Blockchain-Based User Authentication and Data-Sharing Framework for Healthcare Industries

Preeti Soni, SK Hafizul Islam, *Senior Member, IEEE*, Arup Kumar Pal, Nimish Mishra, and Debabrata Samanta, *Senior Member, IEEE*

***Abstract*—In this paper, we developed a Blockchain-based User Authentication Data-Sharing (BC-UADS) framework. In BC-UADS, several hospital servers form a consortium blockchain network to maintain the transparency, immutability, and authenticity of the patient's Electronic Healthcare (Record EHR) medical data. The BC-UADS framework allows doctors to share or retrieve a patient's EHR metadata from the blockchain network. Since, the metadata is stored on a blockchain platform, it is more secure and trusted for real-time applications to utilize the medical data. The data-sharing protocol of the BC-UADS framework is implemented based on Proof-of-Reputation consensus algorithm in a blockchain network. The BC-UADS framework is analyzed in the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool, demonstrating that it is secure against active and passive attacks. Besides, the BC-UADS framework is provably secure in the random oracle model based on the hardness assumption of Elliptic Curve-based Computational Diffie-Hellman (ECCDH) problem. The mutual authentication property of the BC-UADS framework is analyzed in the BAN (Burrows-Abadi-Needham) logic model. We have computed the communication, execution, and storage costs of the BC-UADS framework in different security levels: 80-bit, 112-bit, 128-bit, 192-bit, and 256-bit using PBC library. The proposed BC-UADS framework is compared with the state-of-the-art schemes.**

***Index Terms*—Authentication, blockchain, healthcare, provable security, proof-of-reputation.**

I. Introduction and Literature Review

HEALTHCARE industries has evolved to provide health-related services online to legitimate users, say U_i. Legitimate users can access the services provided they pass the authentication process through valid login credentials. However, the users like doctors (D_i) or patients (P_t) who access the services are associated with a particular hospital/healthcare center through the registration process. A doctor D_i uses this service to upload the medical report of a patient P_t into the hospital server, say HS_i, while P_t uses the service to access his medical report or prescription from HS_i. There may be multiple hospitals where P_t can visit to get better treatment facilities. Assume that P_t registers to H_i and gets treatment from there, and at a later stage, P_t visits another hospital H_j. In that situation, H_j needs the previous medical reports, say EHR_t, of P_t. Note that EHR_t of P_t is stored at the hospital server HS_i, which is maintained by H_i. In such a scenario, the healthcare industry must share healthcare data among hospitals for further clinical investigation, studies, research, and informatics. Since, there is a lack of a communication platform among the hospitals, one common approach is that P_t login into its previous hospital server HS_i and get EHR_t. After that P_t login at HS_j, which is maintained by H_j, and after the session key generation P_t will be able to share EHR_t to HS_j. Here, verifying EHR_t produced by P_t is a challenging task for H_j. So, the hospital servers must achieve transparency in medical data-sharing to provide error-free treatment. Further, the process is frustrating for P_t as he/she has to login separately for each hospital server. Therefore, the requirement is to overcome the obstacle by making a common platform wherein P_t does not need to bother about the login process for each hospital.

Blockchain technology is an emerging data-sharing platform among cloud servers, and it can be used in applications requiring data integrity, immutability, and transparency [1]. Its distributed and decentralized architecture helps to maintain an identical copy of data on all the servers. This system eliminates the control of the central server and allows the other party to verify the data integrity by themselves. This property of blockchain can be utilized in the healthcare system to maintain the trustworthiness of the system. However, the challenge is that before publishing the data block in the blockchain network, the data must be verified and validated. So, the role of blockchain architecture and consensus algorithm selection is an important task. Therefore, in this work, we have designed an authentication scheme that can collaborate with blockchain technology to provide secure data-sharing with data confidentiality and integrity. Omar et al. [2] have mentioned that decentralizing medical data stored in a cloud server can minimize the risk of cyberattacks. Cao et al. [3] proposed an eHealth system for tamper-proof EHR using the

Manuscript received 16 April 2023; revised 23 November 2023; accepted 14 March 2024. Date of publication 26 March 2024; date of current version 12 June 2024. This work was supported by Rochester Institute of Technology, Kosovo, Europe under faculty research fund. Recommended for acceptance by Dr. Bin Xiao. *(Corresponding author: SK Hafizul Islam.)*

Preeti Soni is with the Department of Computer Science and Engineering, National Institute of Technology Hamirpur, Hamirpur 177005, India (e-mail: pritusoni.28@gmail.com).

SK Hafizul Islam and Nimish Mishra are with the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India (e-mail: hafi786@gmail.com; neelam.nimish@gmail.com).

Arup Kumar Pal is with the Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India (e-mail: arupkrpal@iitism.ac.in).

Debabrata Samanta is with the Department of Computing and Information Technologies, Rochester Institute of Technology, Prishtina 10000, Kosovo, Europe (e-mail: debabrata.samanta369@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNSE.2024.3381723, provided by the authors.

Digital Object Identifier 10.1109/TNSE.2024.3381723

TABLE I
COMPARATIVE ANALYSIS OF THE STATE-OF-THE ART DATA-SHARING SCHEMES

Scheme	Tools/Methods Used	Service Provided	Limitations
[2]	Elliptic curve, Smartcard	Data Upload and retrieval	No data-sharing mechanism
[3]	Bilinear pairing, AES, Diffie-Hellman Key Exchange	Data upload and data retrieval	Password guessing and Man-in the-middle attacks
[5]	Bilinear pairing-based signature	Data verification and validation	No data confidentiality
[6]	Elliptic curve, Multiserver authentication, Smartcard, Hyperledger fabric	Authentication	No data-sharing
[7]	Blind signature, Certificateless Authenticated Key Agreement	Data transmission and data storage	No data-sharing
[8]	Biometric-based authentication	User authentication and data-sharing	The details of achieving authentication and blockchain related process are not included
[9]	Bilinear pairing	Metadata storage and sharing	At the time of registration phase each hospital (A) provides the details of record required from peer hospital (B) which create scalability issue.
[10]	Elliptic curve, Hash function	Data storage and retrieval	No data-sharing
[11]	Elliptic curve, Signature scheme	Multiserver authentication	No data-sharing
[12]	Symmetric key encryption for data privacy	Inter-hospital data-sharing	Key pair and health information of user is stored in public blockchain
[13]	Elliptic curve, Private blockchain	Authentication and key sharing	Designed for private blockchain, No data-sharing
[14]	Elliptic curve, Hash function	Data storage in blockchain	Authentication and data retrieval is not discussed in detail
[15]	Smartcontarct, Public blockchain, PoW	Certificate generation, verification and data access	No data sharing concept
[16]	Smart-contract, Public blockchain, PoW	User registration, Data upload	No data-sharing

Ethereum blockchain. Several other authentication schemes with data-sharing techniques have been proposed for healthcare industries [4], [5], [6], [7], [8], [9], [10]. Xiang et al. [11] proposed a blockchain-based multiserver authentication scheme using the Proof-of-Stake (PoS) consensus algorithm. Yazdinejad et al. [12] proposed a decentralized authentication scheme for blockchain-based hospital networks. Garg et al. [13] proposed a blockchain-enabled authenticated key management protocol using the ripple consensus algorithm for block generation and validation. Singh et al. [14] proposed a trust management scheme for smart enterprises. They have applied machine learning techniques with augmented intelligence for data authentication. Sharma et al. [15] gives an overview of a patient's medical certificate generation, storage, access, and verification process using a public blockchain. Rai [16] proposed a blockchain-enabled eHealth system. This system explained the efficient user registration and data upload process in the blockchain.

Wang et al. [17] proposed a handover authentication scheme for an intelligent telehealthcare system using consortium blockchain. Since, the healthcare application shares highly sensitive data, a public blockchain is not feasible for secure storage. Therefore, the private blockchain may be useful in this case. However, if the data is stored in a private blockchain, then it will be controlled by a central authority, violating the working principle of blockchain technology. Apart from that, the selection of a consensus algorithm is a decidable factor for the proper functioning of the medical blockchain network. In the Proof-of-Work (PoW) consensus mechanism, a minor must solve an NP-hard problem, for which minors must invests huge computation power and time. Similarly, minors must put their stake in the PoS consensus algorithm. So, as a motivation, minor will require some form of money as a reward, increasing the cost of operating the blockchain. Therefore, it is desirable that blockchain must have a least expensive incentive mechanism that motivates the hospital servers to participate in the mining process so that the blockchain network functions continuously. In the Proof-of-Reputation (PoR) consensus algorithm, the incentive for each participant is the recognition they will get by performing valid transactions. So, the node that will mine the next block to the blockchain is predicted according to the reputation. Hence, the transaction and validation rate is faster in PoR than the PoW and PoS [18]. In Table I, we reviewed and compared the existing data-sharing scheme proposed for healthcare industries.

A. Motivations and Contributions

Based on the above discussions, we found that a Blockchain-based User Authentication and Data-Sharing (BC-UADS) scheme is needed to provide service for inter-hospital networks. Blockchain technology requires an efficient consensus algorithm that is faster and less complex. Making reputation as an incentive, every hospital is motivated to cooperate with the blockchain network, which is mandatory for the continuous execution of the transaction. The main contributions of this paper are as follows.

- We have proposed the BC-UADS framework using an elliptic curve and three-factor authentication technology to ensure the confidentiality of patients' EHR data shared with different hospital servers.
- We have used consortium blockchain to provide a distributed and decentralization data-sharing platform in a permission network. Also, we have used PoR consensus algorithm, which is more efficient than PoW and PoS.
- We have verified the mutual authentication property of the BC-UADS through the BAN logic model. In addition, the AVISPA simulation shows that the BC-UADS framework is secure against active and passive attacks.

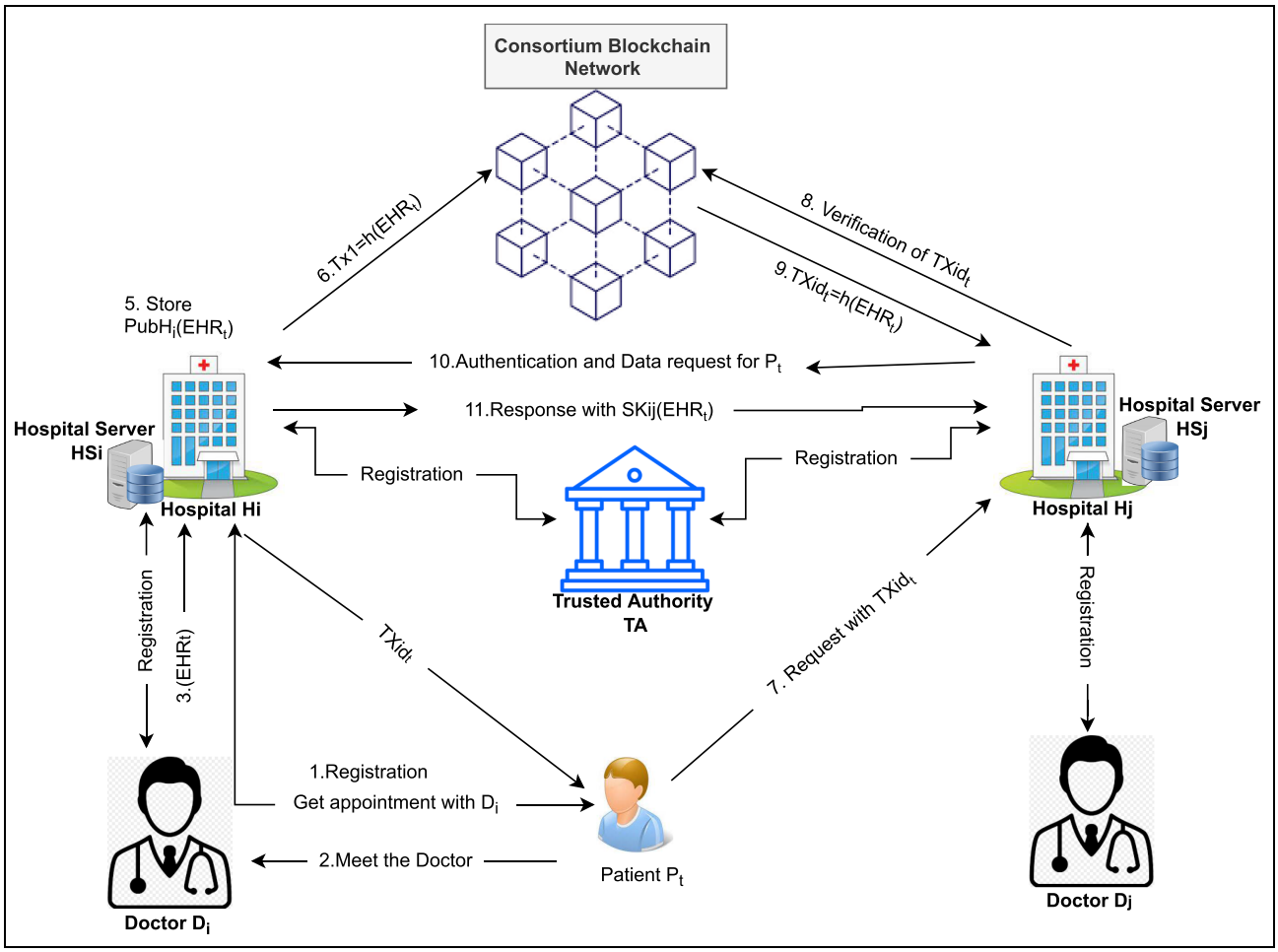

Fig. 1. Blockchain-based user authentication and data-sharing framework.

- We have analyzed the provable security of the BC-UADS framework based on the hardness assumption of the Elliptic curve-based Computational Diffie-Hellman (ECCDH) problem in the random oracle model (ROM).
- We have computed the execution costs using PBC library, and communication cost of the BC-UADS framework for different security levels: 80-bit, 112-bit, 128-bit, 192-bit, and 256-bit on a Laptop (RAM: 8 GB, Intel(R) Core(TM) i5-1035G1 CPU@ 1.19 GHz, Ubuntu 22.04LTS OS) as client, and on a Desktop (RAM: 8 GB, Intel(R) Core(TM) i7-10700 CPU@ 2.90 GHz, Ubuntu 22.04LTS OS) as server. We also estimated the storage costs of the proposed BC-UADS framework. The BC-UADS framework is compared with the state-of-the-art schemes.
- The data-sharing phase of the BC-UADS framework is implemented based on the reputation-based consensus algorithm in a consortium blockchain. The time to generate and verify the 1000 blocks of data is estimated to 33 seconds and 38 minutes, respectively.

B. Paper Organization

The remaining part of this paper is organized as follows. Section II describes the network model for BC-UADS framework. Section III describes the proposed BC-UADS framework in detail. Section IV explains the the process of storing the metadata of a patient in a consortium blockchain. Section V provides the formal security analysis of the BC-UADS framework. Section VI provides the comparative analysis of the communication and execution costs for the BC-UADS framework and other schemes. Section VII simulates the data-sharing technique of the BC-UADS framework in a consortium blockchain network. Section VIII concludes the paper.

II. Proposed System Model

The proposed network model for the BC-UADS framework has been shown in Fig. 1. The role of different entities involved in this model is described below.

1) *Trusted Authority (TA):* The TA generates all the public parameters required to execute different phases of the BC-UADS framework successfully. In addition, the TA approves the public keys of all the hospital servers.
2) *Hospital Server (HS_i):* We considered the number of hospital servers to be n. A hospital server HS_i $(1 \leq i \leq n)$ maintained at the hospital H_i will register and get a pair of private-public key from the TA.

TABLE II
NOTATIONS AND THEIR MEANING

Notation	Meaning
TA	Trusted authority
s/P_{ta}	Private/Public key of TA
H_i	i^{th} hospital, $1 \leq i \leq n$
HS_i	i^{th} server maintained by H_i , $1 \leq i \leq n$
n	Number of hospitals
ID_i	Identity of HS_i
P_t	t^{th} patient, $1 \leq t \leq m$
m	Number of patients registers at a hospital H_i
p	A large prime number
$k_t/k_i/k_j$	Private key of P_t, H_i and H_j
EHR_t	Electronic health report of P_t
ID_t/SC_t	Identity/Personalize smartcard of P_t
PW_t/Bio_t	Password/Biometric of P_t
D_i	Doctor working at H_i
Tr_{ij}	Rating transaction: rating given by H_j to H_i
Gen$(\cdot)$	Biometric key ***generation function*** used in fuzzy extractor
Rep$(\cdot)$	Biometric key ***reproduction function*** used in fuzzy extractor
σ_t	Biometric key generated from Bio_t
τ_t	Public reproduction parameter obtained from Bio_t
$\oplus$,$\|$	Bitwise XOR, Concatenation operators
$h(\cdot)$	Hash function

3) *User:* We have considered that the doctors (D_i) and the patients (P_t) are the users of H_i. All the users register at HS_i and get a fresh personalized smartcard. P_t can visit more than one hospital, so he/she needs to perform mutual authentication between two hospital servers HS_i and HS_j before sharing his/her medical data EHR_t.
4) *Blockchain Network:* All HS_i ($1 \leq i \leq n$) will agree on a consortium blockchain network to share their medical data related to P_t and take part in the execution of consensus algorithm to validate and add a data block into the blockchain network.

In this framework, HS_i is local storage server maintained at a hospital H_i. All HS_i will register to TA via a secure channel (offline mode). All the user (D_i/P_t) resisters to their HS_i via secure channel and obtains a personalized smartcard. The smartcard holds some secret credentials for further login to HS_i over an insecure channel. We have presented the authentication process for two different situations. In the first scenario, when a user U_i of hospital H_i performs login and authentication with H_i. Here we consider two types of users, the doctors D_i working at hospital H_i and P_t who once visited H_i. The doctor (D_i) performs authentication and session key agreement with HS_i and uploads the data into HS_i using the shared session key. P_t performs login and authentication with HS_i and gets EHR_t report with the shared session key. In the second scenario, the users are only P_t who once visit H_i. After some time, P_t visits to another hospital H_j, which requires the previous EHR_t report of P_t stored at HS_i. In that case, a three-party authentication will be performed between P_t, HS_i, and HS_j, which is shown in Section III-C2. The proposed BC-UADS framework is the combination of two frameworks: first is authentication framework (Section III) which has system initialization, hospital server registration, patient registration, login and authentication phase. The second framework is blockchain construction and addition in a blockchain (Section IV). The notations used in this work are presented in Table II.

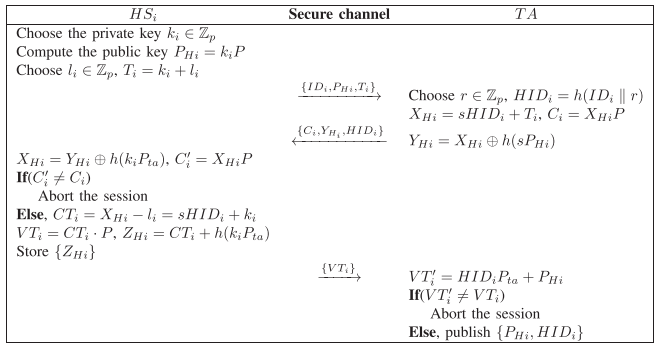

Fig. 2. Hospital server registration.

III. PROPOSED BC-UADS FRAMEWORK

This framework is the description of the authentication process performed by the various entity. This framework has the following steps: system initialization, hospital server registration, patient registration, login and authentication phase which is described below:

A. System Initialization Phase

TA executes this phase and generates all the public parameters, and private and public key pairs as follows:

1) TA chooses an elliptic curve $E(\mathbb{Z}_p) : y^2 \equiv (x^3 + ax + b) \text{mod} p$ over a prime field $\mathbb{Z}_p$.
2) TA selects a base point (generator) P of $E(\mathbb{Z}_p)$.
3) TA selects $s \in \mathbb{Z}_p$ as a private key, and computes the public key as $P_{ta} = sP$.
4) TA selects a collision-resistant hash function $h(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^\ell$, where $p \approx 2^\ell$, and a fuzzy extractor $\{\mathbf{Gen}(\cdot), \mathbf{Rep}(\cdot)\}$.
5) TA discloses $\{E(\mathbb{Z}_p), P, h(\cdot), p, P_{ta}\}$ as public parameter and keeps s secret.

B. Registration Phase

In this phase, we have described the registration process of (i) a hospital server HS_i to TA, and (ii) a patient P_t to HS_i.

1) Hospital Server Registration: In this phase, a new HS_i ($1 \leq i \leq n$) managed at H_i resisters to TA via a secure channel. Fig. 2 further depicts this phase.

1) First, HS_i chooses $k_i \in \mathbb{Z}_p$ as the private key and computes the corresponding public key as $P_{Hi} = k_i P$. HS_i also chooses $l_i \in \mathbb{Z}_p$, computes $T_i = k_i + l_i$, and sends $\{ID_i, P_{Hi}, T_i\}$ to TA via a secure channel.
2) Next, TA chooses $r \in \mathbb{Z}_p$, and computes $HID_i = h(ID_i \| r)$, $X_{Hi} = sHID_i + T_i$, $C_i = X_{Hi}P$, $Y_{Hi} = X_{Hi} \oplus h(sP_{Hi})$ for HS_i, and sends $\{C_i, Y_{Hi}, HID_i\}$ to HS_i through a secure channel.
3) Now, HS_i computes $X_{Hi} = Y_{Hi} \oplus h(k_i P_{ta})$, $C_i' = X_{Hi}P$, and verifies whether $C_i' \stackrel{?}{=} C_i$ holds. If this condition is true, HS_i computes $CT_i = X_{Hi} - l_i = sHID_i + k_i$, and sends $VT_i = CT_i P$ to TA over a secure channel.

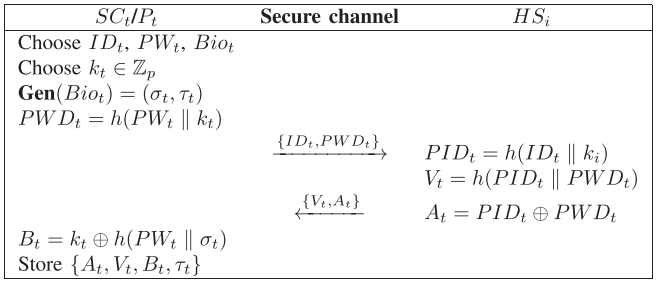

Fig. 3. User (Patient) registration.

HS_i further computes $Z_{Hi} = CT_i + h(k_i P_{ta})$, and stores $\{Z_{Hi}\}$ in its database.

4) Finally, TA computes $VT_i' = HID_i P_{ta} + P_{Hi}$, and verifies whether $VT_i' \stackrel{?}{=} VT_i$ hold. If is true, TA publishes HID_i, and P_{Hi} as the identity and public key of HS_i.

2) User Registration: We have considered that the user is a new patient P_t who registers at HS_i. The following steps can be executed by P_t $(1 \leq t \leq m)$ and HS_i over a secure channel. Fig. 3 further depicts this phase.

1) First, P_t chooses his/her identity ID_t, password PW_t, and imprint the biometric data Bio_t to the biometric-scanner device. P_t also selects $k_t \in \mathbb{Z}_p$, computes $\mathbf{Gen}(Bio_t) = (\sigma_t, \tau_t)$, $PWD_t = h(PW_t \| k_t)$, and sends $\{ID_t, PWD_t\}$ to HS_i over a secure channel.
2) Next, HS_i computes $PID_t = h(ID_t \| k_i)$, $V_t = h(PID_t \| PWD_t)$, $A_t = PID_t \oplus PWD_t$. HS_i issues a new smartcard SC_t, and writes $\{V_t, A_t\}$ to its memory. Finally, HS_i sends SC_t to P_t over a secure channel.
3) Finally, P_t computes $B_t = k_t \oplus h(PW_t \| \sigma_t)$ and replaces $\{V_t, A_t\}$ by $\{A_t, V_t, B_t, \tau_t\}$ in SC_t.

The same process is followed by the doctor D_i to register at H_i, which we have not described here explicitly.

C. Login and Authentication Phase

Here, we have described the process of login and authentication between (i) P_t and HS_i, and (ii) P_t, HS_i and HS_j.

1) Login and Authentication Between P_t and HS_i: Here, we have assumed that a patient P_t wants to access his/her medical data EHR_t stored at HS_i. The following steps describe the login and authentication process between P_t and HS_i. Fig. 4 further depicts this phase.

1) P_t inputs the identity ID_t, password PW_t to SC_t, and the biometric Bio_t to the biometric-scanner device. SC_t computes $\mathbf{Rep}(Bio_t, \tau_t) = \sigma_t'$, $k_t' = B_t \oplus h(PW_t \| \sigma_t')$, $PWD_t' = h(PW_t \| k_t')$, $PID_t' = A_t \oplus PWD_t'$, $V_t' = h(PID_t' \| PWD_t')$. Next, SC_t verifies whether $V_t' \stackrel{?}{=} V_t$ holds. If the verification fails, SC_t/P_t aborts the session; otherwise, chooses $a \in \mathbb{Z}_p$, computes $A = aP$, $R_t = aP_{Hi} = ak_iP$, $M_1 = ID_t \oplus h(R_t)$, $V_1 = h(PID_t \| R_t \| HID_i \| ID_t)$, and sends the login request message $\{M_1, V_1, A\}$ to HS_i over a public channel.

Fig. 4. Login and mutual authentication between P_t and HS_i.

2) Next, HS_i computes $R_t' = k_iA = ak_iP$, $ID_t' = M_1 \oplus h(R_t')$, $PID_t' = h(ID_t' \| k_i)$, $V_1' = h(PID_t' \| R_t' \| HID_i \| ID_t')$, and verifies whether $V_1 \stackrel{?}{=} V_1'$ holds. If the verification fails, HS_i aborts the session; otherwise, chooses $d \in \mathbb{Z}_p$, computes $B = dP$, $dA = daP$, $SK_{it} = h(HID_i \| ID_t' \| dA \| R_t')$, $V_2 = h(SK_{it} \| R_t' \| dA)$, and sends $\{B, V_2\}$ to SC_t/P_t over a public channel.
3) SC_t/P_t computes $B' = aB = adP$, $SK_{it}' = h(HID_i \| ID_t \| B' \| R_t)$, $V_2' = h(SK_{it}' \| R_t' \| aB)$, and verifies whether $V_2' \stackrel{?}{=} V_2$ holds. If the verification fails, SC_t/P_t aborts the session; otherwise, computes the session key as $SK_{it}' = h(HID_i \| ID_t \| B' \| aP_{Hi})$.

The same process is followed by D_i whenever he/she has to upload EHR_t of a patient at HS_i.

2) Login and Authentication Between P_t, HS_i and HS_j for Secure Data-Sharing: This authentication phase is designed to establish a secure channel for medical data sharing between two hospital servers HS_i and HS_j, which is shown in Fig. 5. Since, only P_t visits more than one hospital to avail better treatment facility. Here, the users are only of the patient category registered at H_i and visit H_j. This phase shows how HS_j authenticates P_t and accesses EHR_t of P_t from HS_i. The login and authentication process of P_t with HS_j and HS_i for the medical data-sharing process is described as follows:

1) P_t inputs ID_t, PW_t to SC_t, and imprints Bio_t to the biometric scanner-device [19]. SC_t computes

P_t/SC_t	HS_j	HS_i
Input ID_t, PW_t to SC_t		
Imprint Bio_t to scanner-device		
Rep$(Bio_t, \tau_t) = \sigma'_t$		
$k'_t = B_t \oplus h(PW_t \parallel \sigma'_t)$		
$PWD'_t = h(PW_t \parallel k'_t)$		
$PID'_t = A_t \oplus PWD'_t$		
$V'_t = h(PID'_t \parallel PWD'_t)$		
If $(V'_t \neq V_t)$		
Abort the session		
Else, choose $a \in \mathbb{Z}_p$		
$A = aP$, $R_t = aP_{Hi} = ak_iP$		
$M_1 = ID_t \oplus h(R_t)$		
$V_1 = h(PID_t \parallel R_t \parallel HID_i \parallel ID_t)$		
$\xrightarrow[\textbf{Public channel}]{\{M_1, V_1, A\}}$		
	Choose $b \in \mathbb{Z}_p$, $B_j = bP$	
	$B_{ja} = bA$, $R_j = bP_{Hi}$	
	$Z_j = CT_j + b$, $M_2 = Z_j \oplus h(R_j)$	
	$M_3 = h(Z_j \parallel R_j \parallel HID_j \parallel B_{ja})$	
	$\xrightarrow[\textbf{Public channel}]{\{B_j, B_{ja}, M_2, M_3, M_1, V, A\}}$	
		$R'_j = k_iB_j = bk_iP$, $Z'_j = M_2 \oplus h(R'_j)$
		$C' = Z'_jP = HID_jP_{ta} + P_{Hj} + B_j$
		If$(C' \neq HID_iP_{ta} + PHj + B_j)$
		Abort the session
		Else, $M'_3 = h(Z'_j \parallel R'_j \parallel HID_j \parallel B_{ja})$
		If$(M'_3 \neq M_3)$
		Abort the session
		Else, $R'_t = k_iA = ak_iP$
		$ID_t = M_1 \oplus h(R_t)$, $PID'_t = h(ID_t \parallel K_i)$
		$V'_1 = h(PID_t \parallel R_t \parallel HID_i \parallel ID_t)$
		If$(V'_1 \neq V_1)$
		Abort the session
		Else, choose $n \in \mathbb{Z}_p$
		$N = nP$, $N_i = nA = naP$
		$N_{ijt} = nB_{ja} = banP$
		$R_i = nP_{Hj}$, $Z_i = CT_i + n$, $M_4 = Z_i \oplus h(R_i)$
		$P_{ti} = h(HID_i \parallel PID_t \parallel R'_t)$, $M_5 = P_{ti} \oplus h(R_{ix})$
		$SK_{ijt} = h(N_{ijt} \parallel HID_j \parallel HID_i \parallel P_{ti})$
		$M_6 = h(N_i \parallel HID_i \parallel P_{ti} \parallel N_{ijt} \parallel SK_{ijt})$
		$\xleftarrow[\textbf{Public channel}]{\{N, N_i, M_4, M_5, M_6\}}$
	$R'_i = k_jN$, $Z'_i = M_4 \oplus h(R'_i)$	
	$C_1 = Z'_iP = (CT_i + n)P$, $C'_1 = HID_iP_{ta} + P_{Hi} + N$	
	If$(C_1 \neq C'_1)$	
	Abort the session	
	Else, $P^*_{ti} = M_5 \oplus h(R_{ix})$, $B_{ijt} = bN_i = abnP$	
	$SK_{ijt} = h(B_{ijt} \parallel HID_j \parallel HID_i \parallel P^*_{ti})$	
	$M'_6 = h(N_i \parallel HID_i \parallel P^*_{ti} \parallel B_{ijt} \parallel SK_{ijt})$	
	If$(M_6 \neq M'_6)$	
	Abort the session	
	Else, $B_{ji} = bN = bnP$	
	$M'_7 = h(HID_i \parallel HID_j \parallel P^*_{ti} \parallel B_{ijt} \parallel SK_{ijt} \parallel B_{ji})$	
$A_{ijt} = aB_{ji} = bnaP$, $P'_{ti} = h(HID_i \parallel PID_t \parallel R_t)$	$\xleftarrow[\textbf{Public channel}]{\{B_{ji}, M_7\}}$	
$SK'_{ijt} = h(A_{ijt} \parallel HID_j \parallel HID_i \parallel P'_{ti})$		
$M'_7 = h(HID_i \parallel HID_j \parallel P'_{ti} \parallel A_{ijt} \parallel SK_{ijt} \parallel B_{ji})$		
If$(M'_7 \neq M_7)$		
Abort the session		
Else, $M_8 = h(SK_{ijt} \parallel A_{ijt})$		
$\xrightarrow[\textbf{Public channel}]{\{M_8\}}$		
	$M'_8 = h(SK_{ijt} \parallel B_{ijt})$	
	If$(M'_8 = M_8)$	
	Abort the session	
	Else, accept SK_{ijt} as session key	

Fig. 5. Login and authentication between P_t, HS_i and HS_j.

$\mathbf{Rep}(Bio_t, \tau_t) = \sigma'_t$, $k'_t = B_t \oplus h(PW_t \parallel \sigma'_t)$, $PWD'_t = h(PW_t \parallel k'_t)$, $PID'_t = A_t \oplus PWD'_t$, $V'_t = h(PID'_t \parallel PWD'_t)$, and verifies whether $V'_t \stackrel{?}{=} V_t$ holds. If the verification fails, SC_t aborts the session; otherwise, chooses $a \in \mathbb{Z}_p$, computes $A = aP$, $R_t = aP_{Hi} = ak_iP$, $M_1 = ID_t \oplus h(R_t)$, $V_1 = h(PID_t \parallel R_t \parallel HID_i \parallel ID_t)$, and sends the login message $\{M_1, V_1, A\}$ to HS_j over a public channel.

2) HS_j chooses $b \in \mathbb{Z}_p$, computes $B_j = bP$, $B_{ja} = bA$, $R_j = bP_{Hi}$, $Z_j = CT_j + b$, $M_2 = Z_j \oplus h(R_j)$, $M_3 = h(Z_j \parallel R_j \parallel HID_j \parallel B_{ja})$, and sends $\{B_j, B_{ja}, M_2, M_3, M_1, V, A\}$ to HS_i over a public channel.
3) HS_i computes $R'_j = k_iB_j = bk_iP$, $Z'_j = M_2 \oplus h(R'_j)$, $C' = Z'_jP = HID_jP_{ta} + P_{Hj} + B_j$, and verifies whether $C' \stackrel{?}{=} HID_iP_{ta} + P_{Hj} + B_j$ holds. If the verification fails, HS_i aborts the session; otherwise,

computes $M_3' = h(Z_j' \parallel R_j' \parallel HID_j \parallel B_{ja})$, and further verifies whether $M_3' \stackrel{?}{=} M_3$ holds. If the verification fails, HS_i aborts the session; otherwise, computes $R_t' = k_i A = ak_i P$, $ID_t = M_1 \oplus h(R_t)$, $PID_t' = h(ID_t \parallel K_i)$, $V_1' = h(PID_t \parallel R_t \parallel HID_i \parallel ID_t)$, and also verifies whether $V_1' \stackrel{?}{=} V_1$ holds. If the verification fails, HS_i terminate the session; otherwise, chooses $n \in \mathbb{Z}_p$, computes $N = nP$, $N_i = nA = naP$, $N_{ijt} = nB_{ja} = banP$, $R_i = nP_{Hj}$, $Z_i = CT_i + n$, $M_4 = Z_i \oplus h(R_i)$, $P_{ti} = h(HID_i \parallel PID_t \parallel R_t')$, $M_5 = P_{ti} \oplus h(R_{ix})$, $SK_{ijt} = h(N_{ijt} \parallel HID_j \parallel HID_i \parallel P_{ti})$, $M_6 = h(N_i \parallel HID_i \parallel P_{ti} \parallel N_{ijt} \parallel SK_{ijt})$, and sends the reply message $\{N, N_i, M_4, M_5, M_6\}$ to HS_j over a public channel.

4) HS_j computes $R_i' = k_j N$, $Z_i' = M_4 \oplus h(R_i')$, $C_1 = Z_i' P = (CT_i + n)P$, $C_1' = HID_i P_{ta} + P_{Hi} + N$, and verifies whether $C_1 \stackrel{?}{=} C_1'$ holds. If it fails, HS_j aborts the session; otherwise, computes $P_{ti}^* = M_5 \oplus h(R_{ix})$, $B_{ijt} = bN_i = abnP$, and the session key $SK_{ijt} = h(B_{ijt} \parallel HID_j \parallel HID_i \parallel P_{ti}^*)$, $M_6' = h(N_i \parallel HID_i \parallel P_{ti}^* \parallel B_{ijt} \parallel SK_{ijt})$. HS_j verifies whether $M_6 \stackrel{?}{=} M_6'$ holds. If the verification fails, HS_j aborts the session; otherwise, computes $B_{ji} = bN = bnP$, $M_7' = h(HID_i \parallel HID_j \parallel P_{ti}^* \parallel B_{ijt} \parallel SK_{ijt} \parallel B_{ji})$, and sends the message $\{B_{ji}, M_7\}$ to SC_t/P_t over a public channel.

5) SC_t/P_t computes $A_{ijt} = aB_{ji} = bnaP$, $P_{ti}' = h(HID_i \parallel PID_t \parallel R_t)$, $SK_{ijt}' = h(A_{ijt} \parallel HID_j \parallel HID_i \parallel P_{ti}')$, $M_7' = h(HID_i \parallel HID_j \parallel P_{ti}' \parallel A_{ijt} \parallel SK_{ijt} \parallel B_{ji})$, and verifies whether $M_7' \stackrel{?}{=} M_7$ holds. If the verification fails, SC_t/P_t terminates the session; otherwise, computes $M_8 = h(SK_{ijt} \parallel A_{ijt})$, and sends the response message M_8 to HS_j over a public channel.

6) HS_j computes $M_8' = h(SK_{ijt} \parallel B_{ijt})$, and verifies whether $M_8' = M_8$ holds. If the verification fails, H_j aborts this session; otherwise, accepts SK_{ijt} as the correct session key.

IV. Block Generation and Addition

This section presents the process of storing the metadata of a patient's EHR in a consortium blockchain. A doctor D_i (who is serving H_i) will give the treatment to the patient P_t. After the treatment, D_i will generate a medical report EHR_t of P_t at time T_g. To upload EHR_t at HS_i, D_i will perform authentication with HS_i, and share a session key SK_{idi}. This process will be same as the authentication and session key agreement between HS_i and P_t discussed in Section III-C1. After that, D_i encrypts EHR_t using SK_{idi}, and sends it to HS_i. HS_i decrypts the message, and gets EHR_t. Next, HS_i encrypts EHR_t with the public key P_{Hi} to obtain $E_{P_{Hi}}(EHR_t \parallel PID_t \parallel PID_i \parallel T_g)$, and then stores it in database, where PID_t and PID_i are is pseudo-identity of P_t and D_i. After that, HS_i generates a transaction for blockchain storage. The work of blockchain starts when it receives the transaction from the various hospital servers which is described below.

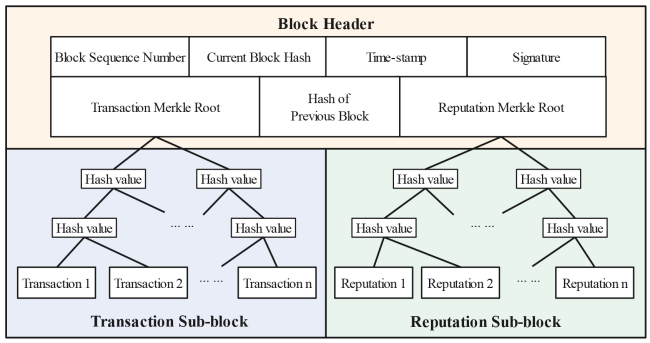

Fig. 6. Block structure for reputation-based transactions.

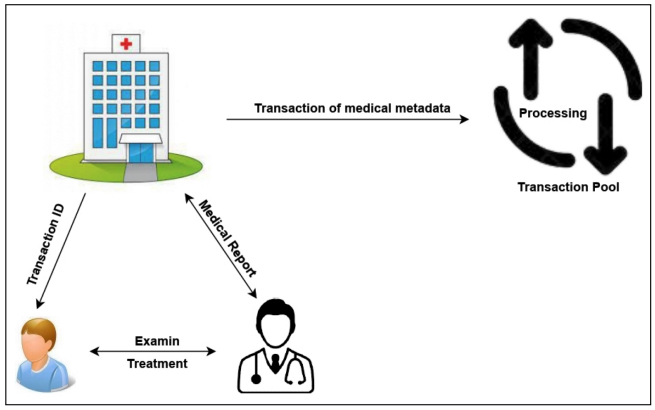

Fig. 7. Transaction generation for medical data.

Transaction ID	Tx_{id}
Hospital ID	HID_i
Transaction type	EHR medical report
Details of EHR report	MD_{tg}, $HEHR_t$, χ_t, χ_i
Timestamp	T_g
Hash of transaction	$h(Tx_{itg})$
Signature of the proposer	$\mathbf{Sign}_{H_i}$

Fig. 8. Transaction format for medical data.

A. Transaction Generation

Here two types of transactions are generated and managed in the blockchain network. One type of transaction is created when HS_i prepares the metadata EHR_t of P_t. Other type of transaction is created when HS_j requests any service from HS_i, and based on the service received from HS_i, HS_j will generate a rating report Tr_{ij}. We have followed the block structure proposed in [20] to manage these two types of transactions. The block structure is shown in Fig. 6.

1) Transaction for Medical Data: After adding the encrypted medical report EHR_t of P_t to the database, HS_i further processes it for adding the metadata in the blockchain network. Fig. 7 shows the transaction generation process for EHR_t metadata. The EHR_t metadata transaction format of the proposed scheme is shown in Fig. 8.

Transaction ID	Tr_{id}
Transaction type	Trustvalue
Provider's details	HID_i
Service requested	Tx_{id}
Timestamp	T_s
Hash value of the service	$h(MD_{tg})$
Signature of the provider	$\mathbf{Sign}_{HS_i}$
Details of rating provider	HID_j
Reputation score	TVi_{s+1}
Time of rating	T_{s+1}
Hash of the service received	$h(MD_{tg})$
Hash of the transaction	$h(Tr_{ij})$
Signature of the rating provider	$\mathbf{Sign}_{HS_j}$

Fig. 9. Transaction format for rating information.

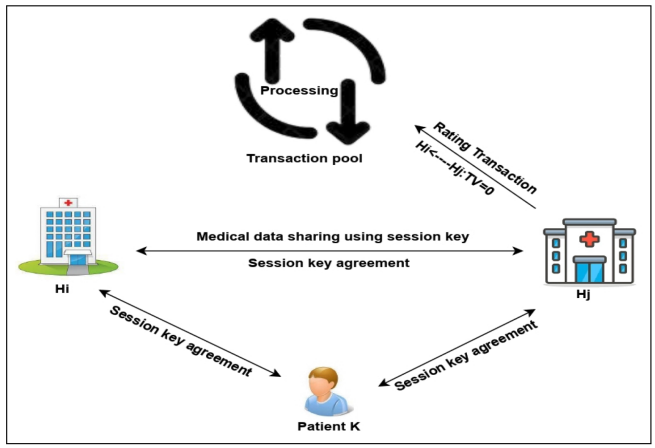

Fig. 10. Transaction generation for reputation information.

Block Header	
Block sequence number	BSN
Timestamp	TS
Merkle Tree root for medical transaction	EHR_{MT}
Hash of the previous block	PBhash
Merkle Tree root for reputation transaction	EHR_{RT}
Hash value of the current block	CBH
Signature of the Leader node on CBH	$CBH.\mathbf{Sign}_{HL}$
Block Payload (encrypted transaction)	
List of medical transaction	Tx_i^{id}, $\mathbf{Sign}_{H_i}(Tx_i)$, $h(Tx_i), 1 \le i \le n$
List of reputation transaction	Tr_i^{id}, $\mathbf{Sign}_{H_j}(Tr_i)$, (Tr_i), $1 \le i \le n$, $1 \le j \le n$

Fig. 11. Structure of a block.

1) HS_i computes $\chi_t = h(PID_t \parallel T_g)$, and $\chi_i = h(PID_i \parallel T_g)$ as pseudo identity of P_t and D_i. Next, HS_i computes $HEHR_t = h(EHR_t \parallel T_g)$, and generates a metadata $MD_{tg} = h(HEHR_t \parallel \chi_t \parallel \chi_i)$ of $HEHR_t$.
2) HS_i generates a transaction $Tx_{itg} = MD_{tg} \parallel \chi_t \parallel \chi_i \parallel HEHR_t$, digitally signs it using ECDSA (elliptic curve digital signature algorithm) by its own private key k_i. HS_i sends the signed Tx_{itg} in the memory pool, and delivers Tx_{id} to P_t for future reference.
3) From the memory pool, other nodes (hospital servers) receive these transactions and verify their validity by calculating $MD_{tg} = h(HEHR_t \parallel \chi_t \parallel \chi_i)$ from Tx_{itg}. If the number of valid transactions reaches a threshold value λ, then this set of transactions will be selected to prepare a block. At the same time, other transactions will remain in the memory pool.

2) Transaction for Rating Information: Another transaction is generated when HS_j requests data of P_t (registered at HS_i) from HS_i. After obtaining the data, HS_j generates a transaction that rates the service quality provided by HS_i. This rating represents the reputation score from $\{-1, 0, +1\}$, where -1 for no service received, 0 satisfactory, $+1$ for good service. This transaction generates a ranking list which shows the trustworthiness of the participant hospital nodes. A hospital node with the highest reputation score is allowed to mine the block for next round. The transaction format is shown in Fig. 9. To avoid any kind of control or manipulation over rating process, there is a restricted selection of rating transaction, it means, if HS_j gives multiple ratings to HS_i for same round of consensus and wants to make highest reputation score for HS_i, then the algorithm will select the latest rating $Tr_{ij} : HS_j \rightarrow HS_i$ for same pair (HS_j, HS_i) and discard all previous rating. In this way for each pair (HS_j, HS_i), only one transaction will be considered in one round of block commitment. Fig. 10 shows the generation of rating transactions.

3) Block Construction and Addition: In this phase, all the transactions (medical data and ratings) generated by hospital servers (blockchain nodes) are transmitted to the transaction pool. Based on the work proposed in [21], the PoR algorithm will decide which node will mine the block. The block format is shown in Fig. 11. All the participant nodes (hospital servers) can extract each service provider's rating log $l_{H_i} = \{sr_1, sr_2, sr_3, \ldots, sr_x\}$, where sr_k is the kth score of l_{H_i} and x is the total number of scores it receives from the raters of the blockchain network. As motioned in the work proposed in scheme [21], the trustworthiness of HS_i can be calculated based on the values received from the rating log as:

$$TV_i = \frac{1}{1 + e^{-\phi \sum_{k=1}^{x} sr_k}}$$

After assigning TV_i to each HS_i, a ranking list is generated, and based on that, HS_i having the highest TV_i score will mine the block B_{t+1} in next round of consensus. So, if in this round of consensus, the leader, say HS_l, has the highest trust-value score from the latest updated successful round (for the addition of B_{t-1}). Then H_l will publish the newly created block B_t, and all other hospital server nodes receive the block B_t and verify its validity. After all, the participant node gives a valid vote reply for the newly created block B_t, then H_l will add this block to the blockchain network. Fig. 12 shows the transaction verification process, and Fig. 13 shows the block validation and addition process. The algorithm 1 shows the consensus procedure for block verification and addition. We can state the PoR consensus algorithm with the following steps.

1) Assume that a transaction pool includes λ number of valid transactions. The participating node with the highest reputation value from the latest committed block $B_{t-1}$is elected as leader H_{lead}.
2) H_{lead} construct a block B_t and set the *count* $= 0$, which means the valid vote reply for B_t is initialized to zero.

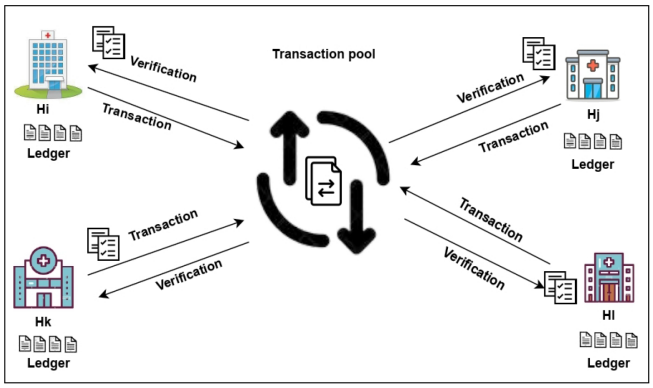

Fig. 12. Transaction verification process.

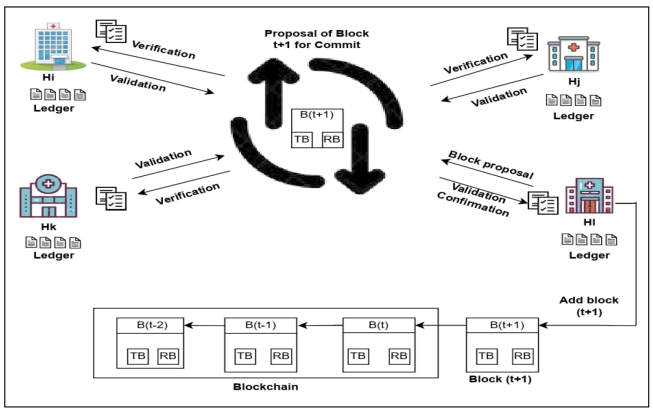

Fig. 13. Block validation and addition process.

Algorithm 1: Consensus algorithm.

Input : T_{pool}: Transactions pool, N: Number of nodes in the blockchain network, λ: Transaction threshold, A_{pr}: Approval threshold to commit a block $= \frac{2}{3}\times N$, RL: Ranking list, count, flag

Output: Block commitment

if $(T_{pool} = \lambda)$ **then**
  Check RL from previous committed block B_{t-1}
  Select a leader H_{lead} with highest reputation value
  H_{lead} constructs a block B_t and set ***count*** $= 0$,
  H_{lead} broadcasts B_t in the network
  Other participating nodes receives B_t
  if $(CBH.\boldsymbol{Sign}_{HL} \neq \boldsymbol{Sign}_{H_{lead}})$ **then**
    Return B_t as invalid block
  else
    verify$(T_{MRT}, R_{MRT}, h(Tx_i), h(Tr_i), PBhash, CBH)$
  if (*all other nodes found* B_t *is a valid*) **then**
    Return B_t as valid block
  for (*each vote received from other nodes*) **do**
    $flag = 1$
    count $=$ ***count*** $+ 1$
  if (***count*** $\geq A_{pr}$) **then**
    Add B_t into blockchain

H_{lead} broadcast B_t in the network. Other participating nodes receive the newly created block and will validate it by verifying whether the block's signature is the same as H_{lead}. All nodes vote as an **invalid block** if the verification fails. Otherwise, all the nodes further verify the block contents, such as the Merkle tree root of the transaction sub-block (T_{MRT}), reputation sub-block (R_{MRT}), previous block hash (PBhash), current block hash (CBH), and the hash of transactions ($h(Tx_i)$, $h(Tr_i)$). If all these values are verified successfully, the participating node gives a **valid vote** reply to H_{lead}.

3) For a **valid vote** received from a node, H_{lead} set $flag = 1$, which ensures that a node cannot vote again for the same consensus round. For each node after performing $flag = 1$, H_{lead} increment the vote count by 1. If the total valid vote count is greater than the approved threshold $\frac{2}{3}\times N$ selected for adding a block into the blockchain, H_{lead} add the block into the blockchain.

4) Data-Sharing and Integrity Verification: Assume that P_t visits another hospital H_j for better treatment. To access his/her data stored at HS_i, P_t provides the data Tx_{id} to HS_j as a reference of his/her EHR_t metadata stored in the blockchain network. Next, HS_j searches the blockchain ledger to see whether Tx_{id} is available or not. If it is available there, HS_j retrieves $Tx_{itg} = MD_{tg} \parallel \chi_t \parallel \chi_i \parallel HEHR_t$, and sends a request to HS_i for EHR_t of P_t. The process of authentication is performed, and all three parties share a common session key SK_{ijt}. HS_i decrypts $E_{P_{Hi}}(EHR_t \parallel PID_t \parallel PID_i \parallel T_g)$ using his/her private key k_i. Next, HS_i encrypts $\{EHR_t, T_g\}$ using SK_{ijt}, and sends it to HS_j. Thereafter, HS_j receives the requested data and retrieve EHR_t, T_g, then it computes $HEHR'_t = h(EHR_t \parallel T_g)$, and $MD'_{tg} = h(HEHR'_t \parallel \chi_t \parallel \chi_i)$, and verifies whether $MD'_{tg} \stackrel{?}{=} MD_{tg}$ holds. If the verification is successful, HS_j considers that EHR_t is correct, and it then assigns a rating value to HS_i.

V. Security Analysis

In this section, we have performed the provable security analysis of the BC-UADS framework. Due to space limitations, we have provided the BAN logic and AVISPA analysis in Appendix. We have considered the following entities in the BC-UADS framework: (i) a patient P_t with identity ID_t, password PW_t, biometric Bio_t, and a secret value k_t (ii) a hospital server HS_i with identity HID_i and a secret value k_i. (ii) a hospital server HS_j with a secret value k_j. All the participating entities can have multiple instances, i.e., P_t^k, HS_i^k, and HS_j^k, denote the kth instance of P_t, HS_i, and HS_j, respectively. We have considered all of them to be oracles and have three states: (i) **Accept**, (ii) **Reject**, and (iii) $\perp$ (no output). We have denoted the BC-UADS framework as Π.

Definition 1: An adversary $\mathcal{A}$ can ask the following queries:

- *Execute*(P_t^k, HS_i^k, HS_j^k): This query helps to perform the passive attacks against Π. The outcome of this oracle query are the transcripts $\{M_1, V_1, A\}$, $\{B_j, B_{ja}, M_2, M_3, M_1, V, A\}$, $\{N, N_i, M_4, M_5, M_6\}$, $\{B_{ji}, M_7\}$ $\{M_8\}$.

- *Send*$(P_t^k/HS_i^k/HS_j^k, msg)$: This query helps to perform the active attacks. $\mathcal{A}$ can send a message msg to $P_t^k/HS_i^k/HS_j^k$. If msg is valid, the outcome is computed as per the description of Π; otherwise, this query is ignored.
- *Reveal*(P_t^k): This query outputs the current session key held by the instance P_t^k.
- *Corrupt*(P_t^k, c): It outputs the information in three cases.
  1) *Corrupt*$(P_t^k, 0)$: It returns the password PW_t of P_t.
  2) *Corrupt*$(P_t^k, 1)$: It returns the data stored in the smartcard SC_t of P_t.
  3) *Corrupt*$(P_t^k, 2)$: It returns the biometric Bio_t of P_t.
- *Test*(P_t^k): This query test the semantic security of the session key SK_{ijt}. The outcome of this query is the session key computed between P_t^k and its partner. If P_t^k or its partner is not fresh, this oracle returns $\perp$. This oracle query starts by flipping an unbiased coin b, which is kept secret. If $\mathcal{A}$ executes a $\mathbf{Test}(P_t^k)$ query, this oracle responses with a session key if $b = 1$. If $b = 0$, then oracle returns value chosen from $\{0,1\}^l$ uniformly at random, where l denotes the length of SK_{ijt}.

Definition 2 (Partnership): If P_t^k and HS_i^k both entered into an **accept state** and generates a session key, then P_t^k and HS_i^k are considered as **partner**. The partner identity of P_t^k and HS_i^k are computed as $pid_{HS_i}^k$, and $pid_{P_t}^k$, respectively.

Definition 3 (Freshness): We call P_t^k and HS_i^k are **fresh** if the following conditions does not occur:

1) $\mathcal{A}$ queries $\mathbf{Reveal}(P_t^k/HS_i^k)$ to P_t^k or its partner HS_i^k.
2) $\mathcal{A}$ queries $\mathbf{Corrupt}(P_t^k, c)$.
3) $\mathcal{A}$ queries $\mathbf{Corrupt}(P_t^k/HS_i^k)$ before executing $\mathbf{Test}(P_t^k/HS_i^k)$ query.

Definition 4 (Semantic Security): Let the success probability of $\mathcal{A}$ in breaking Π is denoted as Pr[S]. $\mathcal{A}$ executes a Test query, which guess a bit b'. $\mathcal{A}$ is said to break the semantic security of the session key of Π if $\mathcal{A}$ successfully differentiates between a correct session key and a random key. This event occurs only when the condition $b' == b$ holds. Therefore, the initial probability of $\mathcal{A}$ breaching the semantic security of Π is defined as $\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t) = |\text{Pr[S]} - \frac{1}{2}|$. The scheme Π is secure if $\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t) \leq \varepsilon$, where ε is a negligible function.

Theorem 1 (Provable Security): Suppose $E(\mathbb{Z}_p)$ is an elliptic curve group, $\mathcal{D}$ is a uniformly distributed password dictionary of size $|\mathcal{D}| \leq 10^6$ [22], y is the length of biometric key σ_t which is obtained from $\mathbf{Rep}(Bio_t, \tau_t)$, and ℓ is the bit length of the digest of the hash function $h(\cdot)$. Assume that $\mathcal{A}$ executes at most q_{snd} times **Send** query, q_{exe} times **Execute** query, and q_{hash} times ***Hash*** query within polynomial time t to breach the semantic security of Π. Let $\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t)$ is the breaching probability and $\text{Exp}_{\mathcal{A}}^{\text{ECCDH}}(t')$ is the advantage of $\mathcal{A}$ to solve the ECCDH problem in polynomial time t'. Then, we have,

$$\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t) \leq \frac{q_{hash}^2}{2^{\ell+1}} + \frac{(q_{snd}+q_{exe})^2}{2(p-1)} + q_{snd}\text{Max}\left\{\frac{1}{|\mathcal{D}|}, \frac{1}{2^y}, f_p\right\} + \frac{q_{snd}}{2^\ell} + q_{hash}(q_{snd}+q_{exe})^2\text{Exp}_{\mathcal{A}}^{\text{ECCDH}}(t')$$

where, $t' \approx t + q_{hash}(q_{snd}+q_{exe})T_{ecc}$, and T_{ecc} denotes the elliptic curve scalar point multiplication

Proof: In this proof, a series of games G_i, $0 \leq i \leq 5$ have been executed by $\mathcal{A}$ with a polynomial time bounded challenger $\mathcal{C}$. Based on the outcomes of these games, $\mathcal{A}$ is trying to break the semantic security of Π. During the simulation of a Test query, $\mathcal{C}$ chooses a bit $b \in \{0,1\}$ and starts the game. After analyzing the outcomes of the games G_i $\mathcal{A}$ guesses a bit $b' \in \{0,1\}$. Suppose, S_i is an event where $b' == b$ holds. The probability that S_i occurs is denoted by $\text{Pr}[\text{S}_i]$. If G_i and G_{i+1} are the two distinct games, the difference between them is denoted as $\triangle_i$ [23]. Therefore, we have

$$\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t) = |\text{Pr[S}_n] - \frac{1}{2}| + (|\text{Pr[S}_0] - \text{Pr[S}_n]|)$$
$$\leq |\text{Pr[S}_n] - \frac{1}{2}| + \sum_{i=0}^{n-1} \triangle_i \quad (1)$$

- G_0: In this game, $\mathcal{C}$ executes all the steps of Π and returns all the outputs to $\mathcal{A}$. Thus, this game is indistinguishable from the real attack. Therefore, we have:
$$\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t) = |\text{Pr[S}_0] - \frac{1}{2}| \quad (2)$$
- G_1: Based on the oracle queries received from $\mathcal{A}$, $\mathcal{C}$ simulates $h(\cdot)$ by maintaining an initial-empty list L_h that stores the tuples of the form (x, y), where x is input, and y is output. When $\mathcal{A}$ asks to simulate the oracle $h(\cdot)$ for an input x_i, $\mathcal{C}$ outputs y_i if a tuple $(x_i, y_i) \in L_h$. If $(x_i, y_i) \notin L_h$, $\mathcal{C}$ chooses a random value y_i of length l, and returns it as output. $\mathcal{C}$ inserts the new tuple (x_i, y_i) in L_h. In this game, the output of $h(\cdot)$ is random, therefore, G_1 is indistinguishable from G_0. So, we get:
$$\triangle_0 = |\text{Pr[S}_1] - \text{Pr[S}_0]| = 0 \quad (3)$$
- G_2: In this game, $\mathcal{A}$ simulate all the oracles **Execute**, **Send**, **Hash**, **Reveal**, **Corrupt**, and **Test**. $\mathcal{A}$ simulates $h(\cdot)$ and tries to get the collisions in the transcripts $\{M_1, V_1, A\}$, $\{B_j, B_{ja}, M_2, M_3, M_1, V, A\}$, $\{N, N_i, M_4, M_5, M_6\}$, $\{B_{ji}, M_7\}$, and $\{M_8\}$. According to the birthday attack, the probability of finding at least one collision is $\frac{q_{hash}^2}{2^{\ell+1}}$. The transcripts $\{M_1, V_1, A\}$, $\{B_j, B_{ja}, M_2, M_3, M_1, V, A\}$, $\{N, N_i, M_4, M_5, M_6\}$, $\{B_{ji}, M_7\}$, and $\{M_8\}$ are generated by either executing **Execute** and ***Send*** queries. Therefore, the advantage of $\mathcal{A}$ to find a collision is at most $\frac{(q_{snd}+q_{exe})^2}{2(p-1)}$. In this case, G_2 is indistinguishable from previous game because $\{a, b, n\} \in \mathbb{Z}_p$ are randomly selected. Therefore, we get:
$$\triangle_1 = |\text{Pr[S}_2] - \text{Pr[S}_1]| \leq \frac{q_{hash}^2}{2^{\ell+1}} + \frac{(q_{snd}+q_{exe})^2}{2(p-1)} \quad (4)$$
- G_3: This game is based on the stolen smartcard attack. $\mathcal{A}$ is allowed to perform a Corrupt$(P_t^k, 1)$ query, and then gets information stored in SC_t of P_t based on which $\mathcal{A}$ guesses the password and biometric of P_t.

1) If Corrupt$(P_t^k, 2)$ is queried, $\mathcal{A}$ attempts to guess the password PW_t of P_t from $|\mathcal{D}|$ in offline mode. The maximum attempt to guess the password, $\mathcal{A}$ can have is q_{snd} number of **Send** queries. Therefore, we get $\frac{q_{snd}}{|\mathcal{D}|}$.
2) If Corrupt$(P_t^k, 0)$ is queried, $\mathcal{A}$ tries to guess the biometric-based secret key σ_t of P_t.
   a) In first case, $\mathcal{A}$ can selects a binary string y to forge σ_t. The probability to perform this task is $\frac{q_{snd}}{2^y}$.
   b) In second case, $\mathcal{A}$ uses the biometric that is different from the biometric of P_t and take advantage of false positive f_p. Since, $\mathcal{A}$ has asked Corrupt$(P_t^k, 1)$ previously and he can ask at most two **Corrupt** queries. Therefore, $\mathcal{A}$ cannot guess PW_t and σ_t simultaneously. Hence, the probability of success of $\mathcal{A}$ is:

$$\triangle_2 = |\Pr[S_3] - \Pr[S_2]| \leq \text{Max}\left\{q_{snd}\left(\frac{1}{|\mathcal{D}|}, \frac{1}{2^y}, f_p\right)\right\} \quad (5)$$

- G_4: $\mathcal{A}$ executes this game to breach the mutual authentication property of Π. To do so, $\mathcal{A}$ asks $\mathcal{C}$ to simulate Send$(Start, P_t)$, Send$(\{M_1, V_1, A\}, HS_j)$, and Send$(\{B_j, B_{ja}, M_1, M_2, M_3\, V, A\}, HS_i)$, Send$(\{N, N_i, M_4, M_5, M_6\}, HS_j)$, Send$(\{B_{ji}, M_7\}, P_t)$ Send$(\{M_8\}, HS_j)$ queries. This game is indistinguishable from the previous game unless HS_i, HS_j, or P_t aborts on a valid response message. So, we have:

$$|\Pr[S_4] - \Pr[S_3]| \leq \frac{q_{snd}}{2^\ell} \quad (6)$$

- G_5: In this game, $\mathcal{A}$ tries to break the semantic security of the session key. If $\mathcal{A}$ compute the session key $SK_{ijt} = h(* \parallel HID_j \parallel HID_i \parallel *)$, where $*$ denotes the parameter unknown to $\mathcal{A}$. $\mathcal{A}$ solve the ECCDH problem from the instances (N_{jit}, N, B_{ja}) and (B_{ja}, A, B_j), where $N_{ijt} = ECCDH(N, B_{ja})$ and $B_{ja} = ECCDH(A, B_j)$. Since $\{a, b, n\}$ are selected randomly from $\mathbb{Z}_p$, and $\mathcal{A}$ either perform **Execute** query or **Send** query. So, the advantage of $\mathcal{A}$ is bounded by $\frac{1}{(q_{snd}+q_{exe})^2}$. Hence, we get:

$$\triangle_3 = |\Pr[S_5] - \Pr[S_4]| \leq q_{hash}(q_{snd} + q_{exe})^2 \text{Exp}_{\mathcal{A}}^{\text{ECCDH}}(t') \quad (7)$$

If $\mathcal{A}$ uses the private oracle $h(\cdot)$, and simulates a Test(P_t^k) query, $\mathcal{C}$ selects $b \in \{0, 1\}$, searches for the value available in L_h and forward it to $\mathcal{A}$. Otherwise, $\mathcal{C}$ selects a value $SK_{ijt} \in \{0, 1\}^l$ uniformly at random as session key and returns it to $\mathcal{A}$. $\mathcal{A}$ guess $b' \in \{0, 1\}$ and wins the game if $b' = b$ holds. The probability of this condition is $\frac{1}{2}$. Therefore, we get:

$$\Pr[S_5] = \frac{1}{2} \quad (8)$$

From Equation (1) – (8), we obtain that the success probability and time of $\mathcal{A}$ to breach the semantic security of Π as

$$\text{Exp}_{\Pi,\mathcal{A}}^{\text{AKE}}(t) \leq \frac{q_{hash}^2}{2^{\ell+1}} + \frac{(q_{snd} + q_{exe})^2}{2(p-1)} + q_{snd}\text{Max}\left\{\frac{1}{|\mathcal{D}|}, \frac{1}{2^y}, f_p\right\} + \frac{q_{snd}}{2^\ell} + q_{hash}(q_{snd} + q_{exe})^2 \text{Exp}_{\mathcal{A}}^{\text{ECCDH}}(t')$$

Where,

$$t' \approx t + q_{hash}(q_{snd} + q_{exe})T_{ecc}$$

TABLE III
DESCRIPTION OF VARIOUS SECURITY LEVEL

Security Level (in bits)	$P \in G$ (in bits)	h (in bits)	p (in bits)	q (in bits)
80	1024	160	512	160
112	2048	224	1024	224
128	3072	256	1536	256
192	7680	384	3840	384
256	15360	512	7680	512

VI. PERFORMANCE ANALYSIS

This section shows the comparative analysis of the communication, execution, and storage costs of the BC-UADS framework and Wang et al.'s scheme [17] for the security levels: 80-bit, 112-bit, 128-bit, 192-bit, and 256-bit AES key size. The cryptographic operations are executed based on a singular Type-A elliptic curve: $y^2 \equiv (x^3 + x) mod\ p$ over a prime field $\mathbb{Z}_p$. The order of G_p is $p = q \times h - 1$, where h is a cofactor, $q = 2^{exp2} + sign1 \times 2^{exp1} + sign0$ and $p \equiv 3 \bmod 4$. Table III shows the five different levels of security [24], size of the elements of G_p and G_q, and size of q and p, and the size of the digest for $h(\cdot)$. In the communication cost calculation, we assume that $|ID_i| = |PW_i| = |\text{timestamp}| = 128$ bits. We have compared the communication cost of [17] with BC-UADS framework. We have computed the communication cost the BC-UADS framework for 80-bit security level. P_t sends $\{M_1, V_1, A, M_8\}$ to HS_j. The communication cost for this message is $|h| + |h| + |P| + |h| = 1504$ bits. HS_j sends $\{B_j, B_{ja}, M_2, M_3, M_1, V, A\}$ to HS_i and $\{B_{ji}, M_7\}$ to P_t. Therefore, the communication costs are $|P| + |P| + |p| + |h| + |h| + |h| + |P| = 4064$ bits, and $|P| + |h| = 1184$ bits. So, the total communication cost at HS_j is $4064 + 1184 = 5248$ bits. HS_i sends the message $\{N, N_i, M_4, M_5, M_6\}$ to HS_j, and therefore, the communication cost is $|P| + |P| + |p| + |h| + |h| = 2880$ bits. Similarly, we have computed the communication cost of the scheme in [17] for security level of 80-bits. In this scheme, the user U sends the messages $\{A, W_u, \sigma, T_u\}$ and $\{M\}$ to the edge server ES_1. Therefore, the communication costs are $|P| + |h| + |p| + 128 = 1824$ bits and 128 bits, respectively. So, the total cost is $1824 + 128 = 1952$ bits. Next, ES_1 sends the reply message $\{B, w, T_1\}$ to U. So, the communication cost is $|P| + |h| + 128 = 1312$ bits. Thereafter, ES_1 sends the message $\{A', B, W_1, l, T_2\}$ to the edge server ES_2. So, the communication cost is $|P| + |P| + |h| + |p| + 128 = 2848$ bits. Therefore, total communication cost at ES_1 is $1312 + 2848 = 4160$ bits. After that, ES_2 sends the Message $\{B', w_2, T_3\}$ to

TABLE IV
COMPARATIVE ANALYSIS FOR COMMUNICATION COSTS (IN BITS)

Protocol	Security level	P_t/D_i	HS_i	HS_j
Wang et al. [17]	80	$\|P\|+\|h\|+\|p\|+2\times128=1952$	$3\|P\|+\|p\|+2\|h\|+2\times128=4160$	$\|P\|+\|h\|+128=1312$
	112	$\|P\|+\|h\|+\|p\|+2\times128=3552$	$3\|P\|+\|p\|+2\|h\|+2\times128=7872$	$\|P\|+\|h\|+128=2400$
	128	$\|P\|+\|h\|+\|p\|+2\times128=5120$	$3\|P\|+\|p\|+2\|h\|+2\times128=11520$	$\|P\|+\|h\|+128=3456$
	192	$\|P\|+\|h\|+\|p\|+2\times128=12160$	$3\|P\|+\|p\|+2\|h\|+2\times128=27904$	$\|P\|+\|h\|+128=8192$
	256	$\|P\|+\|h\|+\|p\|+2\times128=23808$	$3\|P\|+\|p\|+2\|h\|+2\times128=55040$	$\|P\|+\|h\|+128=16000$
BC-UADS	80	$\|P\|+3\|h\|=1504$	$2\|P\|+\|p\|+2\|h\|=2880$	$4\|P\|+\|p\|+4\|h\|=5248$
	112	$\|P\|+3\|h\|=2720$	$2\|P\|+\|p\|+2\|h\|=5568$	$4\|P\|+\|p\|+4\|h\|=10112$
	128	$\|P\|+3\|h\|=3840$	$2\|P\|+\|p\|+2\|h\|=8192$	$4\|P\|+\|p\|+4\|h\|=14848$
	192	$\|P\|+3\|h\|=8832$	$2\|P\|+\|p\|+2\|h\|=19968$	$4\|P\|+\|p\|+4\|h\|=36096$
	256	$\|P\|+3\|h\|=16896$	$2\|P\|+\|p\|+2\|h\|=39424$	$4\|P\|+\|p\|+4\|h\|=71168$

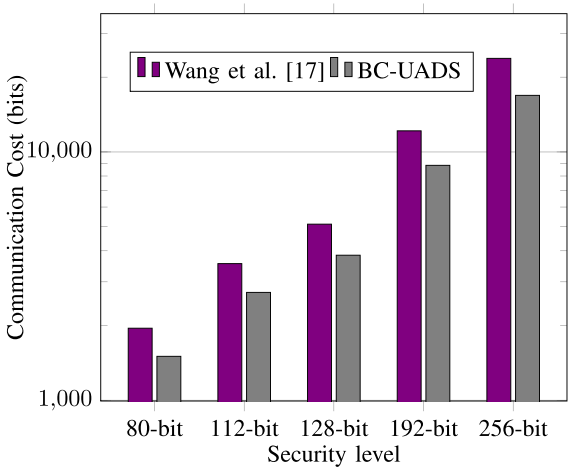

Fig. 14. Communication cost of P_t/D_i for different security level.

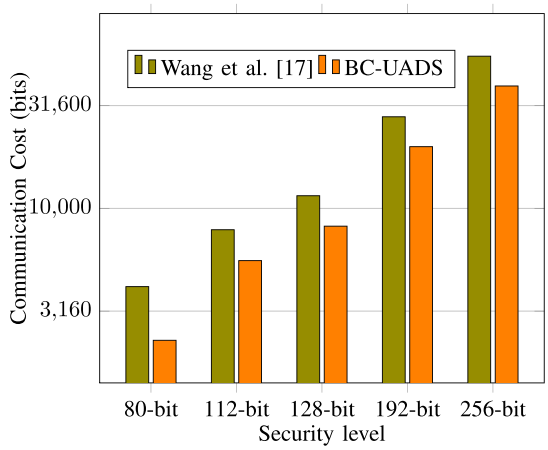

Fig. 15. Communication cost of HS_i for different security level.

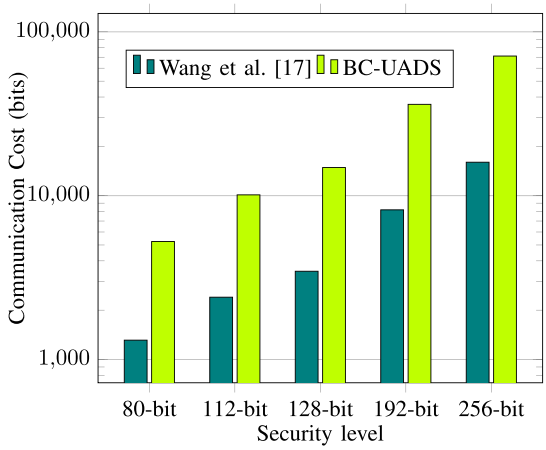

Fig. 16. Communication cost of HS_j for different security level.

U. So, the communication cost at ES_2 is $|P|+|h|+128=1312$ bits. In the same way, we further calculated the communication cost for security levels 112 bits, 128 bits, 192 bits, and 256 bits and the results are provided in Table IV. From the Fig. 14 and Fig. 15, we can conclude that communication costs at P_t and HS_i of the BC-UADS framework are less compared to Wang et al.'s scheme [17]. However, from Fig. 16, we found that the communication cost of HS_j of BC-UADS is higher than Wang et al.'s scheme.

We have compared the execution cost of the BC-UADS framework and Wang et al.'s scheme [17]. To calculate the execution cost, we used a Laptop (RAM: 8 GB, Intel(R) Core(TM) i5-1035G1 CPU@ 1.19 GHz, Ubuntu 22.04LTS OS) as client (P_t/D_i), and a Desktop (RAM: 8 GB, Intel(R) Core(TM) i7-10700 CPU@ 2.90 GHz, Ubuntu 22.04LTS OS) as server (HS_i/HS_j). The cryptographic operation on these systems are executed using the PBC library, and the results are given in Table V and Table VI, respectively. For five security levels, we have executed the execution time of the proposed BC-UADS framework and Wang et al.'s scheme [17], and the comparative results are given in Table VII. In the proposed BC-UADS framework, after initial login steps, P_t uses $6T_h+3T_{ecml}$ operations for authentication and session key agreement phase. So, the execution cost for security level of 80 bits can be calculated as $6T_h+3T_{ecml}=6\times0.003120+3\times3.4273\approx10.300$ ms. Similarly, the execution cost of HS_i is calculated as $10T_h+8T_{ecml}+T_{mad}+2T_{ecad}=10\times0.00022+8\times0.7604+0.000190+2\times0.003740\approx6.0930$ ms. The execution cost of HS_j is $8T_h+8T_{ecml}+2T_{ecad}+T_{mad}=8\times0.00022+8\times0.7604+2\times0.003740+0.000190\approx6.09263$ ms. Accordingly, we have calculated execution cost of the scheme in [17] for the security level of 80 bits. In this scheme, U needs the execution cost $7T_h+8T_{ecml}+3T_{ecad}+T_{se}+T_{mad}=7\times0.003120+8\times3.4273+3\times0.01989+0.45300+0.0017\approx27.9546$ ms. Similarly, the execution cost of ES_1 is $7T_h+10T_{ecml}+3T_{ecad}+T_{sd}+T_{mad}=7\times0.00022+10\times0.7604+3\times0.003740+0.02449+0.000190\approx7.64144$ ms. The execution cost of ES_2 is $5T_h+7T_{ecml}+3T_{ecad}=5\times0.00022+$

TABLE V
EXECUTION TIME (IN MS) OF VARIOUS CRYPTOGRAPHIC OPERATIONS FOR P_t/D_i

Notation	Description	80-bit	112-bit	128-bit	192-bit	256-bit
T_{ecad}	Elliptic curve point addition	0.019890	0.0420000	0.067210	0.2141200	0.7575200
T_{ecml}	Elliptic curve scalar point multiplication	3.4273	10.2005000	21.703480	118.430890	556.3865
T_h	Hash function	0.003120	0.002880	0.002750	0.004120	0.006570
T_{mad}	Modular addition	0.0017	0.001450	0.001190	0.001030	0.00189
T_{se}	AES encryption	0.45300	0.0789	0.07785	0.8262	0.100140
T_{sd}	AES decryption	0.085060	0.15460	0.137970	0.230580	0.30189

TABLE VI
EXECUTION TIME (IN MS) OF VARIOUS CRYPTOGRAPHIC OPERATIONS FOR HS_i/HS_j

Notation	Description	80-bit	112-bit	128-bit	192-bit	256-bit
T_{ecad}	Elliptic curve point addition	00.003740	00.00770	00.01359	00.04570	00.150340
T_{ecml}	Elliptic curve scalar point multiplication	00.7604000	02.2584000	04.6260000	24.5302000	110.2453000
T_h	hash function	0.00022	0.00019	0.00020	0.000210	0.000230
T_{mad}	Modular addition	0.000190	0.000150	0.000120	0.000160	0.000200
T_{se}	AES encryption	0.045830	0.05120	0.056910	0.05838	0.07395
T_{sd}	AES decryption	0.02449	0.030480	0.036770	0.04344	0.06460

TABLE VII
COMPARATIVE ANALYSIS FOR THE EXECUTION TIME (IN MS)

Protocol	Security Level	P_t/D_i	HS_i	HS_j
Wang et al. [17]	80	27.9546	7.64144	5.33512
	112	82.5406	22.63906	15.8328
	128	173.9277	46.3390	32.42377
	192	948.938	245.4841	171.8495
	256	4453.512	1102.97043	772.1692
BC-UADS	80	10.300	6.09307	6.0926
	112	30.6187	18.086	18.0842
	128	65.1269	37.0373	37.0369
	192	355.3173	196.3352	196.3348
	256	1669.1989	882.2655	882.2651

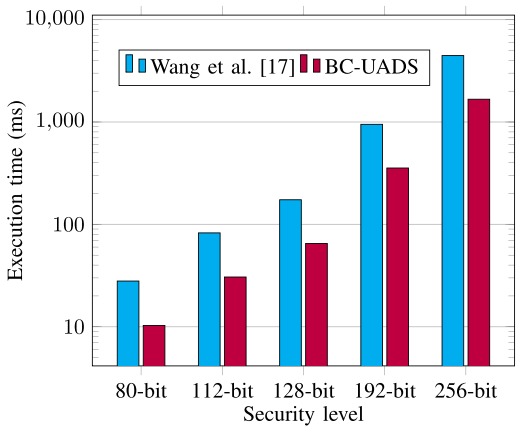

Fig. 17. Execution cost of P_t/D_i for different security level.

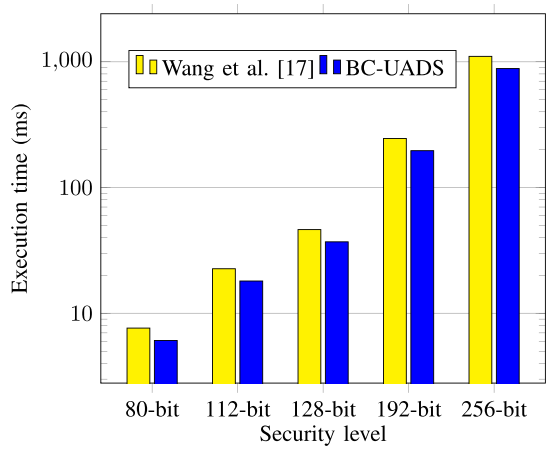

Fig. 18. Execution time of HS_i for different security level.

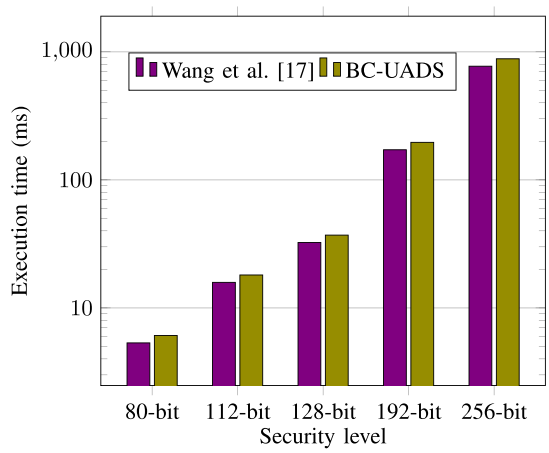

Fig. 19. Execution time of HS_j for different security level.

$7 \times 0.7604 + 3 \times 0.003740 \approx 5.33512$ ms. Similarly, we have computed the execution cost for the security level of 112 bits, 128 bits, 192 bits and 256 bits, and the results are shown in Fig. 17 for P_t/D_i, Fig. 18 for HS_i, and Fig. 19 for HS_j.

We have computed the storage requirements to store some login related credentials at the user's device (smartcard/mobile device). In the proposed BC-UADS framework, P_t/D_i holds a smartcard SC_t which stores the parameters $A_t = PID_t \oplus PWD_t = h(ID_t \parallel k_i) \oplus h(PW_t \parallel k_t)$, $V_t = h(PID_t \parallel PWD_t)$, $B_t = k_t \oplus h(PW_t \parallel \sigma_t)$, and τ_t. Therefore, the storage cost of the BC-UADS framework is $|h| + |h| + |p| + |P| = 160 + 160 + 512 + 1024 = 1856$ bits for the security level of 80 bits. In Wang et al.'s scheme [17], the U's device stores the private keys (s_u, x_u) and their public key (R_u, X_u). Therefore, the storage cost in [17] $|p| + |p| + |P| + |P| = 512 + 512 + 1024 + 1024 = 3072$ bits. Similarly, we have estimated the storage costs of the BC-UADS framework and Wang et al.'s scheme for other security levels in Table VIII,

TABLE VIII
COMPARATIVE ANALYSIS OF STORAGE COSTS (IN BITS)

Protocol	Security Level	P_t/D_i
Wang et al. [17]	80	3072
	112	6144
	128	9216
	192	23040
	256	46080
BC-UADS	80	1856
	112	3520
	128	5120
	192	12288
	256	24064

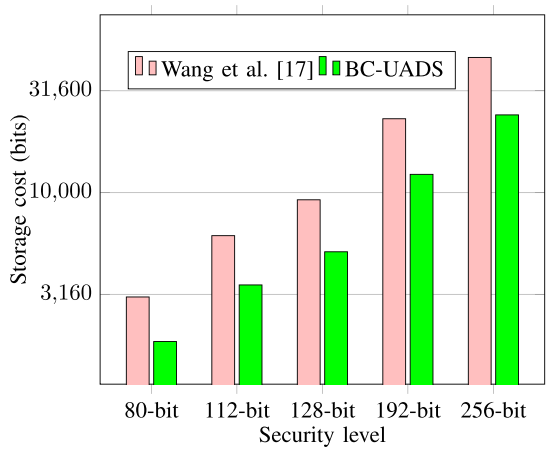

Fig. 20. Storage cost required at P_t/D_i.

TABLE IX
SYSTEM DETAILS

Parameter	Value
Model Name	Intel(R) Core(TM) i5-1035G1 CPU@ 1.19 GHz
Threads per core	2
Cores per socket	6
Sockets	1
Stepping	1
CPU max MHz	4600
CPU min MHz	800
L1d	288 KiB (6 instances)
L1i	192 KiB (6 instances)
L2	3 MiB (6 instances)
L3	12 MiB (1 instance)

and Fig. 20. From the storage requirement analysis, it is found that the BC-UADS framework is more efficient than Wang et al.'s scheme.

VII. SIMULATION OF THE BC-UADS

We have performed the simulation of the blockchain-based data-sharing technique among the hospital servers in the proposed BC-UADS framework. Table IX provides the system details of the simulation process. Fig. 21 provides an overview of the simulation process, where, once all the transactions are received, a leader node with the highest reputation (rank) takes responsibility for mining the new block. The leader node (minor) broadcasts the newly created block to the validator pool, which uploads, through the PoR consensus algorithm, whether the block is valid or not. Finally, if the block is verified and found valid, the new block is added to the blockchain. The

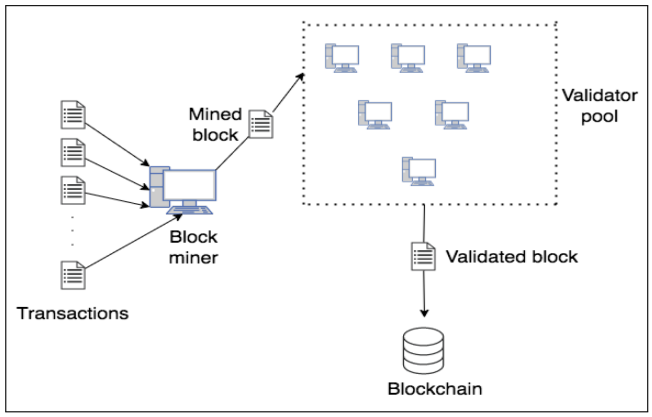

Fig. 21. Data flow diagram of the block generation, verification, and addition.

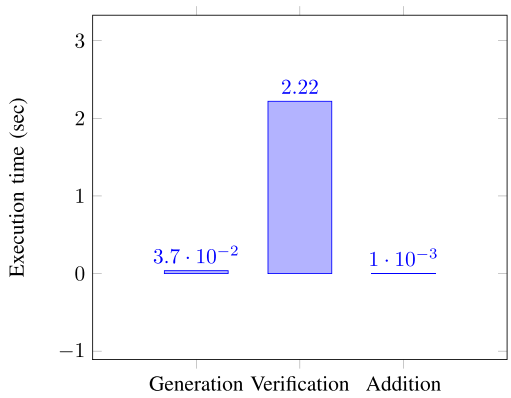

Fig. 22. Time for block generation, verification, and addition.

simulation assumes the presence of 1000 EHR (medical) data transactions and reputation transactions. We use a parallelized implementation found in [25] to compute the Merkel root. We ignored the communication cost of submitting these 1000 medical and reputation transactions since they depend on the connection speeds of the actual nodes used in the system. The block generation process takes a few milliseconds on our test setup. In this simulation environment, the total time to generate 1000 blocks is about 33 seconds.

For the block verification, we assigned 1000 verifiers for each block. The verified identity represents the hashed value of the verified identifier, and the verification status implies the result a particular verifier has given to a certain block. In this simulation, each verifier gives a vote (valid/invalid), and if the consensus is established, the block is considered valid. The total time to verify all the 1000 blocks is about 38 minutes on our setup. The comparison of time taken by the three primary operations is given in Fig. 22.

VIII. CONCLUSION

In this paper, we proposed a blockchain-based user authentication data sharing (BC-UADS) framework to preserve the transparency, immutability, and authenticity of patient's EHR data shared among different hospitals over the Internet. In medical data-sharing, integrity verification is a big challenge, which is solved using the blockchain technology. Once the integrity is verified, the proposed authentication mechanism is a suitable

solution for retrieving or accessing medical data among valid users like patients, doctors, or medical staffs. We implemented the data-sharing protocol of the BC-UADS framework using the PoR consensus algorithm. We analyzed the BC-UADS framework through the AVISPA tool and BAN logic model to validate the attack-resilience and mutual authentication property. The provable security of the proposed BC-UADS framework is analyzed in the RoM based on the hardness assumption of the ECCDH problem.

ACKNOWLEDGMENT

The authors acknowledge Rochester Institute of Technology, Kosovo, Europe.

REFERENCES

[1] I. Zikratov, A. Kuzmin, V. Akimenko, V. Niculichev, and L. Yalansky, "Ensuring data integrity using blockchain technology," in *Proc. IEEE 20th Conf. Open Innov. Assoc.*, 2017, pp. 534–539.

[2] A. Al Omar, M. Z. A. Bhuiyan, A. Basu, S. Kiyomoto, and M. S. Rahman, "Privacy-friendly platform for healthcare data in cloud based on blockchain environment," *Future Gener. Comput. Syst.*, vol. 95, pp. 511–521, 2019.

[3] S. Cao, G. Zhang, P. Liu, X. Zhang, and F. Neri, "Cloud-assisted secure eHealth systems for tamper-proofing EHR via blockchain," *Inf. Sci.*, vol. 485, pp. 427–440, 2019.

[4] Z. Chen, W. Xu, B. Wang, and H. Yu, "A blockchain-based preserving and sharing system for medical data privacy," *Future Gener. Comput. Syst.*, vol. 124, pp. 338–350, 2021.

[5] F. Tang, S. Ma, Y. Xiang, and C. Lin, "An efficient authentication scheme for blockchain-based electronic health records," *IEEE Access*, vol. 7, pp. 41678–41689, 2019.

[6] M. Wu, L. You, G. Hu, L. Li, and C. Cao, "A blockchain-based hierarchical authentication scheme for multiserver architecture," *Secur. Commun. Netw.*, vol. 2021, pp. 1–20, 2021.

[7] G. Mwitende, I. Ali, N. Eltayieb, B. Wang, and F. Li, "Authenticated key agreement for blockchain-based WBAN," *Telecommun. Syst.*, vol. 74, no. 3, pp. 347–365, 2020.

[8] C.-M. Chen, X. Deng, S. Kumar, S. Kumari, and S. Islam, "Blockchain-based medical data sharing schedule guaranteeing security of individual entities," *J. Ambient Intell. Humanized Comput.*, vol. 12, pp. 1–10, 2021.

[9] X. Cheng, F. Chen, D. Xie, H. Sun, and C. Huang, "Design of a secure medical data sharing scheme based on blockchain," *J. Med. Syst.*, vol. 44, no. 2, pp. 1–11, 2020.

[10] X. Liu, W. Ma, and H. Cao, "MBPA: A. medibchain-based privacy-preserving mutual authentication in TMIS for mobile medical cloud architecture," *IEEE Access*, vol. 7, pp. 149282–149298, 2019.

[11] L. Xiong, F. Li, S. Zeng, T. Peng, and Z. Liu, "A blockchain-based privacy-awareness authentication scheme with efficient revocation for multi-server architectures," *IEEE Access*, vol. 7, pp. 125840–125853, 2019.

[12] A. Yazdinejad, G. Srivastava, R. M. Parizi, A. Dehghantanha, K.-K. R. Choo, and M. Aledhari, "Decentralized authentication of distributed patients in hospital networks using blockchain," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 8, pp. 2146–2156, Aug. 2020.

[13] N. Garg, M. Wazid, A. K. Das, D. P. Singh, J. J. P. C. Rodrigues, and Y. Park, "BAKMP-IoMT: Design of blockchain enabled authenticated key management protocol for Internet of Medical Things deployment," *IEEE Access*, vol. 8, pp. 95956–95977, 2020.

[14] S. K. Singh and J. H. Park, "TaLWaR: Blockchain-based trust management scheme for smart enterprises with augmented intelligence," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 626–634, Jan. 2023.

[15] P. Sharma, S. Namasudra, R. G. Crespo, J. Parra-Fuente, and M. C. Trivedi, "EHDHE: Enhancing security of healthcare documents in IoT-enabled digital healthcare ecosystems using blockchain," *Inf. Sci.*, vol. 629, pp. 703–718, 2023.

[16] B. K. Rai, "PcBEHR: Patient-controlled blockchain enabled electronic health records for healthcare 4.0," *Health Serv. Outcomes Res. Methodol.*, vol. 23, no. 1, pp. 80–102, 2023.

[17] W. Wang, H. Huang, L. Xue, Q. Li, R. Malekian, and Y. Zhang, "Blockchain-assisted handover authentication for intelligent telehealth in multi-server edge computing environment," *J. Syst. Archit.*, vol. 115, 2021, Art. no. 102024.

[18] S. Kaur, S. Chaturvedi, A. Sharma, and J. Kar, "A research survey on applications of consensus protocols in blockchain," *Secur. Commun. Netw.*, vol. 2021, pp. 1–22, 2021.

[19] P. Soni, A. K. Pal, S. H. Islam, A. Singh, and P. Kumar, "Provably secure and biometric-based secure access of e-governance services using mobile devices," *J. Inf. Secur. Appl.*, vol. 63, 2021, Art. no. 103016.

[20] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, "Proof of reputation: A reputation-based consensus protocol for blockchain based systems," in *Proc. ACM Int. Electron. Commun. Conf.*, 2019, pp. 131–138.

[21] F. Gai, B. Wang, W. Deng, and W. Peng, "Proof of reputation: A reputation-based consensus protocol for peer-to-peer network," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2018, pp. 666–681.

[22] D. Wang, X. Zhang, Z. Zhang, and P. Wang, "Understanding security failures of multi-factor authentication schemes for multi-server environments," *Comput. Secur.*, vol. 88, 2020, Art. no. 101619.

[23] Q. Xie, D. S. Wong, G. Wang, X. Tan, K. Chen, and L. Fang, "Provably secure dynamic ID-Based anonymous two-factor authenticated key exchange protocol with extended security model," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 6, pp. 1382–1392, Jun. 2017.

[24] K. Parai, D. S. Gupta, and S. H. Islam, "IoT-ID3PAKA: Efficient and robust ID-3PAKA protocol for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10304–10313, Mar. 2024.

[25] A. Flangas, A. Cuellar, M. Reyes, and F. C. Harris Jr., "Parallelized c implementation of a merkle tree," in *Proc. ITNG 18th Int. Conf. Inf. Technol.-New Gener.*, 2021, pp. 107–114.

Preeti Soni received the B.E. degree from Chhattisgarh Swami Vivekanand Technical University, Bhilai, India, in 2012, and the M.Tech and Ph.D. degrees from the Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines) Dhanbad, India, in 2017 and 2023, respectively. She is currently an Assistant Professor with the Department of Computer Science and Engineering, National Institute of Technology Hamirpur, Hamirpur, India. Her research interests include cryptography and authentication.

SK Hafizul Islam (Senior Member, IEEE) received the M.Sc. degree in applied mathematics from Vidyasagar University, Midnapore, India, in 2006, and the M.Tech. degree in computer application and the Ph.D. degree in computer science and engineering from the Indian Institute of Technology [IIT (ISM)] Dhanbad, Dhanbad, India, in 2009 and 2013, respectively, under the INSPIRE Fellowship Ph.D. Program (funded by the Department of Science and Technology, Government of India). He is currently an Assistant Professor with the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani (IIIT Kalyani), West Bengal, India. Before joining IIIT Kalyani, he was an Assistant Professor with the Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani (BITS Pilani), Pilani, India. He has more than 12 years of teaching experience and 15 years of research experience. He has authored or coauthored 150 research articles in journals and conference proceedings of international reputes. His research interests include cryptography, information security, neural cryptography, lattice-based cryptography, IoT & blockchain security, and deep learning. He has edited four books for the publishers Scrivener-Wiley, Elsevier, and CRC Press. He is an Associate Editor for IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE ACCESS, *International Journal of Communication Systems* (Wiley), *Telecommunication Systems* (Springer), *IET Wireless Sensor Systems, Security and Privacy* (Wiley), *Array-Journal* (Elsevier), and *Journal of Cloud Computing* (Springer). He was the recipient of the University Gold Medal, the S. D. Singha Memorial Endowment Gold Medal, and the Sabitri Parya Memorial Endowment Gold Medal from Vidyasagar University, in 2006. He was also the recipient of the University Gold Medal from IIT(ISM) Dhanbad in 2009 and the OPERA award from BITS Pilani in 2015. He is a member of ACM.

Arup Kumar Pal received the B.Tech. degree in computer science and engineering from the Government College of Engineering and Textile Technology, Berhampore, India, in 2006, and the Ph.D. degree in computer science and engineering with IIT (ISM) Dhanbad in 2011. He is currently an Associate Professor with the Department of Computer Science and Engineering, Indian Institute of Technology (ISM) Dhanbad, Dhanbad, India. Prior to joining this institution, he held the position of a Lecturer with the Department of Computer Science and Engineering, NIT Jamshedpur from April 2011 to December 2011. With more than 12 years of experience in teaching and research, he has contributed to more than 100 research papers published in esteemed journals and international conference proceedings. He was the sole or main guide for 11 Ph.D. students who successfully graduated under his guidance. His research interests include data compression, multimedia security, CBIR, and blockchain. He was the recipient of the several best paper awards for his research presentations at various international conferences. He was also the recipient of several R&D projects from prestigious organizations, such as CSIR, DST, MeitY, and C3iHUB IIT Kanpur. He has also organized several Faculty Development Programs (FDPs) in the area of Image Processing and Cryptography and has actively participated as an Advisor Committee member and Technical Programme Committee (TPC) member in various conferences and workshops

Nimish Mishra received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology Kalyani, Kalyani, India, in 2022. He has authored and coauthored more than 15 research articles in international journals and conference proceedings. His primary research interests include cryptography, hardware security, and vulnerability research.

Debabrata Samanta (Senior Member, IEEE) received the Ph.D. degree in computer science and Engineering from the National Institute of Technology, Durgapur, India, in SAR Image Processing. He is currently the Program Head and an Assistant Professor with the Department of Computing and Information Technologies, Rochester Institute of Technology, Kosovo, Prishtina, Serbia. He is keenly interested in interdisciplinary research and development. He has experience spanning fields of SAR image analysis, video surveillance, a heuristic algorithm for image classification, deep learning framework for detection and classification, blockchain, statistical modelling, wireless ad hoc networks, and natural language processing. He has completed six Consultancy Projects. Under his guidance, four students have accomplished their Ph.D. degrees. He has received funding of 8,110 USD under the Open Access Publication fund. He received funding under the International Travel Support Scheme in 2019 for attending the conference in Thailand. He has received a travel grant to speak at conferences and seminars, etc. for two years since July 2019. He owns 22 Patents (four Design Indian Patents and two Australian patents Granted, 16 Indian Patents published) and two copyrights. He has authored and coauthored more than 235 research papers in an international journal (SCI/SCIE/ESCI/Scopus) and conferences, including IEEE, Springer, and Elsevier Conference proceedings. He is a co-author of 13 books and the co-editor of 15 books, available for sale on Amazon and Flipkart. He has presented various papers at international conferences and received Best Paper awards. He is the author and co-author of 14 Book Chapters. He was the recipient of "Scholastic Award" at the 2nd International Conference on Computer Science and IT Application, CSIT-2011, Delhi, India. He is also an acquisition editor for Springer, Wiley, CRC, Scrivener Publishing LLC, Beverly, USA, and Elsevier. He is an Associate Life Member of the Computer Society of India, and a Life Member of the Indian Society for Technical Education. He is a Convener, Keynote speaker, Session chair, Co-chair, Publicity chair, Publication chair, Advisory Board, and Technical Program Committee member in many prestigious International and National conferences. He has invited speakers at several Institutions.