig. 4 which clearly explain that proposed BUAKA-CS is secure against replay and man-in-the-middle attacks.

5.4. Formal security verification of BUAKA-CS using ProVerif

This section presents the formal security verification of the proposed scheme (BUAKA-CS) using the widely-recognized ProVerif simulation tool. This tool is based on applied pi calculus and can be used to verify whether an attacker can attack the session key [38,39]. In addition, the tool has the ability of an attack reconstruction (i.e, when a property cannot be proved, an execution trace which falsifies the desired property is constructed [40,41]. The following are the silent features that are supported in ProVerif:

```
(* —— channels —— *)
free pch: channel. (* public channel *)
free sch: channel [private]. (* private channel *)
(* —— shared keys —— *)
free SKucs:bitstring [private]. (* the session key of user *)
free SKcsu:bitstring [private]. (* the session key of server *)
(* —— Servers secret key —— *)
free RTScs:bitstring [private].
free Kra:bitstring [private].
free RIDcs:bitstring [private].
free TCcs:bitstring [private].
(* —— constants —— *)
free IDu:bitstring [private].
free IDcs:bitstring [private].
free PWu:bitstring [private].
free tcu1:bitstring [private].
free s1:bitstring [private].
free Au:bitstring [private].
free RIDu1:bitstring [private].
free RIDra1:bitstring [private].
free gamma:bitstring [private].
free TIDu1:bitstring [private].
free RIDui:bitstring [private].
free Xui:bitstring [private].
const BIOu:bitstring [private].
(* —— functions and equations —— *)
fun H(bitstring):bitstring. (* hash function *)
fun Generation(bitstring):bitstring. (* Fuzzy extractor function *)
fun xor(bitstring,bitstring):bitstring. (* XOR operation *)
fun con(bitstring,bitstring):bitstring. (* string concatenation *)
equation forall x:bitstring,y:bitstring; xor(xor(x,y),y) = x.
(* —— aims for verification —— *)
query attacker(SKucs).
query attacker(SKcsu).
query id:bitstring; inj-event(UserAuth(id))
==> inj-event(UserStart(id)).
(* —— event —— *)
event UserStart(bitstring). (* User starts authentication *)
event UserAuth(bitstring). (* User is authenticated *)
```

Fig. 5. Code in ProVerif for declaration of channels, keys, constants, functions, equations, queries and events.

- ProVerif is a widely used automatic cryptographic protocol verifier that needs the Horn clause based protocol representation and it also works upon the Dolev-Yao symbolic model [25].
- ProVerif has the ability to handle various cryptographic primitives that include symmetric key and asymmetric key cryptography, Diffie–Hellman key agreements, digital signatures, hash functions, XOR operations etc.
- In an unbounded message space, it can execute an unbounded number of parallel sessions of a tested protocol.
- ProVerif is capable of proving reachability, privacy, traceability, and verifiability properties, which leverage the analysis of secrecy and authentication properties.

The proposed scheme has been modeled through ProVerif and corresponding the source codes have been presented in Fig. 5, Fig. 6, Fig. 7, and Fig. 8. ProVerif simulation of the proposed scheme requires declaration of various channels, constants, free variables, equations, events etc., which are shown in Fig. 5. Fig. 6 shows the ProVerif

```
(* —— user starts —— *)
let User=
let sigma = Generation(BIOu) in
!
(
event UserStart(IDu);
let s = xor(s1,H(con(IDu,con(sigma,PWu)))) in
let RPWu = H(con(s,PWu)) in
let tcu11 = xor(tcu1,H(con(IDu,con(PWu,sigma)))) in
let IDN = xor(Au,RPWu) in
let beta = H(con(IDN,RPWu)) in
let RIDu11 = xor(RIDu1,H(con(IDu,con(sigma,PWu)))) in
let RIDra11 = xor(RIDra1, H(con(sigma,PWu))) in
let gamma1 = H(con(beta,con(tcu11,con(sigma,
con(RIDu11,RIDra11))))) in
if gamma = gamma1 then
new T1:bitstring;
new r1:bitstring;
let TIDu11 = xor(TIDu1,H(con(IDu,sigma))) in
let M1 = xor(H(con(r1,tcu11)),H(con(RIDra11,IDN))) in
let M2 = H(con(H(con(RIDra11,IDN)),con(T1,H(con(r1,tcu11)))))
in out(pch,(TIDu11,M1,M2,T1));
in(pch, xM3:bitstring, xMcs:bitstring, xM4:bitstring, xMT:bitstring,
xT2:bitstring));
new TS2:bitstring;
let zcs1 = xor(xM3,H(con(RIDu11,con(H(con(r1,tcu11)),T1)))) in
let RIDcs1 = xor(xMcs,H(con(zcs1,con(T1,xT2)))) in
let SKucs = H(con(RIDu11,con(H(con(r1,tcu11)),con(RIDcs,
con(Xui,con(zcs1, con(T1,xT2))))))) in
let M41 = H(con(SKucs,xT2)) in
if M41 = xM4 then
new T3:bitstring;
let TIDunew1 = xor(xMT,H(con(SKucs,con(zcs1,
con(Xui,con(T1,xT2)))))) in
let M5 = H(con(SKucs,con(TIDunew1,T3))) in out(pch,(M5,T3));
0
).
```

Fig. 6. Code in ProVerif for the process of user U_i.

source code for the process of user login, authentication and symmetric key-establishment with the cloud server CS_j. Fig. 7 shows the ProVerif source code for the cloud server authentication and shared key-establishment process with a user U_i.

Fig. 8 shows the ProVerif simulation results. The following observations can be drawn from the results:

- RESULT inj-event(UserAuth(id)) ==> inj-event (UserStart(id)) is true.
- RESULT not attacker(SKucs[]) is true.
- RESULT not attacker(SKcsu[]) is true.

From the result set mentioned above, we conclude that an attempt on attack for both session keys SKucs[] and SKcsu[] will fail. Thus, the session keys are safe and the proposed scheme passes the required security verification.

6. Comparative performance analysis

In this section, we provide the comparative analysis of BUAKA-CS and other existing schemes such as Ghaffar et al. [20], Tiwari et al. [19], Amin et al. [21], Zhou et al. [23] and Mo et al. [22]. The computation cost, communication cost, and security & functionality features were analyzed and compared.

```
(* —— server authentication starts —— *)
let SAuth =
in(pch,(mTIDu11:bitstring, mM1:bitstring, mM2:bitstring,
mT1:bitstring));
new TS1:bitstring; (* Received Timestamp *)
let rtcu = xor(mM1,Xui) in
let M21 = H(con(Xui,con(mT1,rtcu))) in

if M21 = mM2 then (* user authenticated *)
new T2:bitstring;
new r2:bitstring;
new TIDunew:bitstring;
let zcs = H(con(r2,TCcs)) in
let M3 = xor(zcs,(H(con(RIDui,con(rtcu,mT1))))) in
let Mcs = xor(RIDcs,H(con(zcs,con(mT1,T2)))) in
let SKcsu = H(con(RIDui,con(rtcu,con(RIDcs,con(Xui,con(zcs,
con(mT1,T2))))))) in
let M4 = H(con(SKcsu,T2)) in
let MT = xor(TIDunew,H(con(SKcsu,con(zcs,
con(Xui,con(mT1,T2)))))) in out(pch,(M3,Mcs,M4,MT,T2));

in(pch,(mM5:bitstring,mT3:bitstring));
let M51 = H(con(SKcsu,con(TIDunew,mT3))) in
if M51 = mM5 then
event UserAuth(IDu);
0.
process !User | !SAuth
```

Fig. 7. Code in ProVerif for the process of the cloud server CS_j.

```
– Query not attacker(SKucs[])
Completing...
Starting query not attacker(SKucs[])
RESULT not attacker(SKucs[]) is true.
– Query not attacker(SKcsu[])
Completing...
Starting query not attacker(SKcsu[])
RESULT not attacker(SKcsu[]) is true.
– Query inj-event(UserAuth(id)) ==> inj-event(UserStart(id))
Completing...
Starting query inj-event(UserAuth(id)) ==> inj-event(UserStart(id))
RESULT inj-event(UserAuth(id)) ==> inj-event(UserStart(id)) is true.
```

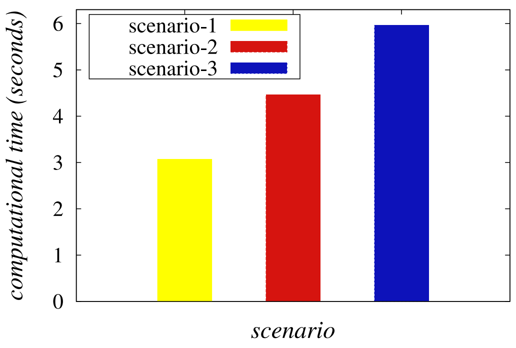
Fig. 8. Analysis of the ProVerif simulation results.

Table 2
Time required for various cryptographic primitives (rough estimate) [42].

Notation	Description (time to compute)	Approx. computation time (in seconds)
T_h	"One-way hash function"	0.00032
T_{ecm}	"ECC point multiplication"	0.0171
T_{eca}	"ECC point addition"	0.0044
T_{senc}	"Symmetric key encryption"	0.0056
T_{sdec}	"Symmetric key decryption"	0.0056
T_{me}	"Modular exponentiation"	0.0192
T_{fe}	"Fuzzy extractor function"	0.0171

6.1. Computation cost

Computation cost is analyzed in terms of the time taken (in seconds) by various cryptographic primitives for a specific protocol. The approximate time taken by various cryptographic primitives are given in Table 2.



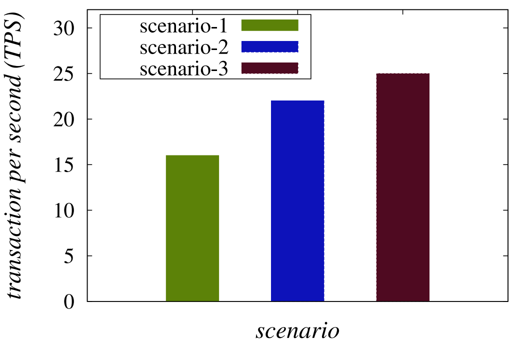


Fig. 9. (a) Influence on computational time (in seconds) (b) Influence on transaction per second (TPS).

Table 3
Comparison of computation costs.

Scheme	Computation cost (in milliseconds)
Ghaffar et al. [20]	$16T_h + 5T_{ecm} + T_{fe} \approx 107.72$ ms
Tiwari et al. [19]	$17T_h + 7T_{ecm} + T_{fe} \approx 142.24$ ms
Amin et al. [21]	$23T_h \approx 7.36$ ms
Zhou et al. [23]	$38T_h \approx 12.16$ ms
Mo et al. [22]	$13T_h + 6T_{ecm} \approx 106.76$ ms
BUAKA-CS	$22T_h + T_{fe} \approx 24.14$ ms

Table 4
Comparison of communication costs.

Scheme	No. of messages	No. of bits
Ghaffar et al. [20]	2	1376
Tiwari et al. [19]	2	1536
Amin et al. [21]	4	4416
Zhou et al. [23]	4	5856
Mo et al. [22]	3	1568
BUAKA-CS	3	2048

The comparisons of computation cost (in milliseconds) for different existing schemes and proposed BUAKA-CS are given in Table 3. The computation cost for the proposed solutions by Ghaffar et al. [20], Tiwari et al. [19], Amin et al. [21], Zhou et al. [23], Mo et al. [22], and proposed BUAKA-CS are $16T_h$+$5T_{ecm}$+$T_{fe} \approx 107.72$ ms, $17T_h$+$7T_{ecm}$+$T_{fe} \approx 142.24$ ms, $23T_h \approx 7.36$ ms, $38T_h \approx 12.16$ ms, $13T_h + 6T_{ecm} \approx 106.76$ ms and $22T_h + T_{fe} \approx 24.14$ ms, respectively. The computation cost of BUAKA-CS is less than the existing scheme except Amin et al. [21] and Zhou et al. [23]. However, on the contrary, BUAKA-CS provides more security and extra functionality as compared to the rest of them.

6.2. Communication costs

The comparison of communication cost for various exiting schemes and the proposed BUAKA-CS is given in Table 4. We assume that the "identity", "timestamp", "random number (nonce)", "elliptic curve point", and "hash output (if SHA-256 hashing algorithm is applied)" need 160 bits, 32 bits, 160 bits, 320 bits and 256 bits, respectively. The communication cost (in bits) for Ghaffar et al. [20], Tiwari et al. [19], Amin et al. [21], Zhou et al. [23], Mo et al. [22] and the proposed BUAKA-CS are 1376, 1536, 4416, 5856, 1568 and 2048, respectively. We note that the communication cost of proposed BUAKA-CS is less than the schemes of Amin et al. [21] and Zhou et al. [23]. However, the communication costs of proposed BUAKA-CS is more than Ghaffar et al. [20], Tiwari et al. [19] and Mo et al. [22]. But, again the proposed BUAKA-CS provides more security and extra features such as "protection against smart card/mobile device/smartphone stolen attack", "protection against privileged insider attack", "protection against stolen verifier attack", "availability of freely and locally password & biometric changing facility", "supports dynamic device addition phase", "availability of smart card/mobile device/smartphone revocation phase", "provides formal security analysis under the ROR model" and "provides blockchain-enabled security".

6.3. Comparison of security and functionality features

The comparisons of security and functionality features for various exiting schemes and proposed BUAKA-CS are provided in Table 5. The solution proposed by Ghaffar et al. [20], Tiwari et al. [19], Amin et al. [21], Zhou et al. [23] and Mo et al. [22] are vulnerable to various types of attacks and do not provide some of the important functionality features as discussed earlier. However, proposed BUAKA-CS is secure and supports most of the required functionality features as mentioned in Table 5. Therefore, BUAKA-CS can be a suitable match for cloud based crowdsourcing system.

7. Practical demonstration

To demonstrate the working of proposed BUAKA-CS, we developed a proof-of-concept using blockchain [28,29,43]. The details of various parameters used in the experimentation are provided in Table 6. The experimentation was preformed on a machine with Windows 10 64-bit OS with Intel (R) core i5-8250U CPU running at 1.60 GHz-1.80 GHz clock and RAM of size 8 GB. Furthermore, the eclipse IDE 2019–12 with Java language programming platform was utilized. We considered three different scenarios as follows. It is worth mentioning that there 5 computed and committed blocks in scenario-1, 10 in scenario-2, and 15 in scenario-3. We considered different number of users, i.e., 50, 100, and 150, in scenario-1, scenario-2 and scenario-3, respectively. We also considered four miner nodes (cloud servers). Additionally, we use voting based mechanism for block verification and its addition to the blockchain. The different fields used in a block of the implemented blockchain are summarized as follows. Block version is 32 bits, hash of previous block is 256-bits (in case of SHA-256), merkle tree root is 256-bits (for SHA-256), timestamps value is 32-bits, block's owner is 160-bits, owner's public key is 320-bits (for ECC) and encrypted transaction details in case of ECC-based encryption needs $(320 + 320) = 640$ bits. In case of 10 encrypted transactions, the payload of block is calculated as $(10 \times 640) = 64{,}00$ bits. Further, hash of the current block is 256-bits (for SHA-256). Moreover, the size of block signature is 320-bits (for ECDSA). Therefore, the total size of the block is estimated as 8,032 bits.

In the following, we discuss the obtained results.

7.1. Effect on computational time

The increasing number of blocks causes increase in the number of users and transactions. Its influence on BUAKA-CS was estimated as the computation time (in seconds) for contemplated scenarios. Various values of computational time (seconds) are 3.07 s for scenario-1, 4.46 s for scenario-2 and 5.96 s for scenario-3, respectively. The obtained results are given in Fig. 9(a). It is important to observe that the value of

Table 5
Security and functionality features comparison.

Feature	Ghaffar et al. [20]	Tiwari et al. [19]	Amin et al. [21]	Zhou et al. [23]	Mo et al. [22]	BUAKA-CS
ϕn_1	✓	✓	✓	✓	✓	✓
ϕn_2	✓	×	✓	✓	✓	✓
ϕn_3	✓	×	✓	✓	✓	✓
ϕn_4	×	×	×	×	×	✓
ϕn_5	✓	✓	✓	✓	✓	✓
ϕn_6	✓	✓	×	×	×	✓
ϕn_7	×	×	×	✓	×	✓
ϕn_8	×	×	×	✓	×	✓
ϕn_9	✓	✓	✓	✓	✓	✓
ϕn_{10}	×	×	×	×	×	✓
ϕn_{11}	×	×	×	×	✓	✓
ϕn_{12}	✓	✓	N/A	N/A	N/A	✓
ϕn_{13}	×	×	✓	✓	✓	✓
ϕn_{14}	×	×	×	×	×	✓
ϕn_{15}	×	×	✓	×	✓	✓
ϕn_{16}	✓	×	×	×	✓	✓
ϕn_{17}	×	×	✓	✓	×	✓
ϕn_{18}	×	×	×	×	×	✓

ϕn_1 : provides mutual authentication; ϕn_2 : protects user impersonation attack; ϕn_3 : protects server impersonation attack; ϕn_4 : supports anonymity and untraceability; ϕn_5 : protects man-in-the-middle attack; ϕn_6 : protects replay attack; ϕn_7 : protects smart card/mobile device/smartphone stolen attack; ϕn_8 : protects privileged insider attack; ϕn_9 : protects stolen verifier attack; ϕn_{10} : protects ESL attack under CK-adversary model; ϕn_{11} : provides session key agreement; ϕn_{12} : protects denial-of-Service (DoS) attack under biometric verification; ϕn_{13} : availability of freely and locally password & biometric changing facility; ϕn_{14} : supports dynamic device addition phase; ϕn_{15} : availability of smart card/mobile device/smartphone revocation phase; ϕn_{16} : provides "formal security analysis under standard Real-Or-Random (ROR) model"; ϕn_{17} : provides "formal security verification through AVISPA tool"; ϕn_{18} : provides blockchain-enabled security; ✓: "a scheme is secure or supports a functionality feature"; ×"a scheme is insecure or does not support a feature"; N/A: "not applicable".

Table 6
Simulation parameters and their values.

Parameter	Considered value
Followed platform	Windows 10 with 64-bit OS
Processor used	Intel (R) core (TM) with i5-8250U and 1.60 GHz-1.80 GHz
Random-access memory (RAM)'s size	8 GB
Platform of programming	Eclipse IDE 2019-12 with Java
Considered users	50, 100 and 150, in scenario-1, scenario-2 and scenario-3, respectively
Considered miner nodes (cloud servers)	4 for all scenarios
Block size	8,032 bits

computational time increases with increasing number of blocks having more users with more transactions. Occasionally, from scenario-1 to scenario-2 and scenario-2 to scenario-3 because it causes the "creation and addition of more number of blocks in the blockchain" which incurs additional computational time.

7.2. Effect on the transactions per second

The Influence of BUAKA-CS on transactions per second (TPS) is also estimated for different scenarios. Various obtained values of TPS are 16 for scenario-1, 22 for scenario-2 and 25 for scenario-3, respectively. Fig. 9(b) shows the obtained results. Further, note that the value of TPS increases with more number of blocks having more number of users and more number of transactions as the blockchain grows. That happens because growing blockchain causes the creation and addition of more number of blocks.

8. Conclusion

In this paper, we proposed a new blockchain-based user authentication and key agreement scheme for CrowdSourcing system (BUAKA-CS). The security of BUAKA-CS is proved through formal methods and also through other mathematical methods. The security analysis proves that BUAKA-CS is secure against different types of possible attacks. The formal security verification of BUAKA-CS using AVISPA tool is also performed which proves its robustness against "replay and man-in-the-middle attacks". BUAKA-CS is compared with the other closely related existing schemes which proves its effectiveness over other schemes. The proposed scheme is more security with less computation and communication costs. The pragmatic study of BUAKA-CS is also provided to measure its impact on the performance parameters of the system.

In future, we plan to work on further reducing the computation and communication costs of BUAKA-CS.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the anonymous reviewers, the associate editor and the editor-in-chief for their valuable feedback on the paper, which helped us to improve its quality and presentation. This work was supported by the Mathematical Research Impact Centric Support (MATRICS) project funded by the Science and Engineering Research Board (SERB), India (Reference No. MTR/2019/000699) and also by the Ripple Centre of Excellence Scheme, CoE in Blockchain (Sanction No. IIIT/R&D Office/Internal Projects/001/2019), IIIT Hyderabad, India.

References

[1] J. Howe, The rise of crowdsourcing, Wired Mag. 53 (10) (2006) 1–4.
[2] M. Li, J. Weng, A. Yang, W. Lu, Y. Zhang, L. Hou, J. Liu, Y. Xiang, R.H. Deng, CrowdBC: A blockchain-based decentralized framework for crowdsourcing, IEEE Trans. Parallel Distrib. Syst. 30 (6) (2019) 1251–1266.
[3] P. Srivastava, A. Mostafavi, Challenges and opportunities of crowdsourcing and participatory planning in developing infrastructure systems of smart cities, Infrastructures 3 (4) (2018) 51.
[4] K. Kearns, 9 great examples of crowdsourcing in the age of empowered consumers, 2013, https://tweakyourbiz.com/marketing/9-great-examples-crowdsourcing-age-empowered-consumers. Accessed on April 2020.
[5] G. Zhuo, Q. Jia, L. Guo, M. Li, P. Li, Privacy-preserving verifiable set operation in big data for cloud-assisted mobile crowdsourcing, IEEE Internet Things J. 4 (2) (2017) 572–582.

[6] K. Yang, K. Zhang, J. Ren, X. Shen, Security and privacy in mobile crowdsourcing networks: challenges and opportunities, IEEE Commun. Mag. 53 (8) (2015) 75–81.
[7] Online medical diagnosis, differential diagnosis, 2020, https://www.crowdmed.com/. Accessed on April 2020.
[8] D. McCarty, Top 4 crowdsourcing challenges, 2021, https://www.onespace.com/blog/2017/08/top-4-crowdsourcing-challenges/. Accessed on April 2021.
[9] M. Wazid, A.K. Das, V. Odelu, N. Kumar, W. Susilo, Secure remote user authenticated key establishment protocol for smart home environment, IEEE Trans. Dependable Secure Comput. 17 (2) (2020) 391–406.
[10] M. Wazid, A.K. Das, J.J.P.C. Rodrigues, S. Shetty, Y. Park, Iomt malware detection approaches: Analysis and research challenges, IEEE Access 7 (2019) 182459–182476.
[11] Z. Wang, X. Pang, Y. Chen, H. Shao, Q. Wang, L. Wu, H. Chen, H. Qi, Privacy-preserving crowd-sourced statistical data publishing with an untrusted server, IEEE Trans. Mob. Comput. 18 (6) (2019) 1356–1367.
[12] V. Kumar, H. Li, J.M.J. Park, K. Bian, Crowd-sourced authentication for enforcement in dynamic spectrum sharing, IEEE Trans. Cogn. Commun. Netw. 5 (3) (2019) 625–636.
[13] G. Sun, S. Sun, J. Sun, H. Yu, X. Du, M. Guizani, Security and privacy preservation in fog-based crowd sensing on the internet of vehicles, J. Netw. Comput. Appl. 134 (2019) 89–99.
[14] W. Wang, D.T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, D.I. Kim, A survey on consensus mechanisms and mining strategy management in blockchain networks, IEEE Access 7 (2019) 22328–22370.
[15] J. Wang, M. Li, Y. He, H. Li, K. Xiao, C. Wang, A blockchain based privacy-preserving incentive mechanism in crowdsensing applications, IEEE Access 6 (2018) 17545–17556.
[16] M. Yang, T. Zhu, K. Liang, W. Zhou, R.H. Deng, A blockchain-based location privacy-preserving crowdsensing system, Future Gener. Comput. Syst. 94 (2019) 408–418.
[17] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi, W. Dou, A blockchain-powered crowdsourcing method with privacy preservation in mobile environment, IEEE Trans. Comput. Soc. Syst. 6 (6) (2019).
[18] D. Dasgupta, J.M. Shrein, K.D. Gupta, A survey of blockchain from security perspective, J. Bank. Financial Technol. (Springer) 3 (2019) 1–17.
[19] D. Tiwari, G.K. Chaturvedi, G.R. Gangadharan, ACDAS: Authenticated controlled data access and sharing scheme for cloud storage, Int. J. Commun. Syst. 32 (15) (2019) e4072.
[20] Z. Ghaffar, S. Ahmed, K. Mahmood, S.H. Islam, M.M. Hassan, G. Fortino, An improved authentication scheme for remote data access and sharing over cloud storage in cyber-physical-social-systems, IEEE Access 8 (2020) 47144–47160.
[21] R. Amin, N. Kumar, G. Biswas, R. Iqbal, V. Chang, A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment, Future Gener. Comput. Syst. 78 (2018) 1005–1019.
[22] J. Mo, Z. Hu, H. Chen, W. Shen, An efficient and provably secure anonymous user authentication and key agreement for mobile cloud computing, Wirel. Commun. Mobile Comput. 2019 (2019) 1–12.
[23] L. Zhou, X. Li, K.-H. Yeh, C. Su, W. Chiu, Lightweight IoT-based authentication scheme in cloud computing circumstance, Future Gener. Comput. Syst. 91 (2019) 244–251.
[24] Z. Duan, M. Yan, Z. Cai, X. Wang, M. Han, Y. Li, Truthful incentive mechanisms for social cost minimization in mobile crowdsourcing systems, Sensors 16 (4) (2016) 481.
[25] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inform. Theory 29 (2) (1983) 198–208.
[26] T.S. Messerges, E.A. Dabbish, R.H. Sloan, Examining smart-card security under the threat of power analysis attacks, IEEE Trans. Comput. 51 (5) (2002) 541–552.
[27] R. Canetti, H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, in: B. Pfitzmann (Ed.), Advances in Cryptology — EUROCRYPT, Springer Berlin Heidelberg, Innsbruck (Tyrol), Austria, 2001, pp. 453–474.
[28] M. Wazid, A.K. Das, S. Shetty, M. Jo, A tutorial and future research for building a blockchain-based secure communication scheme for internet of intelligent things, IEEE Access 8 (2020) 88700–88716.
[29] N. Garg, M. Wazid, A.K. Das, D.P. Singh, J.J.P.C. Rodrigues, Y. Park, BAKMP-IoMT: Design of blockchain enabled authenticated key management protocol for internet of medical things deployment, IEEE Access (2020) 1–23, http://dx.doi.org/10.1109/ACCESS.2020.2995917.
[30] B. Bera, D. Chattaraj, A.K. Das, Designing secure blockchain-based access control scheme in IoT-enabled internet of drones deployment, Comput. Commun. 153 (2020) 229–249.
[31] H. Zhang, J. Wang, Y. Ding, Blockchain-based decentralized and secure keyless signature scheme for smart grid, Energy 180 (2019) 955–967.
[32] X. Wang, P. Zeng, N. Patterson, F. Jiang, R. Doss, An improved authentication scheme for internet of vehicles based on blockchain technology, IEEE Access 7 (2019) 45061–45072.
[33] M. Abdalla, P.A. Fouque, D. Pointcheval, Password-based authenticated key exchange in the three-party setting, in: 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science, Vol. 3386, Les Diablerets, Switzerland, 2005, pp. 65–84.
[34] R. Canetti, H. Krawczyk, Universally composable notions of key exchange and secure channels, in: L.R. Knudsen (Ed.), Advances in Cryptology — EUROCRYPT, Springer Berlin Heidelberg, Amsterdam, The Netherlands, 2002, pp. 337–351.
[35] D. Wang, H. Cheng, P. Wang, X. Huang, G. Jian, Zipf's law in passwords, IEEE Trans. Inf. Forensics Secur. 12 (11) (2017) 2776–2791.
[36] K. Park, Y. Park, A.K. Das, S. Yu, J. Lee, Y. Park, A dynamic privacy-preserving key management protocol for V2G in social internet of things, IEEE Access 7 (2019) 76812–76832.
[37] AVISPA, Automated validation of internet security protocols and applications, 2019, http://www.avispa-project.org/. Accessed on October 2019.
[38] M. Abadi, B. Blanchet, H. Comon-Lundh, Models and proofs of protocol security: A progress report, in: 21st International Conference on Computer Aided Verification (CAV'09), Grenoble, France, 2009, pp. 35–49.
[39] M. Bhattacharya, S. Roy, K. Mistry, H.P.H. Shum, S. Chattopadhyay, A privacy-preserving efficient location-sharing scheme for mobile online social network applications, IEEE Access 8 (2020) 221330–221351.
[40] D. Abbasinezhad-Mood, M. Nikooghadam, Efficient anonymous password-authenticated key exchange protocol to read isolated smart meters by utilization of extended Chebyshev chaotic maps, IEEE Trans. Ind. Inf. 14 (11) (2018) 4815–4828.
[41] Z. Xu, C. Xu, H. Chen, F. Yang, A lightweight anonymous mutual authentication and key agreement scheme for WBAN, Concurr. Comput.: Pract. Exper. 31 (14) (2019) e5295.
[42] D. He, N. Kumar, J.H. Lee, R.S. Sherratt, Enhanced three-factor security protocol for consumer USb mass storage devices, IEEE Trans. Consum. Electron. 60 (1) (2014) 30–37.
[43] M. Fan, X. Zhang, Consortium blockchain based data aggregation and regulation mechanism for smart grid, IEEE Access 7 (2019) 35929–35940.

Mohammad Wazid received M.Tech. degree in Computer Network Engineering from Graphic Era deemed to be University, Dehradun, India and Ph.D. degree in Computer Science and Engineering from the International Institute of Information Technology, Hyderabad, India. He is currently working as an Associate Professor in the Department of Computer Science and Engineering, Graphic Era deemed to be University, Dehradun, India. He is also the head of Cybersecurity and IoT research group at Graphic Era deemed to be University, Dehradun, India. He was also a postdoctoral researcher at cyber security and networks lab, Innopolis University, Innopolis, Russia. His current research interests include information security, remote user authentication, Internet of things (IoT), cloud/fog/edge computing and blockchain. He has published more than 90 papers in international journals and conferences in the above areas.

Ashok Kumar Das received a Ph.D. degree in computer science and engineering, an M.Tech. degree in computer science and data processing, and an M.Sc. degree in mathematics from IIT Kharagpur, India. He is currently an Associate Professor with the Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad, India. His current research interests include cryptography, system and network security, security in vehicular ad hoc networks, smart grids, smart homes, Internet of Things (IoT), Internet of Drones, Internet of Vehicles, Cyber-Physical Systems (CPS) and cloud computing, intrusion detection, blockchain and AI/ML security. He has authored over 280 papers in international journals and conferences in the above areas, including over 240 reputed journal papers. He was a recipient of the Institute Silver Medal from IIT Kharagpur. He has been included in the subject-wise ranking of top 2% scientists from India (all fields) in the area of Networking & Telecommunications, with Rank world-wide (by subject area): 321. He has been also listed in the top H-Index for Scientists in the World for Computer Science and Electronics database maintained by Guide2Research (http://www.guide2research.com/). He is on the editorial board of IEEE Systems Journal, Journal of Network and Computer Applications (Elsevier), Computer Communications (Elsevier), Journal of Cloud Compting (Springer), Cyber Security and Applications (Elsevier), IET Communications, KSII Transactions on Internet and Information Systems, and International Journal of Internet Technology and Secured Transactions (Inderscience), and has served as a Program Committee Member in many international conferences.

He severed as one of the Technical Program Committee Chairs of the first International Congress on Blockchain and Applications (BLOCKCHAIN'19), Avila, Spain, June 2019, International Conference on Applied Soft Computing and Communication Networks (ACN'20), October 2020, Chennai, India, and second International Congress on Blockchain and Applications (BLOCKCHAIN'20), L'Aquila, Italy, October 2020. He also served as an advisory board member of 3rd International Congress on Blockchain and Applications (BLOCKCHAIN'21), Salamanca, Spain, 6–8 October, 2021. His Google Scholar h-index is 60 and i10-index is 178 with over 10,700 citations. He is a senior member of the IEEE.

Rasheed Hussain is currently working as an Associate Professor and the Director of Institute of Information Security and Cyber-Physical Systems at Innopolis University, Innopolis, Russia. He is also the Director of Networks and Blockchain Lab at Innopolis University and serves as an ACM Distinguished Speaker. Previously, he was with University of Amsterdam, Netherlands, and Hanyang University, South Korea. He is a senior member of IEEE, member of ACM, and serves as editorial board member for various journals including IEEE Access, IEEE Internet Initiative, Cluster Computing, Springer, Internet Technology Letters, Wiley, and serves as reviewer for most of the IEEE transactions, Springer and Elsevier Journals. He also serves as chair and technical program committee member of various conferences such as IEEE ICC, IEEE VTC, IEEE VNC, IEEE Globecom, IEEE ICCVE, ICCCN, and so on. Currently, he is working on machine and deep learning for IoT security and API security.

Neeraj Kumar received his Ph.D. in CSE from Shri Mata Vaishno Devi University, Katra (J & K), India, and was a postdoctoral research fellow in Coventry University, Coventry, UK. He is working as a Professor in the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology (Deemed to be University), Patiala (Pb.), India. He has published more than 600 technical research papers in leading journals and conferences from IEEE, Elsevier, Springer, John Wiley etc. He is on the editorial board of ACM Computing Survey, IEEE Transactions on Sustainable Computing, IEEE Network Magazine, IEEE Communication Magazine, Journal of Networks and Computer Applications (Elsevier), Computer Communications (Elsevier), International Journal of Communication Systems (Wiley), and Security and Privacy (Wiley).

Sandip Roy received his Ph.D. degree in Computer Science and Engineering from Jadavpur University, Kolkata, India. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering of Asansol Engineering College, India. He received an M.Tech degree in Computer Science and Technology from West Bengal University of Technology, India. His current research interests include cryptography and network security. He has published several international journal and conference papers in his research areas.