Received 4 July 2024, accepted 25 July 2024, date of publication 6 August 2024, date of current version 2 September 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3439611*

RESEARCH ARTICLE

# RSHealth: A Ring Signature Scheme for Identity Anonymization and Transaction Privacy in Blockchain Based E-Healthcare Systems

**PUNAM PRABHA AND KAKALI CHATTERJEE, (Member, IEEE)**
Department of Computer Science and Engineering, National Institute of Technology, Patna, Bihar 800005, India
Corresponding author: Punam Prabha (punamp.phd18.cs@nitp.ac.in)

**ABSTRACT** The current healthcare infrastructure faces significant challenges, including security vulnerabilities, privacy breaches, data inconsistencies, and overly accessible health records. These issues highlight the urgent need for comprehensive reforms in information management and patient data protection. To address the current issues, the present work proposes RSHealth framework, an e-healthcare ring signature scheme. It is a promising solution for ensuring anti-tampering of transaction fie and identity anonymization of sender healthcare stakeholder during transmission from main healthcare community to a branch healthcare community, maintaining compliance with data privacy regulation. The RSHealth scheme includes five functions: TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen(). These functions respectively initialize system parameters, generate keys, create ring signatures, verify authenticity, and reveal signer identities by authorized receiver healthcare stakeholders. Extensive experimentation is performed for aforementioned framework to judge the latency and throughput; varying sender healthcare stakeholder and sizes of transaction. The performance analysis shows that increasing sender healthcare stakeholder and transaction sizes leads to increased throughput and decreased latency.

**INDEX TERMS** Blockchain, e-healthcare system, privacy, ring signature, security.

## I. INTRODUCTION

All India Institute of Medical Sciences (AIIMS) stands as a cornerstone of India's healthcare system, storing the medical histories of many prominent individuals. The cyber-attack on November 23, 2022, highlighted the vulnerability of medical institutions to data breaches, emphasizing the need for robust security and privacy measures in remote patient treatment [1]. Key challenges include ensuring identity anonymity, anti-tampering, and authentication in accordance with organizational standards [2], [3], [4].

Due to inherent characteristics of decentralization, tamper resistance, anonymity, and public verifiability, blockchain has emerged as the foundational technology underpinning digital cryptocurrencies like Bitcoin and Ethereum [5], [6]. The exceptional technical attributes of blockchain extend its application beyond digital cryptocurrencies, finding widespread use in various other fields [7], [8]. In blockchain systems, the lack of centralized data processing and maintenance promotes transparency and quick consensus among nodes. Blockchain's capabilities extend to smart contracts, decentralized finance, supply chain transparency, secure identity verification, transparent voting systems, medical records management, intellectual property protection, peer-to-peer energy trading, virtual asset ownership, and platform interoperability [9], [3], [10], [11].

By providing data integrity, security, and transparency, blockchain's ledger management capabilities enhance data privacy frameworks [12]. Decentralized data storage reduces risks, while its immutable ledger and powerful cryptography secure sensitive information [13]. Blockchain helps organizations enhance stakeholder confidence and comply with regulations, improving their data privacy policies [14], [15]. However, this transparency can also pose challenges related

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Ni.

to data privacy, potentially leading to information leakage [16]. Numerous blockchain privacy protection schemes have been proposed to enhance the anonymity of blockchain technology and safeguard the privacy of user identities and transaction data [8], [17].

Rivest et al [18] introduced the concept of signature with anonymity through the innovative ring signature-based e-healthcare system (EHS). This system allows a user to sign messages while remaining indistinguishable from other users within a specified group, referred as a ring [19]. The ring signature is a unique type of group signature [8].

An organization-level privacy protection mechanism for blockchain based transaction systems aims to facilitate the transformation of transactions between individual users to organizations. This transition not only enhances privacy but also establishes a balance between privacy protection and security supervision [15]. It urges blockchain based ring signature scheme which allow for anonymous yet accountable transactions within predefined members, thereby enhancing privacy while still allowing for oversight [8].

Implementing simplified cryptography and straightforward signature schemes in blockchain-based EHS can greatly enhance security and data privacy. Elgamal cryptography and its digital signature schemes ensure secure communication between parties [20], [21]. Efficiently addressing security concerns while preserving data privacy is crucial for managing medical data, especially in environments vulnerable to breaches or unauthorized access. Blockchain technology reduces network communication latency and provides rapid response times, making it effective for this purpose [8]. In [22], a blockchain-based EHS has been developed with four functional layers: the system layer (SL), inter-network layer (IL), blockchain layer (BL), and cloud layer (CL), arranged to facilitate medical treatment as shown in Figure 1.

In medical emergencies or when facilities are lacking, the transaction file of sender healthcare stakeholder (SHS) can be transferred from the main healthcare community (MHC) to the branch healthcare community (BHC). Any SHS can join MHC after thorough verification by the BL. However, the current EHS lacks comprehensive facilities to address multiple health complexities simultaneously. Integrating Elgamal cryptography into a ring signature scheme within a blockchain-based EHS can enforce privacy measures while maintaining high security levels. This approach enables a more robust and flexible framework for secure, decentralized transactions.

Table 1 displays the symbols and abbreviations used in the present paper.

## A. OBJECTIVE

The primary objective of the proposed EHS is to securely facilitate the transfer of transaction files from SHS to a receiver healthcare stakeholder (RHS) through their registered organizations, while maintaining data privacy and security. For instance, if an SHS diagnosed with AIDS is registered with MHC and requires services beyond what MHC can offer, such as a CT scan, their transaction file must be transferred to BHC. This data transfer, particularly between SL and BL, risks disclosing the SHS's identity and transaction file privacy.

This work aims to achieve two key objectives within the EHS framework:

- **Implement Anti-Tampering**: Ensure the integrity of medical data during transfer.
- **Establish Identity Anonymity**: Protect the SHS's identity from being disclosed during the data transfer process.

**TABLE 1.** Acronyms used in the proposed framework.

| Symbol | Abbreviation |
|---|---|
| EHS | e-healthcare system |
| SL | system layer |
| IL | inter-network layer |
| BL | blockchain layer |
| CL | cloud layer |
| MHC | main healthcare community |
| BHC | branch healthcare community |
| SHS | sender healthcare stakeholder |
| RHS | receiver healthcare stakeholder |
| SSSL1 | subsystem sublayer 1 |
| SSSL2 | subsystem sublayer 2 |
| DPL | distributed patient ledger |
| $m\_f$ | medical facility |
| $SHS\_id$ | SHS identity |
| $xray\_r$ | x-ray report |
| $blood\_r$ | blood report |
| $urine\_r$ | urine report |
| $sym$ | symptom |
| $sym'$ | similar symptom |
| $antibody$ | antibody test |
| $antigen$ | antigen test |
| $Tl_i$ | transaction list |
| $pseudo\_trans\_file$ | pseudo transaction file |
| $trans\_file$ | transaction file |
| $V_{1i}$ | first verification parameter |
| $V_{2i}$ | second verification parameter |
| $H_0(i)$ | first hash function |
| $H_1(i)$ | second hash function |
| $id*$ | pseudo identity |
| $test_i$ | list of test |
| $disease_i$ | list of disease |
| $nat$ | nucleic acid test |
| $SHS\_file$ | Sender healthcare stakeholder file |
| $Q$ | queue |
| $aids$ | AIDS disease |
| $sys\_p$ | system parameter |
| $sec\_p$ | security parameter |
| $ct\_scan$ | CT scan |
| $S_{RSHealth,d}$ | Success for solving disease |
| $id$ | original identity of SHS |
| $SHS(l)$ | length of SHS's identity |
| $IL\_A$ | attacker on the inter-network layer |
| $SL\_C$ | challenger on the system layer |

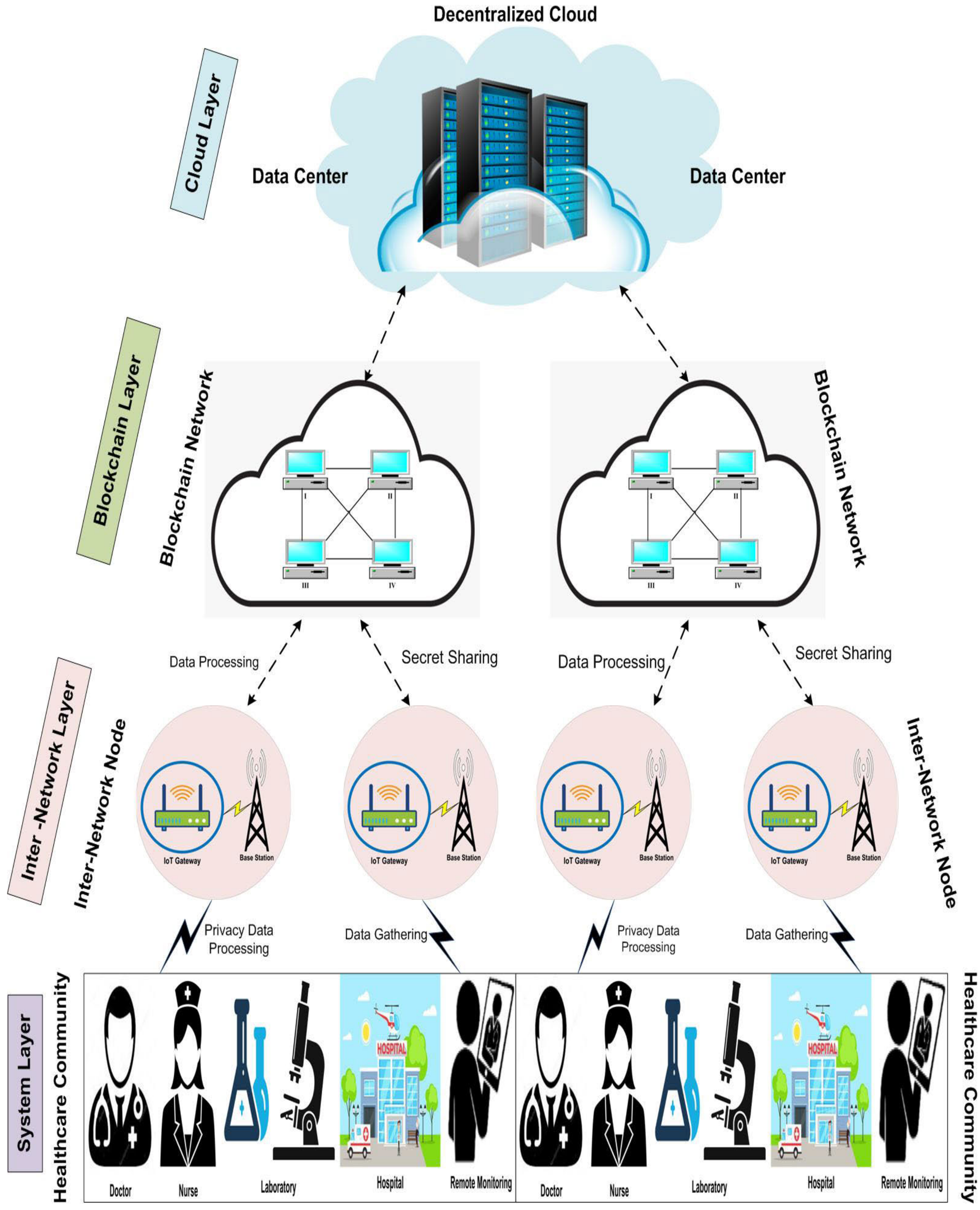


**FIGURE 1.** Existing architecture of e-healthcare system [22].

## B. RESEARCH CONTRIBUTIONS

Ensuring the security of confidential personal information is a paramount concern during the transmission of transaction files from SHS to RHS. In this context, SHS aims to maintain identity anonymity during this transmission of transaction files. RSHealth is employed within the blockchain-based

EHS to oversee security and mitigate malicious activities at IL. The main research contributions of this paper are outlined as follows:

- Implementing a novel anti-tampering measure within blockchain based EHS, enhancing protection against malicious activities and unauthorized access.
- Implementing a novel data privacy protection framework, ensuring identity anonymization of SHS's sensitive information during file transmission.
- Analysing the performance of the implemented framework to demonstrate increased throughput and decreased latency as the number of SHS or transaction size increases.

In Section II, an overview of background studies, encompassing layered architecture of EHS, and related works are discussed. Section III elaborates on the proposed RSHealth framework and its functions in blockchain based EHS for identity anonymization and anti-tampering of SHS's sensitive information. The security analysis, experimental analysis, and performance analysis of the proposed framework is evaluated in Section IV. Finally, Section V summarizes key findings and discusses implications for future research.

## II. BACKGROUND

As technology continues to advance, the healthcare sector has evolved with the integration of cutting-edge technologies such as internet of thing (IoT), internet of medical thing (IoMT), blockchain technology. These systems enable simultaneous monitoring and control of various aspects of a patient's physiology, replacing standalone conventional devices. These modern medical device systems represent a distinctive category of EHS due to their embedded software, networking capabilities, and the complex physical dynamics.

### A. E-HEALTHCARE SYSTEM

The proposed RSHealth scheme applied in EHS adopts a blockchain-empowered cloud architecture to maintain anonymization using ring signature throughout the life cycle of the transaction file. To achieve this objective, various members of the ring are defined in EHS, as depicted in the Figure 2.

- **System layer (SL):** In this layer, various interested SHS join a queue and register with the designated organization. This layer plays a crucial role from the initialization of transaction registration to the closure of transactions. SL is further divided into two sublayers:
  - **Subsystem sublayer 1 (SSSL1):** In this sublayer, SHS represents the patients, while RHS acts as the registered guardian of the patient. SHS is authorized with MHC, while the RHS is authorized with BHC. Transactions are sent by SHS via MHC, and received by RHS via BHC. RHS receives the file only upon successful completion of the life cycle of the transaction file movement process.
  - **Subsystem sublayer 2 (SSSL2):** MHC and BHC function as organizations where hospitals, nurses, lab technicians, and doctors generates an organization key for SHS and RHS respectively. MHC and BHC grants affiliation only upon receiving a request from SHS and RHS for the issuance of an organization certificate.
- **Inter-network layer (IL):** The transfer of transactions from SL to BL and vice versa is accomplished through IL. IL is responsible for creating duplicate files of SHS and forwarding them to BL for verification purposes.
- **Blockchain layer (BL):** In this layer, the blocks are interconnected and maintain a Distributed Patient Ledger (DPL) for the facilities available at BHC. BL verifies these facilities with the needed medical facilities of SHS, and send verified transactions relay signal to BHC.
  Additionally, BL makes use of smart contracts [which are specified in the appendix] to guarantee that the actions of SHS data processing are in accordance with the legal standards. BL also makes use of the cryptographic methods in order to guarantee the safety and integrity of data, which addresses important concerns in the healthcare industry. Through the use of BL, SHS is able to maintain ownership over their medical records and securely share them with healthcare professionals, thereby improving both their privacy and their data integrity.
- **Cloud layer (CL):** The verified transactions relay signal are stored in this layer. External transaction records, other than the original transaction file of SHS, are stored on a cloud storage server.

### B. RELATED WORK

The domain of EHS emerges as a particularly anticipated area for advancement, especially in light of the COVID-19 pandemic and the subsequent post-pandemic landscape [3]. In this section, the overarching security and privacy of EHS, IoMT, and blockchain technology across diverse domains are focused in depth.

With the adoption of cloud-based solutions for the collection and storage of sensitive personal health information, there arises a necessity to share this data among various stakeholders within the network. Consequently, healthcare systems become susceptible to threats pertaining to confidentiality, integrity, and privacy. In this context, secure data access through such systems emerges as a significant challenge [23].

A blockchain-based privacy-preserving solution had been developed in order to solve the security issues present in cloud-assisted healthcare systems [12]. The work [10] is focused on deterministic smart contracts and their vulnerability to different attacks. Moreover, a trust-based consensus and optimization method is suggested in [9] to improve blockchain efficiency in IoT systems. Similarly, a blockchain-based framework is presented for protecting IoT data in smart city applications [24].

In [6], a blockchain-based system with improved encryption is described for safe electronic healthcare record storage

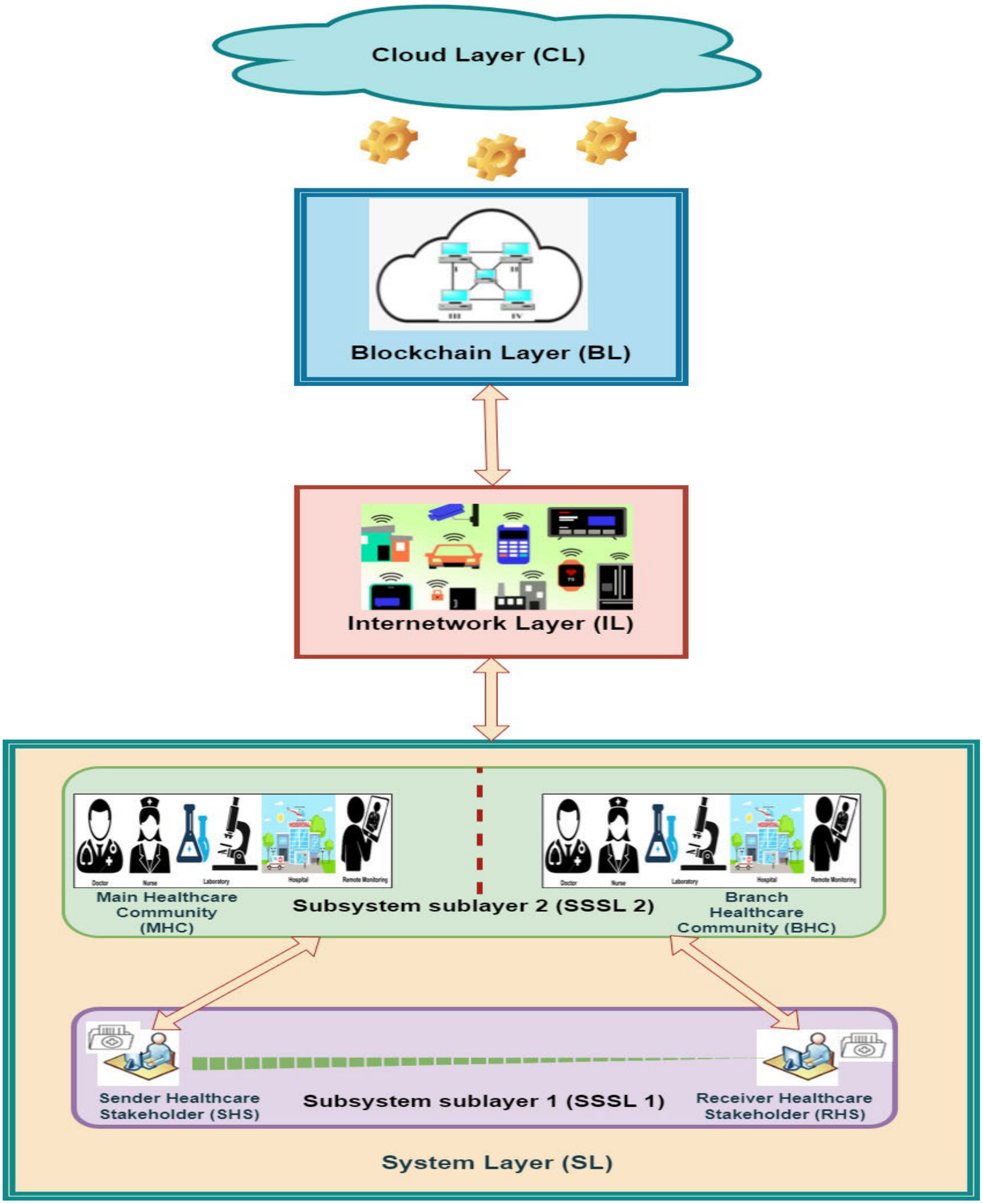

Cloud Layer (CL)
Blockchain Layer (BL)
Internetwork Layer (IL)
Doctor
Nurse
Laboratory
Hospital
Remote Monitoring
Main Healthcare Community (MHC)
Subsystem sublayer 2 (SSSL 2)
Branch Healthcare Community (BHC)
Sender Healthcare Stakeholder (SHS)
Subsystem sublayer 1 (SSSL 1)
Receiver Healthcare Stakeholder (RHS)
System Layer (SL)

**FIGURE 2.** **Layered architecture of e-healthcare system.**

and transfer in mobile cloud settings. A blockchain-enabled access control and key management protocol is presented in [2] to protect IoMT-based EHS. In [25], improvement in the industrial IoT network security is suggested by combining blockchain technology with advanced data processing and encryption methods. In IoMT healthcare systems, Fortified-chain [13] gains efficiency, security, and privacy; but, it does not address blockchain energy consumption, system scalability for large-scale deployment, or interoperability problems with current healthcare infrastructure. Fortified-Chain 2.0 [14] addresses data privacy, security, and latency; However, blockchain energy consumption, and scalability issues still not been addressed.

The integration of artificial intelligence into the healthcare system aims to tackle ethical, legal, and technological challenges to enhance patient care [11]. The medical blockchain ensures patient privacy and data accuracy; however, it struggles with scalability, high computing overhead, and compatibility with current healthcare systems [16]. Solutions proposed in previous studies have not adequately addressed issues such as system compatibility, latency, and processing costs [5]. Additionally, privacy leaks and security vulnerabilities in blockchain technology pose significant barriers to its adoption in EHS [7]. Addressing these concerns is essential to improve the anonymity of blockchain transactions and protect sensitive identity and transaction data in EHS [26].

Numerous studies, as shown in Table 2, focus on improving security and privacy measures. In this context, the identity anonymity and anti-tampering features of ring signatures enhance privacy [17], [27], making them especially useful for electronic payments and auctions [28].

From the aforementioned study, the following research challenges have been identified:

- Despite the numerous advantages of blockchain-based EHS (such as: traceability, decentralization, security and privacy), a significant challenge remains in ensuring sufficient secrecy for SHS's data transmitted over inside and outside system layer.
- The inherent anonymity of blockchain provides a certain level of privacy protection, but it can also bypass the oversight of relevant organizations.
- A blockchain-based EHS necessitates the implementation of a RSHealth to establish a robust data privacy framework.
- In the context of blockchain-based ring signatures, the implementation of anti-tampering measures, and identity anonymization should be designed with consideration for an organization-friendly environment.

## III. PROPOSED RSHealth: RING SIGNATURE SCHEME FOR BLOCKCHAIN BASED E-HEALTHCARE SYSTEM

RSHealth is incorporated into EHS for transaction privacy. At the initial stage, the RSHealth is utilized within SL to create blocks using transaction file of SHS, allows set of multiple parties namely, (SHS, MHC) and (BHC, RHS) to collectively sign a transaction without revealing the identity of SHS. Additionally, the ElGamal public key cryptography [20] and ElGamal digital signature scheme [21] are employed between SSSL1 and SSSL2 to generate organization keys, ensuring the security of the original credentials of SHS. After that, these facilitate validation in BL without disclosing which specific participant in SL produced the signature, thus preserving privacy. At the end, it enables authorized RHS to open the transaction file. These concepts are essential for formulating functions of RSHealth: TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen().

### A. WORKING OF THE PROPOSED FRAMEWORK

In the proposed framework, RSHealth is designed with a focus on organization-friendly features to offer robust security supervision and privacy protection capabilities. When SHS needs to send a file via MHC, the broadcasted file only displays the record of SHS generated by MHC, rather than SHS's identity. Simultaneously, BL validates the record of SHS generated by MHC using a proposed RSHealth. After successful validation, a transaction relay signal is sent to SL via IL. In SL, BHC transfers the successful transaction relay information to the registered RHS. Meanwhile, additional information is stored on CL for potential future treatment purposes.

The operational functionalities of the proposed EHS as seen in Figure 3, are outlined as follows:

1) EHS is starting its activities from a point denoted as SSSL1.
2) One of the Interested SHS from a queue has sent a request to MHC for the issuance of an organization certificate using TFSetup().
3) MHC applies TFKeygen() to issue a certificate of organization for SHS.
4) SHS anonymizes its identity using TFRing() and conducts a transaction with MHC.
5) MHC conceals the credentials of SHS and forwards them to BL via IL.
6) BL validates the transaction with assistance from DPL using TFVerify() and sends transaction relay information to BHC via IL.
7) BHC provides the required medical facilities and sends transaction relay information to RHS.
8) RHS obtains the transaction file from BHC using TFOpen(). The successful transaction relay information is stored on CL using Algorithm 1.

The proposed RSHealth is designed with certain assumptions addressed in the next subsection. Moreover, the security model of the proposed framework and RSHealth based transaction file privacy protection are described in-detail in the subsequent subsections.

### B. ASSUMPTIONS

*Definition 1 (If the Length of SHS Identity is Large Enough, Then it is Considered OK Otherwise Ignored):* Define a function $shs(l)$. When $l_0$ exists for any $p > 0$, and $l \geq l_0$ exists, so that $shs(l) \leq 1/l^p$, then $shs(l)$ is ignored.

*Definition 2 (Transaction File of SHS Must be Large Enough and Contains Multiple Transactions Including Tests and Reports):*Transaction file *trans_file* of SHS contains

**TABLE 2. Literature survey summary.**

| Author's work | Year | Think about security | Think about privacy | Aim of the work | Limitation |
|---|---|---|---|---|---|
| Deepak et al. [29] | [2019] | yes | yes | Anonymity-based user authentication protocols are favored for addressing privacy preservation concerns within the IoMT ecosystem. | The solution involves a complex mechanism. |
| Zheng et al. [15] | [2020] | yes | yes | An organization-level privacy protection mechanism for blockchain-based transaction systems, facilitating the transition from individual user transactions to organizational transactions. This mechanism aims to strike a balance between privacy protection and security supervision | Not specifically designed for use by healthcare professionals. |
| Li et al. [8] | [2020] | yes | yes | The complete anonymity provided by ring signatures ensures both the security of data and user identity privacy in blockchain applications. | Not specifically tailored for use by healthcare professionals. |
| Hara et al. [19] | [2021] | yes | yes | A generic construction of ring signature with unconditional anonymity in the plain model. | Not specifically designed for use by healthcare professionals |
| Egala et al.[13] | [2021] | yes | yes | Develop a blockchain-based decentralized architecture for IoMT healthcare to enhance EHR security, privacy, and efficiency. | Fails to address security supervision facilities. |
| Wu et al. [30] | [2021] | yes | yes | An edge computing paradigm is employed to implement a policy and role-based access control scheme based on local differential privacy. | Lacks adequate security supervision measures. |
| Zheng et al. [5] | [2022] | yes | yes | A blockchain-based scheme for secure and privacy-preserving patient health information sharing to improve diagnosis accuracy in EHS. | Does not address interoperability, latency, or blockchain computational overhead issues. |
| Shreya et al. [23] | [2022] | yes | no | A lightweight authentication technique is used, incorporating a privacy-preserving schema through fully homomorphic encryption. | The anonymity of healthcare professionals' information is not adequately addressed. |
| zhang et al.[12] | [2022] | yes | yes | To tackle security issues in cloud-based EHS using blockchain-based privacy-preserving solution. | Vulnerable to breaches; doctors might collude to alter/sell patient data. |
| Padma et al.[10] | [2022] | yes | no | Emphasizes deterministic smart contracts and attacks. | Lack anonymization of patients' health details. |
| Lee et al.[16] | [2022] | yes | yes | Blockchain and smart contract-based electronic medical record sharing to secure data accuracy and patient privacy. | Lacks security oversight tools that are easy for organizations to use. |
| Kumar et al. [31] | [2023] | yes | yes | A reputation-based blockchain framework is suggested as a means to uphold patient data security while also safeguarding privacy. | Lacks organization-friendly security supervision mechanisms. |
| Wazid et al.[2] | [2023] | yes | no | Blockchain-based access control and key management for IoMT-based EHS | It ignores scalability, interoperability, and latency with healthcare infrastructure. |
| Egala et al.[14] | [2023] | yes | yes | Fortified-Chain 2.0, a decentralized blockchain-based EHS, improves data privacy, security, and latency. | Does not have any security supervision measures that are organization-friendly. |
| Padma et al.[24] | [2024] | yes | yes | Secure IoT data in smart city applications with blockchain. | Not relevant to the healthcare industry. |
| Verma et al.[6] | [2024] | yes | yes | Secure healthcare data storage and transfer in mobile cloud environments using blockchain and enhanced encryption. | Underestimation of scalability and interoperability under the supervision of an organization. |
| Padma et al.[9] | [2024] | yes | yes | Addresses network overhead and energy consumption in blockchain for IoT. | Not address latency and throughput challenges. |
| Bobde et al.[25] | [2024] | yes | no | Blockchain integration with strong encryption and data processing improves Industrial IoT network security. | Does not address encryption and blockchain computational overhead. |
| Williamson et al. [11] | [2024] | yes | no | Critically assess healthcare AI integration's ethical, legal, and technological challenges to improve patient care. | Does not address implementation difficulties like healthcare resource constraints. |

report *report* including (xray report $xray\_r$, blood report $blood\_r$, urine report $urine\_r$) and test *test* including (nucleic acid test *nat*, Ct-scan $ct\_scan$, antibody *antibody*, antigen *antigen*), where multiple reports are available. The probability of finding specific disease *disease* in a limited time is $Adv_{disease,trans\_file}(l) = P_r[disease(report, test) = m\_f : m\_f \in trans\_file]$.

*Definition 3 (If Transaction File Does Not Contain Required Medical Facilities Then Ignored):* For all transaction file $trans\_file$ within a limited time, there are some ignoring medical facilities. $Adv_{disease,p\_trans\_f}(l) \leq SHS\_id$ exists, so the probability $report[disease(report, test) = m\_f : m\_f \in p\_trans\_f]$ is ignored.

### C. SECURITY MODEL OF THE PROPOSED FRAMEWORK

In the proposed framework, the RSHealth considers attackers on the inter-network layer $IL\_A$, which have access to the public keys of $n$ SHSs and the private keys of some SHSs.

**Gamer 1: the anti-tampering of RSHealth**

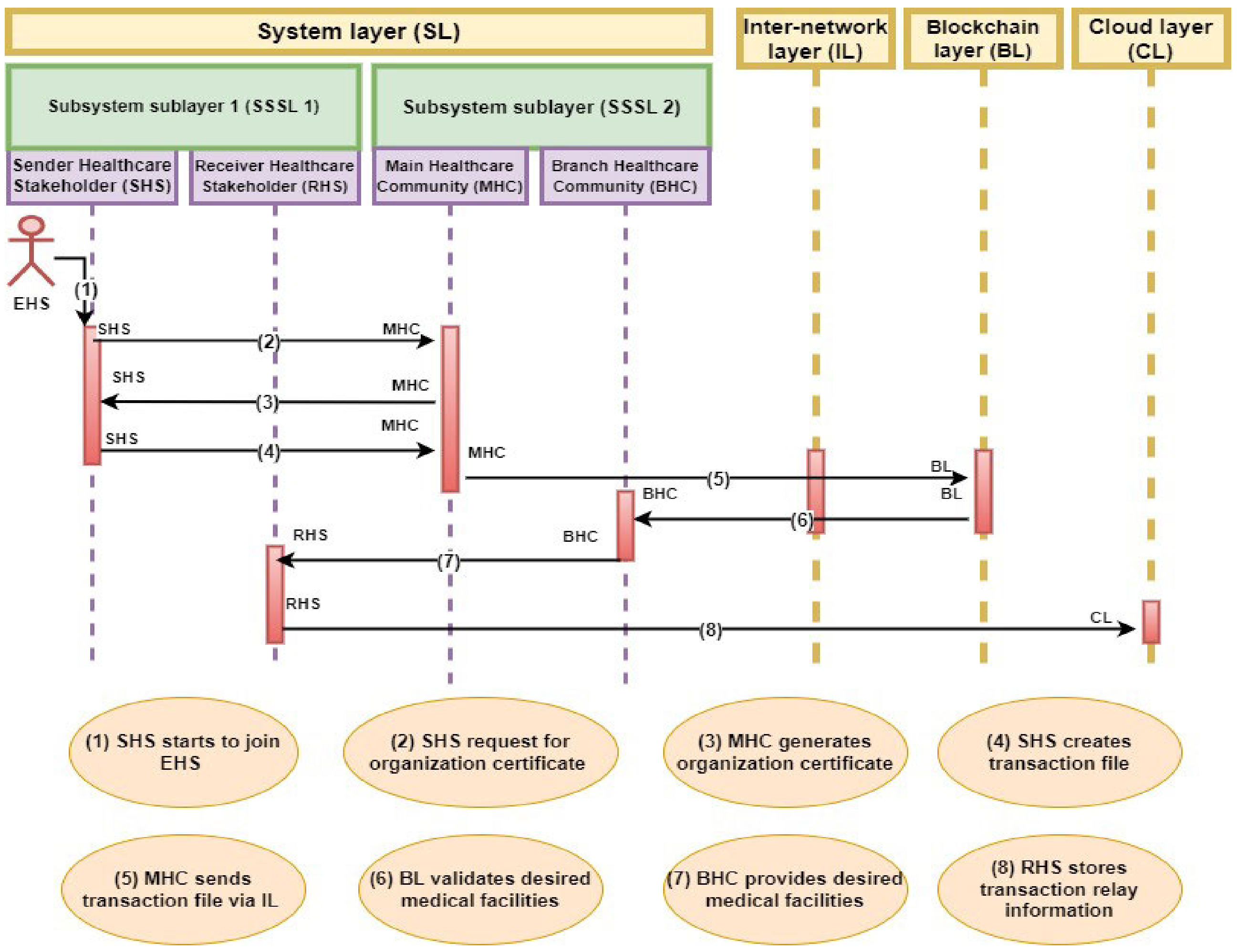

System layer (SL)
Inter-network layer (IL)
Blockchain layer (BL)
Cloud layer (CL)
Subsystem sublayer 1 (SSSL 1)
Subsystem sublayer (SSSL 2)
Sender Healthcare Stakeholder (SHS)
Receiver Healthcare Stakeholder (RHS)
Main Healthcare Community (MHC)
Branch Healthcare Community (BHC)
EHS
(1)
SHS
MHC
(2)
(3)
(4)
(5)
BL
BHC
(6)
RHS
(7)
CL
(8)
(1) SHS starts to join EHS
(2) SHS request for organization certificate
(3) MHC generates organization certificate
(4) SHS creates transaction file
(5) MHC sends transaction file via IL
(6) BL validates desired medical facilities
(7) BHC provides desired medical facilities
(8) RHS stores transaction relay information

**FIGURE 3.** Sequence diagram of the proposed framework.

For attacker on the inter-network layer $IL\_A$ and a challenger on the system layer $SL\_C$, success in entering the blockchain-based EHS with RSHealth $S_{RSHealth,d}$ is determined by the probability as given in Game 1.

- **Phase 1:** Once the security parameter $sec\_p$ is set, the challenger on the system layer $SL\_C$ proceeds to complete Phase 1 to acquire the system parameters $sys\_p$. Subsequently, $SL\_C$ forwards these system parameters $sys\_p$ to the attacker situated on the inter-network layer $IL\_A$.
- **Phase 2:** Once the attacker on the inter-network layer $IL\_A$ selects any value, the challenger on the system layer $SL\_C$ supplies the corresponding hash value to $IL\_A$.
- **Phase 3:** The attacker on the inter-network layer $IL\_A$ selects and requests SHS test $test_i$, prompting the challenger on the system layer $SL\_C$ to provide the corresponding test of SHS $test_i$.
- **Phase 4:** The attacker on the inter-network layer $IL\_A$ queries for SHS's disease $disease_i$, and the challenger on the system layer $SL\_C$ provides the corresponding disease of SHS $disease_i$.
- **Phase 5:** The attacker on the inter-network layer $IL\_A$ selects and submits SHS's identity $id*$, prompting the challenger on the system layer $SL\_C$ to provide the corresponding transaction file $p\_trans\_f$ after applying the RSHealth to $IL\_A$.
- **Phase 6:** At the end, the attacker on the inter-network layer $IL\_A$ generates the transaction file $p\_trans\_f$ of pseudo identity $id*$ that satisfies the following two conditions:
  1) $p\_trans\_f$ is a valid transaction file generated by the attacker on the inter-network layer $IL\_A$.
  2) $id*$ does not available in the transaction file.

*Definition 4 (Anti-Tampering):* If the attacker on the inter-network layer $IL\_A$ creates a maximum of $q_T$ test queries, $q_R$ hash queries for report, a maximum of $q_D$ queries related to disease, and prepares a maximum of $q_{TF}$ transaction files within the maximum time $T$, then the probability of success $S_{RSHealth,d}$ is at least single $SHS$, ensuring SHS file $SHS\_file$ $= (T, q_T, q_R, q_D, q_{TF}, SHS)$ remain anti-tampered under adaptive selection message attack.

**Gamer 2: the anonymity of RSHealth**

Assuming $SHS = SHS_1, SHS_2, SHS_3, \ldots, SHS_n$ represents the $n$ SHSs entities. Similar to Game 1, there exists an attacker on the inter-network layer $IL\_A$ and a challenger on the system layer $SL\_C$.

- **Step 1:**The challenger on the system layer $SL\_C$ executes **Phase 1** of **Game 1** to compute system parameters $sys\_p$ and then transmits obtained result to the attacker on the inter-network layer $IL\_A$.
- **Step 2:** The attacker on the inter-network layer $IL\_A$ adaptively makes polynomial limited-order queries $q_T$ for transaction files *trans_file*.
- **Step 3:** The attacker on the inter-network layer $IL\_A$ outputs an identity $id*$, the test list $T_l$ of $n$ SHSs, two different tests antibody *antibody*, and antigen *antigen* belonging to $T_l$, and sends all of them to the challenger on the system layer $SL\_C$. The challenger on the system layer $SL\_C$ randomly selects a symptom $sym$ from any test $test_i$ and returns SHS file *SHS_file* = **TFRing**($id*$, $Tl_i$, $disease_i$) to the attacker on the inter-network layer $IL\_A$.
- **Step 4:** Finally, the attacker on the inter-network layer $IL\_A$ outputs symptom $sym'$ of any test $test_i$.
- **Step 5:** The attacker on the inter-network layer $IL\_A$ succeeds in this game if and only if $sym = sym'$.

*Definition 5 (Identity Anonymity):* The probability of the attacker on the inter-network layer $IL\_A$ succeeding in Game 2 is given by $Success(IL\_A) = P_r[sym = sym'] = 1/2 + SHS$. The precondition for a SHS's transaction file to have full anonymity is that no attacker can win Game 2 with a non-negligible probability. This implies that for any polynomial-time attacker, the advantage of winning Game 2 to discover a SHS's identity is negligible.

## D. TRANSACTION FILE PRIVACY PROTECTION BASED ON RING SIGNATURE

This section primarily discusses the utilization of RSHealth technology to devise a fully anonymous transaction file storage within the blockchain-based EHS, ensuring the confidentiality of SHS's identity during transmission from MHC to BHC. A smart contract is deployed in the BL to oversee the activities in the layer, and upon meeting predefined conditions, a preset instruction is activated to execute transaction file *trans_file*. The specific function, based on Algorithm 1, is constructed as follows:

1) **TFSetup**($sec\_p \rightarrow sys\_p$): This function is implemented at SSSL1. SHS takes a security parameter $sec\_p$, which is a large prime number, as input. SHS then randomly checks the x-ray report $xray\_r$ to ensure it is greater than least security parameter $sec\_p$ provided by EHS. Next, EHS identifies the disease *disease* (AIDS *aids*) of every SHS in the queue $Q = \langle SHS\_file^*, \times \rangle$ such that $1 \leq aids \leq xray\_r$. Additionally, it collects the report *report* of SHS that are primitive roots in the queue $Q = \langle SHS\_file^*, \times \rangle$. Finally, it produces the system parameters $sys\_p = (xray\_r, aids, Q)$ as output.
2) **TFKeygen** ($sys\_p \rightarrow [test_i, disease_i]$): Each SHS $SHS_i$ $(1 \leq i \leq n)$ of MHC in the system layer applies the nucleic acid test *nat* based on the CT-scan *ct_scan*, disease *aids*, and x-ray report *xray_r*, following the (1):
$$nat = ct\_scan^{aids} \% xray\_r. \tag{1}$$
The public key of an SHS is defined as
$(ct\_scan, nat, xray\_r) \in test_i$
and its private key is $aids \in disease_i$.
3) **TFRing**($[id_i, test_i, disease_i] \rightarrow [SHS\_id_i, Tl_i]$): Here, if the transaction initiator $SHS\_k$ performs a random urine test $urine\_t$ in the queue $Q = \langle SHS\_file^*, \times \rangle$, then MHC calculates (2) and (3) as follows:
$$H_0(i) = H(SHS\_id_i) \tag{2}$$
$$H_1(i) = [id_i - aids_i \times H_0(i)] \times urine\_t_i^{-1} \% \times (xray\_r_i - 1) \tag{3}$$
$$SHS\_id_i = (ct\_scan_i^{urine\_t_i}) \% (xray\_r_i) \tag{4}$$
$$Tl_i = id_i \times (nat_i^{urine\_t_i}) \% (xray\_r) \tag{5}$$
The original identity *id* of $SHS_k$ is $id_k$. SHS's identity is anonymized by MHC, and the output contains transaction file *trans_file* as:
$$(H_0(i), H_1(i), SHS\_id_1, SHS\_id_2, SHS\_id_3, \ldots SHS\_id_k, \ldots SHS\_id_n, Tl_1, Tl_2, Tl_3, \ldots Tl_k \ldots Tl_n).$$
4) **TFVerify**($[V_1i, V_2i] \rightarrow [true, false]$): Using(4) and (5), the transaction file is calculated by MHC. Then, MHC sent this transaction file *trans_file* to BL via IL for verification, as mentioned in (6), and (7).
$$\left\{ \begin{array}{l} V_{1i} = nat_i^{H_0(i)} \times H_0(i)^{H_1(i)} \% (xray\_r_i) \\ V_{2i} = (ct\_scan_i^{id_i}) \% (xray\_r_i) \end{array} \right\} \tag{6}$$
$$\sum_{i=1}^{n} V_{1i} = \sum_{i=1}^{n} V_{2i} \tag{7}$$
To calculate $V_{1i}$ and $V_{2i}$ using (6), BL first need to determine the values of $V_{1i}$ and $V_{2i}$ based on the required medical facilities i.e, ct-scan *ct_scan* and available medical facility in DPL i.e, nucleic acid test *nat*. Then, BL can verify whether (7) is true. If (7) is true, it implies that (4) and (5) have not been used by the attacker on the inter-network layer $IL\_A$, indicating that the transaction file is valid. Conversely, if (7) is false, the transaction file is not valid.
5) **TFOpen**($[SHS\_id_i, Tl_i] \rightarrow id_i$): After proper verification by BL, the valid transaction relay signal is transferred to BHC. BHC then provides the necessary medical facilities as per the requirement. Finally, valid transaction signal is sent to RHS. The original

credentials of SHSs are again obtained by RHS, as given in (8).

$$id_i = Tl_i \times [(SHS\_id_i)^{aids_i}]^{-1}\%(xray\_r_i) \tag{8}$$

**Algorithm 1** Proposed RSHealth

```
/* Intial setup using TFSetup()   */
Input: $sec\_p$
Output: $sys\_p$
/* Key generation using TFKeygen()  */
1  for $i$=1 to $n$ do
2      $nat = ct\_scan^{aids}\%xray\_r$
3      $SHS_i$ = TFKeygen ($sys\_p \rightarrow [test_i, disease_i]$)
/* Transaction creation using TFRing()   */
4  for $i$=1 to $n$ do
5      Perform random urine test $urine\_t$ for transaction
         file $T\_file$ and then calculate:
       /* Hash calculation   */
6      $H_0(i) = H(SHS\_id_i)$
7      $SHSID_i$= $(ct\_scan_i^{urine\_t_i})\%(xray\_r_i)$
8      $Tl_i$= $id_i \times (nat_i^{urine\_t_i})\%(xray\_r)$
/* Transaction verification using TFVerify()   */
9  for $i$=1 to $n$ do
10     $V_1 i = nat_i^{H_0(i)} \times H_0(i)^{H_1(i)}\%(xray\_r_i)$
11     $V_2 i = (ct\_scan_i^{id_i})\%(xray\_r_i)$
12     $\sum_{i=1}^{n} V_1 i = \sum_{i=1}^{n} V_2 i$
/* Transaction opened by RHSs using TFOpen()   */
13 for $i$=1 to $n$ do
14     $id_i = Tl_i \times [(SHS\_id_i)^{aids_i}]^{-1}\%(xray\_r_i)$
```

## IV. RESULTS ANALYSIS

In this section, the proposed RSHealth algorithm is evaluated in context of security analysis, execution analysis and performance analysis.

### A. SECURITY ANALYSIS

A blockchain-based EHS with organization-level privacy protection relies on several security mechanisms, including blockchain technology, public key cryptosystem, and RSHealth. By integrating these security mechanisms into the design of a blockchain-based EHS establishes a robust framework for managing privacy of SHS's identity and security of transaction file. The proposed RSHealth guarantees accuracy, anonymity and anti-tampering features.

*Lemma 1 (Assurance of Accuracy):*

*Proof:* RHS receives $V_1 i$, $V_2 i$ and uses identity $id$ to verify the transaction list $Tl_i$ according to (7), (9), (10), (1), (2), (4), and (11). If it is true, then the RSHealth assures accuracy.

$$\sum_{i=1}^{n} V_1 i = \sum_{i=1}^{n} V_2 i$$

If $0 < V_1 i < xray\_r_i$ and $0 < V_2 i < (xray\_r_i - 1)$ exists then calculate:

$$V_1 i = ct\_scan_i^{id_i}\%(xray\_r_i) \tag{9}$$
$$V_2 i = (nat_i^{H_0(i)}) \times (H_0(i)^{H_1(i)})\%(xray\_r_i) \tag{10}$$

If $V_1 i$ is congruent to $V_2 i$, the transaction file is accepted by RHS; otherwise, it is rejected.

$$\begin{aligned}
nat_i &= ct\_scan_i^{aids_i}\%(xray\_r_i)\\
H_0(i) &= H(SHS\_id_i)\\
SHSID_i &= (ct\_scan_i^{urine\_t_i})\%(xray\_r_i)\\
\sum_{i=1}^{n} V_2 i &= nat_i^{H_0(i)} \times H_0(i)^{H_1(i)}\%(xray\_r_i)\\
&= ct\_scan_i^{aids_i \times SHSID_i} \times SHS\_id_i^{H_1(i)}\%(xray\_r_i)\\
&= ct\_scan_i^{aids_i \times SHSID_i} \times ct\_scan_i^{urine\_t_i \times H_1(i)}\%\\
&\quad (xray\_r_i)\\
&= ct\_scan_i^{aids_i \times SHSID_i + urine\_t_i \times H_1(i)}\%(xray\_r_i)\\
&\quad ct\_scan_i^{id_i}\%(xray\_r_i)\\
&\equiv \sum_{i=1}^{n} V_1 i
\end{aligned} \tag{11}$$

Here, $ct\_scan_i$ is available in DPL. Hence,

$$id_i = aids_i \times SHS\_id_i + urine\_t_i \times H_1(i)$$

or,

$$H_1(i) = [id_i - aids_i \times H_0(i)] \times urine\_t_i^{-1}\%(xray\_r_i - 1)$$

■

*Lemma 2 (Assurance of Anti-Tampering):* In Game 1, the attacker situated on the inter-network layer $IL\_A$ has the capability to dynamically choose an identity for launching an attack. Additionally, it is assumed that anti-tampering measures are successfully implemented within a reasonable time frame, with a probability of significant success.

*Proof:* If the challenger situated on the system layer $SL\_C$ receives any report, then the challenger's response to the attacker's query $q_T$ is as follows:

- **Phase 1:** With a security parameter $sec\_p$, the challenger operating on the system layer $SL\_C$ initiates Phase 1 to acquire the system parameters $sys\_p$. Subsequently, the challenger on $SL\_C$ transmits these system parameters $sys\_p$ to the attacker situated on the inter-network layer $IL\_A$.
- **Phase 2:** When the attacker positioned on the inter-network layer $IL\_A$ requests $H_0(i)$, the challenger situated on the system layer $SL\_C$ furnishes the corresponding hash value for $H_0(i)$.

- **Phase 3:** When the attacker operating on the inter-network layer $IL\_A$ requests SHS test $test_i$, the challenger located on the system layer $SL\_C$ supplies the corresponding SHS test $test_i$.
- **Phase 4:** If the attacker on the inter-network layer $IL\_A$ queries for the disease $disease_i$ of SHS, the challenger on the system layer $SL\_C$ ceases operation. Otherwise, the challenger furnishes the corresponding disease of SHS, denoted as $disease_i$.
- **Phase 5:** When the attacker situated on the inter-network layer $IL\_A$ selects and submits the original identity $id*$ of SHS, the challenger on the system layer $SL\_C$ supplies the corresponding transaction file $p\_trans\_f$ after applying Algorithm 1 as specified in (1), (2), (3), (4) and (5).

$$\begin{aligned} nat &= ct\_scan^{aids}\%xray\_r.\\ H_0(i) &= H(SHS\_id_i)\\ H_1(i) &= [id_i{-}aids_i \times H_0(i)]\\ &\quad \times urine\_t_i^{-1}\%(xray\_r_i - 1)\\ SHS\_id_i &= (ct\_scan_i^{urine\_t_i})\%(xray\_r_i)\\ Tl_i &= id_i \times (nat_i^{urine\_t_i})\%(xray\_r) \end{aligned}$$

Ultimately, the original identity $id^*$ is present in the transaction file

$$\begin{aligned} &p\_trans\_f\\ &\quad = (id_i*, SHS\_id_1, SHS\_id_2, \ldots, SHS\_id_n,\\ &\qquad Tl_1, Tl_2, \ldots., Tl_n) \end{aligned}$$

- **Phase 6:** In conclusion, the attacker on the inter-network layer $IL\_A$ generates the transaction file $p\_trans\_f$ for the pseudo identity $id*$. Thus, both the generated transaction files $trans\_file$ and $p\_trans\_f$ are considered valid.
  1) $trans\_file = (id_i,$ $SHS\_id_1, SHS\_id_2, \ldots, SHS\_id_n,$ $Tl_1, Tl_2, \ldots., Tl_n)$
  2) $p\_trans\_f = (id_i*,$ $SHS\_id_1, SHS\_id_2, \ldots, SHS\_id_n,$ $Tl_1, Tl_2, \ldots., Tl_n)$

  Therefore, in Game 1, the attacker situated on the inter-network layer $IL\_A$ can indeed fabricate the $p\_trans\_f$. However, assuming that anti-tampering measures are effectively implemented within a reasonable time frame with a substantial probability of success, it becomes challenging for the attacker to tamper the $trans\_file$. Even if the attacker randomly selects test such as urine test $urine\_t$ to falsify $nat$, $H\_0(i)$, $H\_1(i)$, $SHS\_id_i$, and $Tl_i$, the private key aids $aids_i$ of SHS is required for these calculations. Without knowing this private key, it is infeasible for the attacker to create $id_i$, rendering the forging of the $trans\_file$ impossible. This indicates that the RSHealth effectively mitigates tampering attempts. ■

*Lemma 3 (Assurance of SHS's Identity Anonymity):* The RSHealth ensures identity anonymity for all SHSs, guaranteeing that all reports have an equal probability of $P_r[sym = sym'] = 1/2$. This means that regardless of which SHS generates the transaction file, the likelihood of anonymity being preserved is consistent.

$$\begin{aligned} trans\_file = (id_i,&\\ &SHS\_id_1, SHS\_id_2, \ldots, SHS\_id_n,\\ &Tl_1, Tl_2, \ldots., Tl_n). \end{aligned}$$

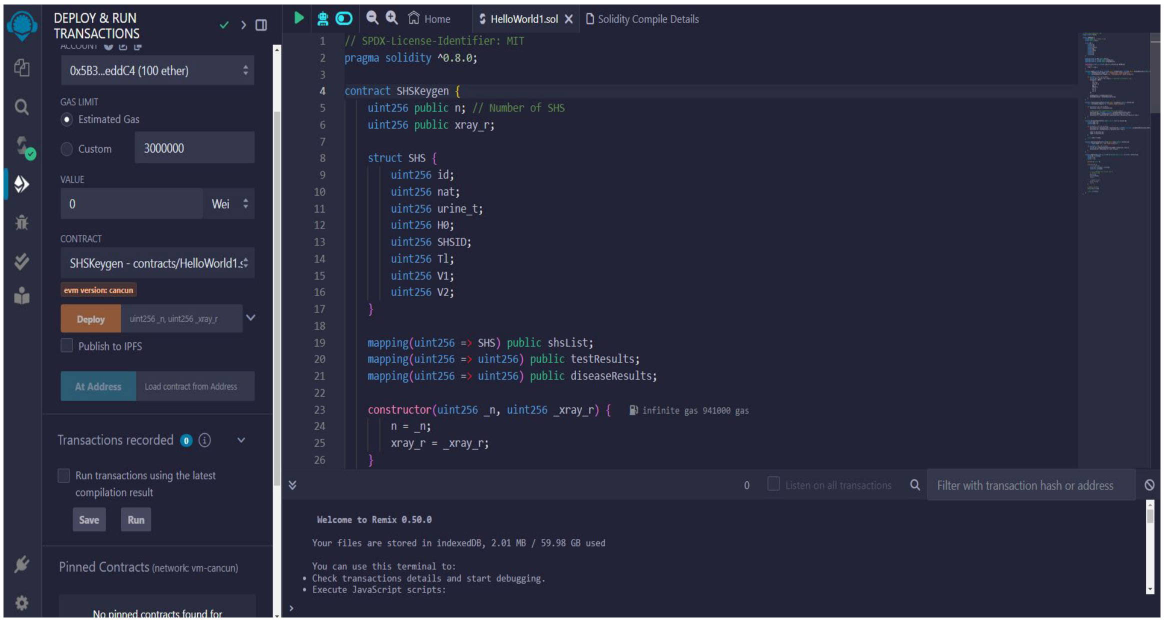
*Proof:* In a manner similar to Game 1, both the attacker on the inter-network layer $IL\_A$ and the challenger on the system layer $SL\_C$ engage in Game 2.

- **Step 1:** The challenger on the system layer $SL\_C$ initiates Phase 1 of Game 1 to compute the system parameters $sys\_p$ and transmits it to the attacker situated on the inter-network layer $IL\_A$.
- **Step 2:** The attacker positioned on the inter-network layer $IL\_A$ adaptively issues polynomial-limited-order queries $q_T$ for transaction files.
- **Step 3:** The attacker on the inter-network layer $IL\_A$ provides an identity $id*$, a test list $T_l$ comprising tests for $n$ SHSs, and two distinct tests $antibody$ and $antigen$ both belonging to $T_l$. This information is then transmitted to the challenger on the system layer $SL\_C$. The challenger randomly selects a symptom $sym$ from any test $test_i$ and responds by sending the transaction file **TFRing**$(id, Tl_i, disease_i)$ to the attacker on the inter-network layer $IL\_A$.
- **Step 4:** At the end, the attacker situated on the inter-network layer $IL\_A$ outputs the symptom $sym'$ corresponding to any test $test_i$.
- **Step 5:** The attacker on the inter-network layer $IL\_A$ achieves success in this game if and only if $sym = sym'$.

Prior to any SHS or RHS actively disclosing all the information contained within the transaction file, the original identity $id$ of the created transaction file remains concealed from any third party. In the key generation TFKeygen(), and TFOpen() functions, knowledge of the secret disease $aids$ is imperative. In the signature TFRing(), and verification TFVerify() functions, the values $H_0(i)$ and $H_1(i)$ are based on $SHS\_id_i$. Both $SHS\_id_i$ and $Tl_i$ are derived by randomly selecting $urine\_t_i$. The likelihood of an outsider of the blockchain-based EHS member correctly guessing the original identity of an SHS $id$ suffering from AIDS or another disease does not exceed $1/(n+1)$, while the probability for an insider member of the blockchain-based EHS does not exceed $1/n$. Consequently, RSHealth ensures identity anonymity. ■

## B. EXPERIMENTAL ANALYSIS

A system comprises an Intel(R) Core(TM) i5-8250u CPU @ 1.60 GHz with 8GB RAM and a 64-bit operating system, x-64 based processor is utilized for implementing the proposed RSHealth data privacy framework. The framework



ACCOUNT: 0x5B3...eddC4 (100 ether)
GAS LIMIT: Estimated Gas [✓]; Custom 3000000
VALUE: 0 Wei
CONTRACT: SHSKeygen - contracts/HelloWorld1.s
evm version: cancun
Deploy: uint256 _n, uint256 _xray_r
Publish to IPFS
At Address: Load contract from Address
Transactions recorded 0
Run transactions using the latest compilation result
Pinned Contracts (network: vm-cancun)
No pinned contracts found for

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SHSKeygen {
    uint256 public n; // Number of SHS
    uint256 public xray_r;

    struct SHS {
        uint256 id;
        uint256 nat;
        uint256 urine_t;
        uint256 H0;
        uint256 SHSID;
        uint256 Tl;
        uint256 V1;
        uint256 V2;
    }

    mapping(uint256 => SHS) public shsList;
    mapping(uint256 => uint256) public testResults;
    mapping(uint256 => uint256) public diseaseResults;

    constructor(uint256 _n, uint256 _xray_r) {    infinite gas 941000 gas
        n = _n;
        xray_r = _xray_r;
    }
```

0 Listen on all transactions  Filter with transaction hash or address

```
Welcome to Remix 0.50.0

Your files are stored in indexedDB, 2.01 MB / 59.98 GB used

You can use this terminal to:
• Check transactions details and start debugging.
• Execute JavaScript scripts:
>
```

**FIGURE 4.** Execution of RSHealth.

Algorithm 1 is executed step-by-step using the Remix IDE open source web and desktop application as demonstrated in Figure 4.

**TABLE 3.** Computation time, transaction fee, and transaction file of SHS.

| Algorithms | Time(s) | Transaction_Fee (ETH) | Transaction file (KB) |
|---|---|---|---|
| TFSetup() | 0.00503 | 0.00647853753848344 | 1808 |
| **TFKeygen()** | 0.00603 | 0.00647897154924584 | 1603 |
| TFRing() | 0.00945 | 0.00634729249825985 | 4061 |
| TFVerify() | 0.00753 | 0.00264783587424182 | 5427 |
| TFOpen() | 0.00802 | 0.00647753753848343 | 3205 |
| RSHealth | 0.014785 | 0.947859523641245625 | 6808 |

**TABLE 4.** Normalization on computation time and transaction file.

| Algorithms | Time (msec) | Transaction_Fee (ETH) | Transaction file (MB) |
|---|---|---|---|
| TFSetup() | 50 | 0.00647853753848344 | 2 |
| TFKeygen() | 60 | 0.00647897154924584 | 2 |
| TFRing() | 90 | 0.00634729249825985 | 4 |
| TFVerify() | 70 | 0.00264783587424182 | 5 |
| TFOpen() | 80 | 0.00647753753848343 | 3 |
| RSHealth | 100 | 0.947859523641245625 | 6 |

After deployment, computation time, transaction fee and size of transaction required for every functions of RSHealth TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() along with RSHealth itself is listed in Table 3. Moreover, the normalized computation time, and transaction size are represented in Table 4. RSHealth is employed in a manner that facilitates its throughput or performance by estimating gas consumption during deployment of each function of RSHealth considering TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() by varying number of SHSs $SHS_i$ and sizes of transaction.

For instance, with 100 SHSs, the RSHealth requires less computation time for TFSetup() (50msec), TFKeygen() (60msec), TFRing() (90 msec), TFVerify() (70msec),

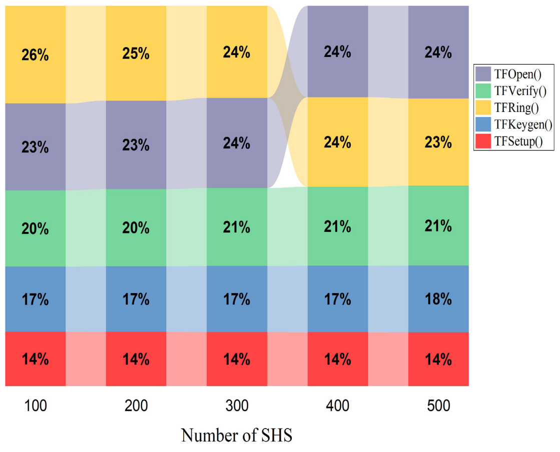


**FIGURE 5.** Gas consumption in each function of RSHealth.

and TFOpen()(80m sec). Likewise with 200 SHSs, the RSHealth gains less computation time for TFSetup() (52msec), TFKeygen() (63msec), TFRing() (91m sec), TFVerify() (74msec), and TFOpen()(85 msec). Simultaneously, with 300 SHSs, the RSHealth requires less computation time for TFSetup() (54msec), TFKeygen() (66msec), TFRing() (92msec), TFVerify() (78msec), and TFOpen()(90 msec). Likewise, with 400 SHSs, the RSHealth requires less computation time for TFSetup() (56msec), TFKeygen() (69msec), TFRing() (93msec), TFVerify() (82msec), and TFOpen()(95 msec). Finally, with 500 SHSs, the RSHealth reaches computation time for TFSetup() (58msec), TFKeygen() (72msec), TFRing() (94msec), TFVerify() (86msec), and TFOpen()(100 msec).

In evaluating the RSHealth in terms of TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen(), Figure 5

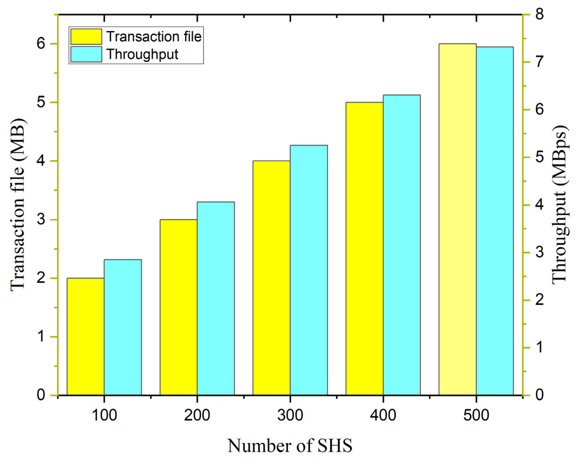


**FIGURE 6.** Throughput with increasing no. of SHS and transaction file.

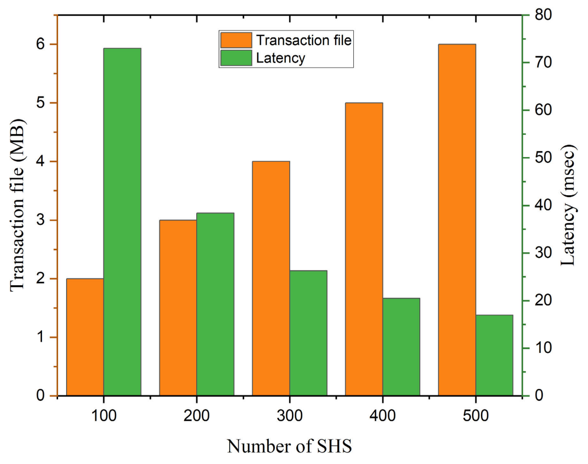


**FIGURE 7.** Latency with increasing no. of SHS and transaction file.

presents a layered graphical representation of the test results. This provides robust framework under the supervision of relevant organization, ensuring identity anonymity and anti-tampering measures are satisfied.

**TABLE 5.** Throughput and latency of RSHealth.

| S.N. | no. of SHS | Transaction file (MB) | Throughput (MBps) | latency (msec) |
|---|---|---|---|---|
| 1. | 100 | 2 | 2.85 | 73.00 |
| 2. | 200 | 3 | 4.06 | 38.39 |
| 3. | 300 | 4 | 5.25 | 26.28 |
| 4. | 400 | 5 | 6.31 | 20.47 |
| 5. | 500 | 6 | 7.32 | 16.95 |

## C. PERFORMANCE ANALYSIS

The performance of the proposed RSHealth is evaluated utilizing five functions: TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() with varying sizes of transaction file *trans_file* from 2MB to 6MB (2 MB, 3 MB, 4 MB, 5 MB, and 6 MB). The throughput and latency is calculated using (12), and (13) respectively.

$$Throughput = \sum_{i=1}^{n}(Gas \div Time) * trans\_file_i \tag{12}$$

$$Latency = \sum_{i=1}^{n}(Gas * Time) \div SHS_i \tag{13}$$

**TABLE 6.** Comparison of RSHealth with existing works [15] and [28].

| No. of SHS | Transaction file (MB) | RSHealth (MBps) | Existing scheme1 [15] | Existing scheme2 [28] |
|---|---|---|---|---|
| 100 | 2 | 2.85 | 1.23 | 1.10 |
| 200 | 3 | 4.06 | 2.10 | 2.20 |
| 300 | 4 | 5.25 | 2.40 | 2.30 |
| 400 | 5 | 6.31 | 3.21 | 3.05 |
| 500 | 6 | 7.32 | 3.35 | 3.48 |

As for example, if number of SHS is 100, size of transaction file is 2MB, gas consumption (as seen in Figure 5) and time spend by every functions of RSHealth TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() are

**TABLE 7.** Existing works with RSHealth.

| Paper | EHS | signature scheme | layer security | annonymity | anti-tampering |
|---|---|---|---|---|---|
| kumar et al. [31] | yes | no | yes | no | no |
| hara et al. [19] | no | ring | no | yes | no |
| li et al. [8] | no | ring | no | yes | yes |
| prabha et al. [22] | yes | no | yes | no | no |
| mamta et al. [4] | yes | no | yes | no | no |
| Existing scheme 1 [15] | no | group | no | yes | yes |
| Existing scheme 2 [28] | no | ring | no | yes | yes |
| RSHealth | yes | ring | yes | yes | yes |

(14%, 50msec), (17%, 60msec), (26%, 90msec), (20%, 70msec), and (23%, 80msec) respectively then throughput is 2.85 MBps and latency is 73 msec using (12), and (13) respectively. Similarly, if number of SHS is 200, size of transaction file is 3MB, gas consumption (as seen in Figure 5) and time spend by every functions of RSHealth TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() are (14%, 52msec), (17%, 63msec), (25%, 91msec), (20%, 74msec), and (25%, 85msec) respectively then throughput is 4.06MBps and latency is 38.39 msec using (12), and (13) respectively. Consequently, if number of SHS is 300, size of transaction file is 4MB, gas consumption (as seen in Figure 5) and time spend by every functions of RSHealth TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() are (14%, 54msec), (17%, 66msec), (24%, 92msec), (21%, 78msec), and (24%, 90msec) respectively then throughput is 5.25 MBps and latency is 26.28 msec using (12), and (13) respectively. Similarly, if number of SHS is 400, size of transaction file is 5MB, gas consumption (as seen in Figure 5) and time spend by every functions of RSHealth TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() are (14%, 56msec), (17%, 69msec), (24%, 93msec), (21%, 82msec), and (24%, 95msec) respectively then throughput is 6.32 MBps and latency is 20.47 msec using (12), and (13) respectively. Finally, if number of SHS is 500, size of transaction file is 6MB, gas consumption (as seen in Figure 5) and time spend by every functions of RSHealth TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen() are (14%, 58msec), (18%, 72msec), (23%, 94msec), (21%, 86msec), and (24%, 100msec) respectively then throughput is 7.32 MBps and latency is 16.95 msec using (12), and (13) respectively. The tested results are shown in Table 5.

The proposed method RSHealth obtains high throughput as the number of SHS, and size of transaction file increases as shown in Figure 6. Also, it provides low latency when increasing the number of SHS, and transaction files as shown in Figure 7.

The comparative results provided in Table 6. The tested results as shown in Table 6 works as follows: while observing the overall framework performance in terms of throughput, RSHealth having superior outcomes resulting in Existing scheme 1 [15], and Existing scheme 2 [28] are shown in Figure 8.

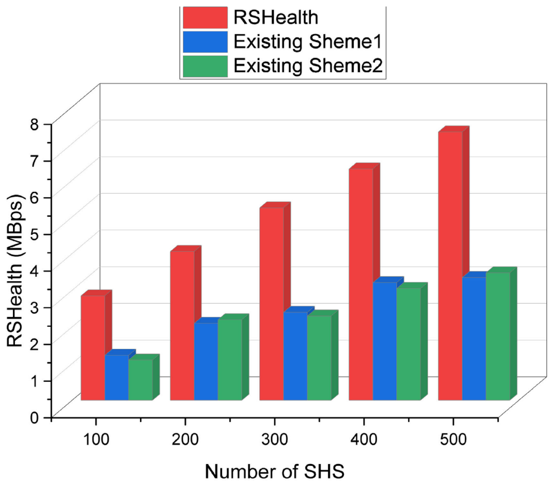


**FIGURE 8.** **Performance of RSHealth with existing schemes.**

Despite a substantial increase in the number of SHS from 100 to 500, computation time of RSHealth remain modest due to the employment of efficient cryptosystem. Additionally, Table 7 shows a uniqueness of the proposed framework in comparison with existing works.

## V. CONCLUSION AND FUTURE WORK

The present study focuses on the security and privacy of sensitive healthcare transaction files in a blockchain-based EHS. Specifically, it addresses the challenges of ensuring security and privacy when transmitting transaction files from MHC to BHC in the lack of necessary medical facilities at MHC. The RSHealth framework has five functions: TFSetup(), TFKeygen(), TFRing(), TFVerify(), and TFOpen(). A comprehensive security analysis confirms that RSHealth is highly effective in ensuring both anti-tampering and identity anonymity for SHS. The performance research shows that there is a higher rate of data transfer and a lower delay in processing as the number of SHS and the size of each transaction rise. One significant drawback of this study is the omission of IoT device authentication provided on IL. In order to successfully utilize the blockchain-based EHS, it is imperative to do additional research and apply the suggested framework to authenticate devices for the IL. These efforts are essential for improving and deploying the framework in blockchain-based EHS.

## DECLARATION

- Ethical approval: Not applicable.
- Competing interests: The authors declare that they have no known competing financial interests or personal relationship that could have appeared to influence the work reported in this paper.
- Authors' contributions: The authors confirm contribution to the paper as follows: Punam Prabha: Conceptualization, Data curation, Formal analysis, Writing-original draft, Writing-review & editing, Validation. Kakali Chatterjee: Writing-review & editing, Validation.
- Funding: Not applicable.
- Availability of data and materials: Not applicable.

## APPENDIX

### *A. SMART CONTRACT CODE IN SOLIDITY OF RSHEALTH*

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract RSHealthComputation {
    uint256 public transactionFileSize = 2 * 1024 * 1024; // 2MB in bytes

    struct Computation {
        uint256 startTime;
        uint256 endTime;
        uint256 gasUsed;
    }

    mapping(string => Computation) public computations;

    event ComputationStarted(string functionName, uint256 startTime);
    event ComputationEnded(string functionName, uint256 endTime, uint256 gasUsed);

    modifier trackComputation(string memory functionName) {
        uint256 startGas = gasleft();
        uint256 startTime = block.timestamp;
        emit ComputationStarted(functionName, startTime);

        _;
        uint256 endTime = block.timestamp;
        uint256 gasUsed = startGas - gasleft();
        computations[functionName] = Computation(startTime, endTime, gasUsed);
        emit ComputationEnded(functionName, endTime, gasUsed);
    }

    function keygen(uint256 sys_p, uint256[] memory testResults, uint256[] memory diseaseResults)
    public trackComputation("Keygen") {
        uint256 n = testResults.length;
        uint256 xray_r = 123456; // Arbitrary value for example

        for (uint256 i = 0; i < n; i++) {
            uint256 nat = (sys_p * (i + 1)) % xray_r;
            // Perform Keygen computations...
        }
    }

    function createTransactionFile(uint256[] memory urineTests)
    public trackComputation("CreateTransactionFile") {
        uint256 n = urineTests.length;
        uint256 xray_r = 123456; // Arbitrary value for example

        for (uint256 i = 0; i < n; i++) {
            uint256 urine_t = urineTests[i];
            // Perform transaction file creation computations...
        }
    }

    function verifyTransactionFile(uint256[] memory testResults)
    public  trackComputation("VerifyTransactionFile") returns (bool) {
        uint256 n = testResults.length;
        uint256 xray_r = 123456; // Arbitrary value for example

        uint256 sumV1 = 0;
        uint256 sumV2 = 0;

        for (uint256 i = 0; i < n; i++) {
            uint256 V1 = (testResults[i] * (i + 1)) % xray_r;
            uint256 V2 = (testResults[i] * (i + 1)) % xray_r;

            sumV1 += V1;
            sumV2 += V2;
        }

        return sumV1 == sumV2;
    }

    function openTransactionFile(uint256[] memory aids)
    public trackComputation("OpenTransactionFile") {
        uint256 n = aids.length;
        uint256 xray_r = 123456; // Arbitrary value for example

        for (uint256 i = 0; i < n; i++) {
            uint256 inv = modInverse((aids[i] * (i + 1)), xray_r);
            // Perform opening transaction file computations...
        }
    }

    function modInverse(uint256 a, uint256 m) internal pure returns (uint256) {
        int256 m0 = int256(m);
        int256 y = 0;
        int256 x = 1;

        if (m == 1) return 0;

        while (a > 1) {
            int256 q = int256(a) / int256(m);
            int256 t = int256(m);
            m = a % m;
            a = uint256(t);
            t = y;

            y = x - q * y;
            x = t;
        }

        if (x < 0) x += m0;

        return uint256(x);
    }
}
```

### B. SMART CONTRACT CODE IN SOLIDITY OF EXISTING SCHEME 1

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/math/SafeMath.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract RSHealthSystem is Ownable {
    using SafeMath for uint256;

    // Define system parameters
    struct SystemParams {
        uint256 p;
        uint256 g1;
        uint256 g2;
    }

    SystemParams public params;

    constructor(uint256 _p, uint256 _g1, uint256 _g2) {
        params.p = _p;
        params.g1 = _g1;
        params.g2 = _g2;
    }

    // Define keys and certificates
    struct RegManagerKey {
        uint256 x;
        uint256 y;
        uint256 u;
        uint256 v;
    }

    struct OrgKey {
        uint256 h1;
        uint256 u1;
        uint256 v1;
        uint256 w1;
        uint256 nu1;
        uint256 n1;
        uint256 g1;
    }

    struct UserKey {
        uint256 h;
        uint256 u;
    }

    struct OrgCertificate {
        uint256 sigma0;
        uint256 r;
    }

    struct UserCertificate {
        uint256 A;
        uint256 a;
    }

    mapping(address => RegManagerKey) public regManagerKeys;
    mapping(address => OrgKey) public orgKeys;
    mapping(address => UserKey) public userKeys;
    mapping(address => OrgCertificate) public orgCertificates;
    mapping(address => UserCertificate) public userCertificates;

    // Helper functions
    function modExp(uint256 base, uint256 exp, uint256 mod) internal pure returns (uint256) {
        uint256 result = 1;
        base = base % mod;
        while (exp > 0) {
            if (exp % 2 == 1) {
                result = (result * base) % mod;
            }
            exp = exp >> 1;
            base = (base * base) % mod;
        }
        return result;
    }

    // Key generation for RegManager
    function regManagerKeyGen(uint256 x, uint256 y) public onlyOwner {
        uint256 u = modExp(params.g2, x, params.p);
        uint256 v = modExp(params.g2, y, params.p);
        regManagerKeys[msg.sender] = RegManagerKey(x, y, u, v);
    }

    // Key generation for Org
    function orgKeyGen(
        uint256 h1,
        uint256 x1,
        uint256 y1,
        uint256 r1,
        uint256 nu1,
        uint256 p1,
        uint256 q1,
        uint256 g1
    ) public {
        require(h1 != 1, "h1␣should␣not␣be␣1");
        uint256 u1 = modExp(h1, x1, params.p);
        uint256 v1 = modExp(h1, y1, params.p);
        uint256 w1 = modExp(params.g2, r1, params.p);
        uint256 n1 = p1.mul(q1);
        uint256 lambda1 = lcm(p1.sub(1), q1.sub(1));
        require(gcd(L(modExp(g1, lambda1, n1.mul(n1)), n1), n1) == 1, "Invalid␣g1");

        orgKeys[msg.sender] = OrgKey(h1, u1, v1, w1, nu1, n1, g1);
    }

    // Key generation for User
    function userKeyGen(uint256 h, uint256 x) public {
        require(h != 1, "h␣should␣not␣be␣1");
        uint256 u = modExp(h, x, params.p);
        userKeys[msg.sender] = UserKey(h, u);
    }

    // Issue Org Certificate
    function issueOrgCert(address orgAddress, uint256 r) public onlyOwner {
        RegManagerKey storage regKey = regManagerKeys[msg.sender];
        OrgKey storage orgKey = orgKeys[orgAddress];
        uint256 sigma0=modExp(params.g1,regKey.x.add(orgKey.nu1).add(regKey.y.mul(orgKey.w1)),params.p);
        orgCertificates[orgAddress] = OrgCertificate(sigma0, r);
    }

    // Issue User Certificate
    function issueUserCert(address userAddress, uint256 a, uint256 r1) public {
        UserKey storage userKey = userKeys[userAddress];
        OrgKey storage orgKey = orgKeys[msg.sender];
        uint256 A = modExp(params.g1.div(userKey.u), r1.add(a), params.p);
        userCertificates[userAddress] = UserCertificate(A, a);
    }

    // Transaction Generation (simplified for example purposes)
    function generateTransaction(address receiver, uint256 amount) public {
        // Implement transaction generation logic
    }

    // Transaction Verification (simplified for example purposes)
    function verifyTransaction(bytes memory transaction) public view returns (bool) {
        // Implement transaction verification logic
        return true;
    }

    // Tracking (simplified for example purposes)
    function trackUser(address userAddress) public view returns (bool) {
        // Implement user tracking logic
        return true;
    }

    // Utility functions
    function lcm(uint256 a, uint256 b) internal pure returns (uint256) {
        return a.mul(b).div(gcd(a, b));
    }

    function gcd(uint256 a, uint256 b) internal pure returns (uint256) {
        while (b != 0) {
            uint256 temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    function L(uint256 x, uint256 n) internal pure returns (uint256) {
        return (x.sub(1)).div(n);
    }
}
```

## C. SMART CONTRACT CODE IN SOLIDITY OF EXISTING SCHEME2

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CARS {
    // Parameters
    uint256 public p;
    uint256 public q;
    uint256 public g;
    uint256 public h;
    address public CA;

    // Events
    event UserRegistered(address indexed user, uint256 publicKey);
    event TransactionSigned(address indexed user, bytes signature);
    event TransactionVerified(bool success);
    event MisbehaviorDetected(address indexed user);

    // Structs
    struct User {
        uint256 publicKey;
        uint256 secretKey;
    }

    struct Signature {
        uint256 q;
        uint256[] rs;
        uint256 a;
        uint256 b;
        uint256 c;
        uint256 h;
    }

    // Mappings
    mapping(address => User) public users;
    mapping(bytes => bool) public transactions;

    // Constructor to initialize the system
    constructor(uint256 _p, uint256 _q, uint256 _g, uint256 _h) {
        p = _p;
        q = _q;
        g = _g;
        h = _h;
        CA = msg.sender;
    }

    // Modifier to restrict access to CA
    modifier onlyCA() {
        require(msg.sender == CA, "Only␣CA␣can␣call␣this␣function");
        _;
    }

    // Key Generation
    function registerUser(address user, uint256 secretKey) public onlyCA {
        uint256 publicKey = modExp(g, secretKey, p);
        users[user] = User(publicKey, secretKey);
        emit UserRegistered(user, publicKey);
    }

    // Signature generation function
    function signTransaction(bytes memory txData, uint256 u, uint256 v, uint256 w) public {
        require(users[msg.sender].publicKey != 0, "User␣not␣registered");

        uint256 a = modExp(h, u, p);
        uint256 b = modExp(a, v, p);
        uint256 c = modExp(a, w, p);
        uint256 hValue = (w + v * uint256(keccak256(abi.encodePacked(txData, a, c)))) % p;

        uint256 txHash = uint256(keccak256(abi.encodePacked(txData, a, b)));
        uint256 R = modExp(g, uint256(keccak256(abi.encodePacked(txData))), p);

        uint256[] memory rs = new uint256[](1);
        rs[0] = uint256(keccak256(abi.encodePacked(txData, txHash, R))) % q;

        uint256 rk=rs[0]-(uint256(keccak256(abi.encodePacked(txData)))*users[msg.sender].secretKey) % q;
        uint256 qValue=(uint256(keccak256(abi.encodePacked(txData))-rk*users[msg.sender].secretKey) % q;

        Signature memory sig = Signature(qValue, rs, a, b, c, hValue);
        transactions[txData] = true;

        emit TransactionSigned(msg.sender, abi.encode(sig));
    }

    // Verification function
    function verifyTransaction(bytes memory txData, bytes memory sigData) public returns (bool) {
        Signature memory sig = abi.decode(sigData, (Signature));

        uint256 txHash = uint256(keccak256(abi.encodePacked(txData, sig.a, sig.b)));
        uint256 R = modExp(g, sig.q, p);
        bool valid = (sig.h == (sig.c * uint256(keccak256(abi.encodePacked(txData, sig.a, sig.c))) % p)
        && (modExp(g, sig.q, p) * modExp(users[msg.sender].publicKey,
        uint256(keccak256(abi.encodePacked(txData))), p)) % p == R);

        emit TransactionVerified(valid);
        return valid;
    }

    // Misbehavior detection function
    function detectMisbehavior(address user, uint256 a, uint256 e, uint256 v) public onlyCA {
        uint256 b = modExp(a, e, p);
        bool misbehavior = (modExp(g, a + v * e, p) == (b * modExp(users[user].publicKey, e, p)) % p);

        if (misbehavior) {
            emit MisbehaviorDetected(user);
        }
    }

    // Modular exponentiation function
    function modExp(uint256 base, uint256 exp, uint256 mod) internal pure returns (uint256) {
        uint256 result = 1;
        while (exp > 0) {
            if (exp % 2 == 1) {
                result = (result * base) % mod;
            }
            base = (base * base) % mod;
            exp /= 2;
        }
        return result;
    }
}
```

## REFERENCES

[1] T. Navaneethakrishnan, R. Manohar, and V. Srinivasan, "Decoding ransomware attacks: Unmasking legal solutions in the IT Act of 2000," *Indian J. Law Legal Res.*, vol. 1, pp. 1–13, 2023.

[2] M. Wazid and P. Gope, "BACKM-EHA: A novel blockchain-enabled security solution for IoMT-based E-healthcare applications," *ACM Trans. Internet Technol.*, vol. 23, no. 3, pp. 1–28, Aug. 2023.

[3] H. Habibzadeh, K. Dinesh, O. R. Shishvan, A. Boggio-Dandry, G. Sharma, and T. Soyata, "A survey of healthcare Internet of Things (HIoT): A clinical perspective," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 53–71, Jan. 2020.

[4] B. Mamta, B. Gupta, K. Li, V. Leung, K. Psannis, and S. Yamaguchi, "Blockchain-assisted secure fine-grained searchable encryption for a cloud-based healthcare cyber-physical system," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 12, pp. 1877–1890, Dec. 2021.

[5] D. Zhang, S. Wang, Y. Zhang, Q. Zhang, and Y. Zhang, "A secure and privacy-preserving medical data sharing via consortium blockchain," *Secur. Commun. Netw.*, vol. 2022, pp. 1–15, May 2022.

[6] G. Verma, "Blockchain-based privacy preservation framework for healthcare data in cloud environment," *J. Experim. Theor. Artif. Intell.*, vol. 36, no. 1, pp. 147–160, Jan. 2024.

[7] R. Gupta, P. Bhattacharya, S. Tanwar, N. Kumar, and S. Zeadally, "GaRuDa: A blockchain-based delivery scheme using drones for healthcare 5.0 applications," *IEEE Internet Things Mag.*, vol. 4, no. 4, pp. 60–66, Dec. 2021.

[8] X. Li, Y. Mei, J. Gong, F. Xiang, and Z. Sun, "A blockchain privacy protection scheme based on ring signature," *IEEE Access*, vol. 8, pp. 76765–76772, 2020.

[9] A. Padma and M. Ramaiah, "GLSBIoT: GWO-based enhancement for lightweight scalable blockchain for IoT with trust based consensus," *Future Gener. Comput. Syst.*, vol. 159, pp. 64–76, Oct. 2024.

[10] A. Padma and R. Mangayarkarasi, "Detecting security breaches on smart contracts through techniques and tools a brief review: Applications and challenges," in *Proc. Int. Conf. Inf. Manage. Eng.* Singapore: Springer, 2022, pp. 361–369.

[11] S. M. Williamson and V. Prybutok, "Balancing privacy and progress: A review of privacy challenges, systemic oversight, and patient perceptions in AI-driven healthcare," *Appl. Sci.*, vol. 14, no. 2, p. 675, Jan. 2024.

[12] G. Zhang, Z. Yang, and W. Liu, "Blockchain-based privacy preserving e-health system for healthcare data in cloud," *Comput. Netw.*, vol. 203, Feb. 2022, Art. no. 108586.

[13] B. S. Egala, A. K. Pradhan, V. Badarla, and S. P. Mohanty, "Fortified-chain: A blockchain-based framework for security and privacy-assured Internet of Medical Things with effective access control," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11717–11731, Jul. 2021.

[14] B. S. Egala, A. K. Pradhan, P. Dey, V. Badarla, and S. P. Mohanty, "Fortified-chain 2.0: Intelligent blockchain for decentralized smart healthcare system," *IEEE Internet Things J.*, vol. 10, no. 14, pp. 12308–12321, Jul. 2023.

[15] H. Zheng, Q. Wu, J. Xie, Z. Guan, B. Qin, and Z. Gu, "An organization-friendly blockchain system," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101598.

[16] J.-S. Lee, C.-J. Chew, J.-Y. Liu, Y.-C. Chen, and K.-Y. Tsai, "Medical blockchain: Data sharing and privacy preserving of EHR based on smart contract," *J. Inf. Secur. Appl.*, vol. 65, Mar. 2022, Art. no. 103117.

[17] L. Wang, G. Zhang, and C. Ma, "A survey of ring signature," *Frontiers Electr. Electron. Eng. China*, vol. 3, no. 1, pp. 10–19, Jan. 2008.

[18] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Cham, Switzerland: Springer, 2001, pp. 552–565.

[19] K. Hara and K. Tanaka, "Ring signature with unconditional anonymity in the plain model," *IEEE Access*, vol. 9, pp. 7762–7774, 2021.

[20] A. V. Meier, "The ElGamal cryptosystem," *Joint Adv. Students Seminar*, Jun. 2005.

[21] R. A. Haraty, A. N. El-Kassar, and B. M. Shebaro, "A comparative study of ElGamal based digital signature algorithms," *J. Comput. Methods Sci. Eng.*, vol. 6, no. s1, pp. S147–S156, Apr. 2007.

[22] P. Prabha and K. Chatterjee, "Design and implementation of hybrid consensus mechanism for IoT based healthcare system security," *Int. J. Inf. Technol.*, vol. 14, no. 3, pp. 1381–1396, May 2022.

[23] S. Shreya, K. Chatterjee, and A. Singh, "A smart secure healthcare monitoring system with Internet of Medical Things," *Comput. Electr. Eng.*, vol. 101, Jul. 2022, Art. no. 107969.

[24] A. Padma and M. Ramaiah, "Blockchain based an efficient and secure privacy preserved framework for smart cities," *IEEE Access*, vol. 12, pp. 21985–22002, 2024.

[25] Y. Bobde, G. Narayanan, M. Jati, R. Raj, I. Cvitić, and D. Peraković, "Enhancing industrial IoT network security through blockchain integration," *Electronics*, vol. 13, no. 4, p. 687, Feb. 2024.

[26] Y. Tang, F. Xia, Q. Ye, M. Wang, R. Mu, and X. Zhang, "Identity-based linkable ring signature on NTRU lattice," *Secur. Commun. Netw.*, vol. 2021, pp. 1–17, Sep. 2021.

[27] C. Gamage, B. Gras, B. Crispo, and A. S. Tanenbaum, "An identity-based ring signature scheme with enhanced privacy," in *Proc. Securecomm Workshops*, Aug. 2006, pp. 1–5.

[28] X. Zhang and C. Ye, "A novel privacy protection of permissioned blockchains with conditionally anonymous ring signature," *Cluster Comput.*, vol. 25, no. 2, pp. 1221–1235, Apr. 2022.

[29] B. D. Deebak, F. Al-Turjman, M. Aloqaily, and O. Alfandi, "An authentic-based privacy preservation protocol for smart e-healthcare systems in IoT," *IEEE Access*, vol. 7, pp. 135632–135649, 2019.

[30] G. Wu, S. Wang, Z. Ning, and J. L. records, "Blockchain-enabled privacy-preserving access control for data publishing and sharing in the Internet of Medical Things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8091–8104, Jun. 2022.

[31] A. Kumar and K. Chatterjee, "A lightweight blockchain-based framework for medical cyber-physical system," *J. Supercomput.*, vol. 79, no. 11, pp. 12013–12041, Jul. 2023.

**PUNAM PRABHA** received the B.Tech. degree in computer science and engineering from Aryabhatta Knowledge University, Patna, Bihar, India, in 2013, and the M.Tech. degree in computer science and engineering with a specialization in information security from Indian Institute of Technology (Indian School of Mines), Dhanbad, India, in 2016. She is currently pursuing the part-time Ph.D. degree in computer science engineering with the National Institute of Technology, Patna, Bihar, India.

Since 2018, she has been an Assistant Professor with the Computer Science and Engineering Department, Department of Science, Technology, and Technical Education, Patna. She has authored one Scopus-indexed journal article and six international conference papers. Her research interests include information security, blockchain technology, and data privacy. She received the best paper awards at "PEEIC-2019" and "FICTA-2022."

**KAKALI CHATTERJEE** (Member, IEEE) received the M.Tech. degree from Guru Gobind Singh Indraprastha University, Delhi, and the Ph.D. degree from Delhi University (formerly Delhi College of Engineering). She is currently an Assistant Professor with the Computer Science and Engineering Department, National Institute of Technology, Patna, India. She has completed a research project at the Centre for Development of Advanced Computing, a Research and Development and Academic Centre of the Government of India. She has published many SCI/SCIE/Scopus indexed international journals of Springer and Elsevier. She has attended and presented research papers in reputed conferences and also published some book chapters in Springer. She is also a Joint Principal Investigator of the ISEA Project (Phase-II), Government of India. She is doing research in healthcare security, biometrics, and malware detection. Her research interests include information security and cryptography.