Multi-User Oriented Data Sharing Scheme for Internet of Medical Things Based on Dual Cryptography Mechanism

Guiping Zheng, Bei Gong, Muhammad Waqas, *Senior Member, IEEE*, Iftekhar Ahmad, *Senior Member, IEEE*, Hisham Alasmary, *Member, IEEE*, and Sheng Chen, *Life Fellow, IEEE*

***Abstract*—Encrypted sharing of Internet of Medical Things (IoMT) data is essential for facilitating collaboration, safeguarding patient privacy, and advancing clinical research. However, existing encryption schemes face numerous challenges in multi-user environments. Traditional proxy re-encryption requires exclusive ciphertext for each user, which is evidently unsuitable for IoMT's multi-user scenarios. Meanwhile, attribute-based encryption provides flexible data access control, but its complex computations and high resource demands limit its use in large-scale IoMT environments. Additionally, challenges like single-point failure and redundant backups emerge in ciphertext storage. To address these challenges, we propose a dual-cryptography mechanism integrating enhanced proxy re-encryption and attribute-based encryption. Our scheme enables unified ciphertext access for authorized users while applying attribute encryption exclusively to small data keys. To mitigate potential data loss from storage server failures, we propose a decentralized ciphertext storage and recovery mechanism with verifiable secret sharing. Furthermore, we implement decentralized ciphertext storage using verifiable secret sharing, ensuring recoverability from server failures. Formal analysis proves confidentiality under the random oracle model. Experimental results demonstrate high security strength, computational efficiency, and robustness. The solution prevents single-point failures, resists collusion attacks, and maintains traceability through blockchain-integrated audit trails.**

Received 19 May 2024; revised 11 August 2025; accepted 5 October 2025. Date of publication 8 October 2025; date of current version 10 December 2025. The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Large Groups Project under Grant RGP.2/637/46. The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Large Groups Project under Grant RGP.2/637/46. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102303 and in part by the National Natural Science Foundation of China under Grant 61971014 and Grant 11675199. Recommended for acceptance by Y. Wang. *(Corresponding author: Muhammad Waqas.)*

Guiping Zheng and Bei Gong are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: zhenggp@emails.bjut.edu.cn; gongbei@bjut.edu.cn).

Muhammad Waqas is with the School of Computing and Mathematical Sciences, Faculty of Engineering and Science, University of Greenwich, SE10 9LS London, U.K., and also with the School of Engineering, Edith Cowan University, Perth 6007 WA, Australia (e-mail: engr.waqas2079@gmail.com).

Iftekhar Ahmad is with the School of Engineering, Edith Cowan University, Perth 6007 WA, Australia (e-mail: i.ahamad@ecu.edu.au).

Hisham Alasmary is with the Department of Computer Science, King Khalid University, Abha 62521, Saudi Arabia (e-mail: alasmary@kku.edu.sa).

Sheng Chen is with the School of Electronics and Computer Science, University of Southampton, SO17 1BJ Southampton, U.K., and also with the Faculty of Information Science and Engineering, Ocean University of China, Qingdao 266100, China (e-mail: sqc@ecs.soton.ac.uk).

Digital Object Identifier 10.1109/TSUSC.2025.3619389

***Index Terms*—Data encryption, data sharing, decentralized storage, Internet of Medical Things, privacy protection.**

I. Introduction

THE Internet of Medical Things (IoMT) is an extension of Internet of Things (IoT) technology applied in medical scenarios [1], [2]. It involves connecting various sensors and medical terminals to collect, transmit, process, and analyze medical data, aiming to enhance the efficiency of medical services and hospital operational management capabilities [5]. In IoMT, patients' personal health data, medical records, etc., are digitized and stored in cloud servers for access and analysis by doctors, patients, and other healthcare practitioners [6], [7]. This model of data sharing brings great convenience to the healthcare field [3], [4]. However, with the popularity of terminal devices and the increase in data volume, IoMT data faces challenges in privacy protection [8], [9]. Medical data contains a large amount of sensitive information, such as personal identity data, financial data, communication data, medical health data, etc. Once leaked or tampered with, it can cause serious harm to patients' privacy and rights. Therefore, ensuring the integrity, confidentiality, and availability of data during the sharing process is crucial [10]. In this context, the encryption process and storage stage become key components of data security sharing, directly affecting the security of data transmission and storage, as well as healthcare information privacy protection and compliance. To ensure the confidentiality and integrity of data during transmission, Raghav et al. [11] and Muthukumaran et al. [12] proposed data security sharing schemes based on proxy re-encryption (PRE) mechanisms. The basic principle is for a third-party proxy to convert data owner's ciphertext into user's ciphertext. However, in IoMT data sharing, which involves numerous doctors, patients, and other healthcare professionals, their schemes require providing exclusive transformed ciphertext for each user [13], [14]. This increases system overhead and also fails to achieve fine-grained access control. To tackle this issue, Das et al. [15] proposed an Ciphertext-policy attribute-based encryption (CP-ABE) scheme for healthcare data sharing, achieving flexible and fine-grained data access control suitable for multi-user scenarios. Nevertheless, the computational complexity of this scheme increases system burden, especially in large-scale IoMT environments. Additionally, changes in user attributes require extra mechanisms to maintain data accessibility [16].

Moreover, ensuring the secure storage of encrypted data is an indispensable aspect of data privacy [17], [18], [19]. Centralized storage poses risks of single point of failure leading to data leakage and loss, while backup storage incurs additional costs. Therefore, Wang et al. [20] proposed a privacy-preserving cloud storage scheme based on fog computing. They utilized the Hash-Solomon encoding algorithm for data grouping, storing a portion of the data locally or on fog servers and the remainder on cloud servers. This setup prevents cloud servers from accessing all data, thus protecting data owners' privacy. However, in this scheme, if a storage server is tampered with or damaged, data cannot be promptly recovered. For instance, in cases of cloud server failures or data loss, some data may remain unrecoverable, leading to significant losses. In response to these challenges, we focus on researching the issues of secure data transmission and storage in multi-user scenarios of IoMT, aiming to provide more reliable guarantees for secure sharing of data. Our main contributions are as follows.

1) Leveraging the characteristics of PRE and CP-ABE, we propose a dual-cipher mechanism for multi-user data sharing. We improve the PRE scheme to enables all users to access shared data using a unified ciphertext and key. Concurrently, we employ CP-ABE to encrypt the small amounts of keys, enabling fine-grained access control. Even if user attributes change, it does not affect the ciphertext generated by PRE. Furthermore, after permission validation, decryption keys are encrypted using CP-ABE, eliminating the need for separate attribute revocation mechanisms.
2) We introduce a decentralized ciphertext storage and recovery mechanism based on Shamir secret sharing [21]. This approach distributes ciphertext fragments across multiple servers, enabling data recovery during server failures and preventing single points of failure.
3) To streamline permission determination, we implement a data sharing chain and attribute change chain. When user attributes change, the attribute change chain enforces constrains, while new attribute sets undergo validation to prevent unauthorized access. All data sharing activities are immutably recorded on the data sharing chain.
4) We formally verify the scheme's correctness and prove its confidentiality under the random oracle model. The scheme not only prevents single point of failure in storage servers, but also implements fine-grained access control for dynamic multi-users, resists conspiracy attacks, and realizes data access traceability. Performance evaluations confirm the scheme's suitability for resource-constrained IoMT environments.

The remaining sections of this paper are structured as follows. Section II discusses the existing related work. Section III introduces the preliminaries used in our scheme. Section IV presents our proposed IoMT data sharing architecture and security model. The details of the scheme are described in Section V. The correctness proofs and security analysis of the scheme are presented in Section VI. The performance of the scheme is evaluated in Section VII. Finally, we conclude our work in Section VIII.

II. Related Work

We review the related work from two aspects, data encryption sharing strategies and data sharing storage models.

A. Data Encryption Sharing Strategies

Mohammadali and Haghighi [22] presented a smart grid data aggregation scheme with multi-dimension and fault tolerance, supporting multi-category aggregation and batch authentication. Su et al. [23] introduced LCEDA, a lightweight and communication-efficient smart grid data aggregation scheme enabling efficient shield value sharing update and dynamic registration and revocation of smart meters, with freely formed aggregation zones for low communication and computational costs. Subsequently, Su et al. [24] proposed a scalable centralized data aggregation scheme using edge node aggregation groups for correct, secure, and efficient aggregation, leveraging symmetric encryption and online/offline signature for reduced online computation. Additionally, Guo et al. [25] introduced accountable proxy re-encryption to address proxy misbehavior without focusing on data visitor authentication and authorization, offering insights for efficient data sharing. Pei et al. [26] proposed a IoMT data security sharing scheme based on proxy re-encryption mechanism, which ensures the integrity of ciphertext through ciphertext verification mechanism to prevent tampering. However, these schemes do not focus on the use of shared data and ignore the importance of fine-grained control in data sharing.

CP-ABE is well-suited for fine-grained access control in data sharing [33]. Therefore, most researchers have considered secure data storage and privacy sharing using CP-ABE [27], [28], [29], [30], [31], [32]. For example, Xue et al. [34] introduced a CP-ABE data sharing scheme to counter economic denial of sustainability (EDoS) attacks, employing fine-grained access control policies and ciphertext randomization to regulate data downloads and prevent resource abuse. Zhang et al. [35] employed CP-ABE for confidentiality and fine-grained access control of shared telematics data in cloud and fog, introducing auditable user revocation for dynamic vehicle groups, and enhancing efficiency and correctness through online/offline and verifiable outsourcing techniques, though data integrity and traceability sharing were not addressed. Yang et al. [36] proposed an efficient and secure data sharing scheme for cloud storage. The scheme was implemented based on a multi-trusted third-party attribute encryption algorithm, which is also effective against collusion attacks. Vaanchig et al. [37] proposed a scalable and fine-grained cloud storage access control that supports effective user revocation policies. Zhang et al. [38] proposed an efficient attribute-based data sharing scheme with enhanced policy hiding and policy updating, addressing collusion issues between revoked users and the cloud by protecting user privacy through hiding sensitive attribute values in access policies. Muhammad et al. [39] proposed a secure data aggregation collection and transmission scheme, providing anonymity for patients' mobile devices and intermediate fog nodes. Ren et al. [40] introduced a certificateless autonomous path proxy re-encryption (CLAP-PRE) scheme utilizing multilinear maps, enabling fine-grained

access control to delegation paths in e-health systems, ensuring the security and privacy of patient health data shared with doctors. While the aforementioned schemes can achieve fine-grained access to data users, they impose heavy computation loads. Moreover, once a data user gains access to the data, it can have unlimited access to the data, which can be serious security risk for the complex environment of IoMT [41], [42].

Professor Jiguo Li's research team has made a series of groundbreaking advancements in the field of ABE, covering a complete technological chain from foundational architecture innovation to application scenario expansion. The team innovatively proposed a multi-authority ABE scheme supporting verifiable data deletion [43], which employs Merkle hash trees to achieve data deletion proofs, addressing the key escrow problem in traditional single-authority schemes while proving security under the DBDH assumption. The team's research system demonstrates a clear trajectory of technological evolution: in foundational architectures, they developed a Registered ABE framework [44] to eliminate key escrow risks entirely; in privacy preservation, they successively proposed a policy-hiding scheme with structural traceability [45] and a flexible encryption framework for multi-group scenarios [46]; in dynamic management, they constructed an efficient scheme integrating revocation mechanisms and integrity verification [47]. These achievements collectively form a comprehensive ABE technology ecosystem applicable to cloud storage, IoT, medical data, and other scenarios.

B. Data Sharing Storage Models

Sun et al. [48] proposed a secure data sharing model for smart terminals, utilizing self-authentication and blockchain for integrity verification and traceability, but doesn't address resource limitations or ensure safe data transmission to the cloud server. Ning et al. [49] proposed an encryption scheme by limiting the maximum number of times that a user can access privileges within a specified time. This achieves fine-grained access control and addresses the problem of granting unlimited access privileges to a user once the user's attribute set satisfies a given ciphertext access policy. Wang et al. [50] proposed a secure data sharing encryption scheme for cloud computing by introducing the concept of weighted attribute and enhancing the expression of attribute, which not only extends the expression of attribute from binary to arbitrary state but also reduces the complexity of access policy. However, Lan et al. [51] proved that this scheme is insecure in that users can obtain higher-level decryption privileges by tampering with their own attribute weights. Kumar et al. [52] addressed critical issues in medical data processing such as authentication, scheduling, redundant data removal, and data access time by introducing a solution based on Bloom filters. Jiang and Guo [53] proposed a secure cloud storage scheme utilizing conditional proxy broadcast re-encryption for dynamic user management without changing encryption keys, streamlining sharing while ensuring data privacy. However, these schemes do not address the issue of data corruption and loss due to a single point of failure in the storage server.

Wang et al. [54] proposed an efficient, revocable, and searchable privacy protection scheme using CP-ABE for mobile cloud storage. The scheme supports attribute revocation and outsources decryption to alleviate user-side computational burden, but requires verification of the outsourcing service provider's credibility and security to mitigate the risk of data leakage. Seth et al. [55] realized secure data storage in the cloud based on an integrated encryption technique. By devising a procedure for the secure distribution of cloud data, it provides reliability guarantee to customers and motivates them to store the information in confidential records. Islam et al. [56] introduced a lightweight image encryption technique based on substitution permutation networks, safeguarding the privacy of medical data through the generation of transformation magic blocks by subsystems and subsequent permutation processes. However, the complexity introduced by substitution permutation networks and their associated transformation magic blocks may escalate implementation and maintenance costs. Pu et al. [57] proposed a privacy-preserving edge data sharing scheme with data recoverable and attribute revocation. In this scheme, a blockchain-base attribute revocation chain was proposed to implement attribute revocation in CP-ABE, and a secret sharing scheme was introduced to assist data recovery. To cope with the case of a single server being hijacked, the efficient detection mechanism and key update strategy were proposed to ensure the security of the whole system. However, the data provider and the data consumer use the same encryption and decryption scheme, which is more suitable for data sharing based on edge servers, and is not suitable for data sharing in IoMT [58].

III. Preliminaries

This section introduces the necessary preliminaries.

A. Bilinear Mapping

Let G_1 and G_2 be two multiplication cyclic groups of order q, namely, $|G_1| = |G_2| = q$, and g be the generating element of G_1. Bilinear mapping $e : G_1 \times G_1 \to G_2$ satisfies the following properties.

1) *Bilinearity:* For any $x, y \in G_1$ and any $a, b \in Z_q$, $e(x^a, y^b) = e(x, y)^{ab}$.
2) Non-degeneracy. $e(g, g) \neq 1$.
3) Computability. For any $x, y \in G_1$, there exists an efficient algorithm to compute $e(x, y)$.

B. DBDH Assumption

1) Decisional Bilinear Diffie-Hellman (DBDH) Problem: Give multiplicative groups G_1 and G_2 of order p, and let g be the generating element of G_1. Randomly choose $a_1, a_2, a_3 \in Z_p$, $T \in G_2$, and use algorithm Φ to determine whether $T == e(g, g)^{a_1 a_2 a_3}$ holds. If it does, output 1; otherwise, output 0. The advantage Adv_{Φ}^{DBDH} of algorithm Φ is:

$$Adv_{\Phi}^{DBDH} = \Big| \Pr\left[\Phi\left(g, g^{a_1}, g^{a_2}, g^{a_3}, e(g, g)^{a_1 a_2 a_3}\right) = 1\right] - \Pr\left[\Phi\left(g, g^{a_1}, g^{a_2}, g^{a_3}, T\right) = 1\right] \Big|. \quad (1)$$

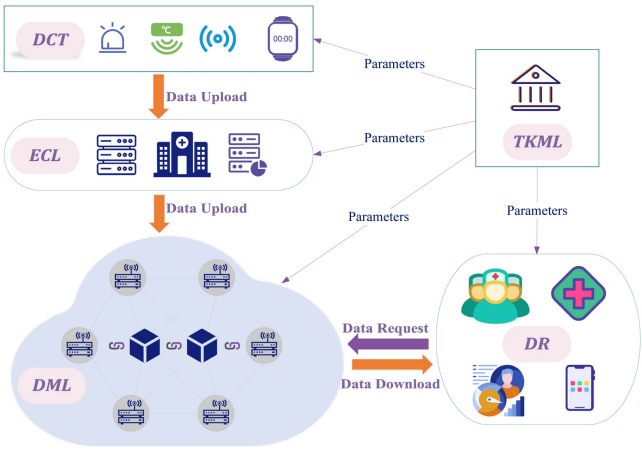


Fig. 1. The system model of the IoMT data sharing.

2) DBDH Assumption: The DBDH assumption is said to hold in groups G_1 and G_2 if for any probabilistic polynomial-time algorithm Φ, the advantage Adv_{Φ}^{DBDH} is negligible.

C. Cipher Policy Attribute-Based Encryption

CP-ABE is an access policy based attribute-based encryption scheme. It allows data owner to define the access policy, and only users who comply with the access policy can access the encrypted data. The formal definition of CP-ABE includes the following components [59].

1) Parameter Setting Algorithm $Setup_{CP-ABE}$*:* It generates the system master key MK and public parameters $param_{CP-ABE}$.

2) Key Generation Algorithm $KeyGeneration_{CP-ABE}(MK, AS)$*:* It uses the system master key MK and the user's attribute set AS to generate the attribute key sk_{AS} for the user.

3) Data Encryption Algorithm $Encrypt_{CP-ABE}(param_{CP-ABE}, M, \Lambda)$*:* It inputs the public parameters $param_{CP-ABE}$, the plaintext M to be encrypted and the access structure Λ, and outputs the ciphertext CT.

4) Ciphertext Decryption Algorithm $Decrypt_{CP-ABE}(param_{CP-ABE}, CT, sk_{AS})$*:* It inputs the public parameters $param_{CP-ABE}$, the ciphertext CT to be decrypted and the attribute key sk_{AS} based on the attribute set AS. If AS satisfies Λ, the plaintext M is outputted.

IV. System Overview

We begin by presenting the system model of the IoMT data sharing scheme. Subsequently, we provide a formal definition of the scheme, followed by a description of the security model.

A. System Model

As shown in Fig. 1, the IoMT data sharing system model primarily consists of the following 5 entities.

1) Data Collection Terminal DCT*:* It can be wearable device, medical sensor, and smart medical equipment, among others, with small computing and storage capabilities. It is responsible for collecting physiological data from patients or users, such as heart rate, body temperature, blood pressure, etc. It performs personalized encryption on the collected raw data and then uploads the result to the Edge Computing Layer.

2) Edge Computing Layer ECL*:* It is located near the healthcare data collection devices. In real life, it plays the role of a local medical institution and is trustworthy. ECL consists of small edge servers, denoted as $ECServer$, with robust computational and storage resources. The $ECServer$ is responsible for managing access control policies, customizing access permissions based on specific application requirements, data sensitivity, and other factors. Due to the large number of DCT, resource constraints, and strong decentralization, it is difficult to update the key. Therefore, ECL is primarily used for normalizing personalized terminal ciphertext into uniformly encrypted shared ciphertext using a unified key. The resulting ciphertext is then uploaded to the Data Management Layer. This allows for key updates to be performed only on the re-encryption key during key updates.

3) Data Management Layer DML*:* This entity realizes healthcare data storage and management. It comprises multiple servers, denoted as $DMServer$, with ample storage and computing resources. This layer serves as a crucial platform for IoMT data owners and users to store and share data. It assists data owners in storing their shared data and provides services to data users, facilitating access to authorized data. However, $DMServer$ are honest but curious and may be interested in the stored data. That is, it may try to gain as much private information as possible from the information they observe, potentially seeking unlawful gains by selling or leaking the data to unauthorized parties.

4) Data Requester DR*:* It is the initiator of data sharing services, and represents users in different roles such as doctors, pharmaceutical experts, and data analysts. Each DR has a set of attributes, and DR with legitimate permissions accesses healthcare data by requesting data from DML.

5) Trusted Key Management Layer $TKML$*:* As a fully trusted entity, $TKML$ is responsible for key management and registering other entities in the system. In the event of attribute changes for a DR, $TKML$ takes charge of regenerating and distributing a new key for the DR.

The process of data sharing is illustrated in Fig. 1, which can be divided into three main parts: data encryption processing, ciphertext distributed storage, and data sharing access. We describe this process in the context of IoMT data sharing.

In the data encryption stage, DCT comprises various devices: (1) Wearable devices (2) Medical sensors (3) Smart medical equipment. These devices collect patients' physiological and health data. The DCT performs preliminary encryption on the collected raw healthcare data and uploads the results to the nearby $ECServer$ within the ECL. The $ECServer$, which can be seen as a local clinic or healthcare facility, devises specific access policies and re-encrypts the data previously encrypted by DCT.

In the second stage, the ciphertext distributed storage stage, the re-encrypted data is uploaded to the DML, where the ciphertext is stored in distributed $DMServer$.

Finally, in the data sharing access stage, the DR, who could be a user such as a doctor, pharmacy specialist, or data analyst, requests access to the data. DML verifies whether the DR has the necessary permission for data access. If the DR passes the authentication, it can access and decrypt the requested shared data.

Hence, the data flow from collection to sharing and reaching the data requester involves several steps: '$DCT \rightarrow ECL \rightarrow DML \rightarrow DR$'. Here, DR represents the group requesting data access, indicating that the same data is transmitted multiple times during the previous flow, but transmitted to DR multiple times. Our focus is on securely sharing IoMT data with data requesters, while protecting the healthcare data shared by data owners from being accessed by illegal entities and being decrypted or revealed by unauthorized entities during the storage and sharing.

B. Formal Definition of the Scheme

By leveraging the features of proxy re-encryption and attribute-based encryption, we design a multi-user-oriented IoMT data sharing scheme, which incorporates dual cryptographic mechanisms to enable efficient data mining, sharing, and application. The scheme is formally defined as follows.

1) $Setup_1(1^\lambda)$*:* It generates the system parameters $param$ for the IoMT data sharing system.

2) $KeyGen(param)$*:* It contains $KeyGen_1(param)$ and $KeyGen_2(param)$, which are used to generate the public-private key pair for DCT and the specific public-private key pair, respectively.

3) $ReKeyGen(sk_{DCT_i}, pk_{peculiar})$*:* It generates the re-encryption key.

4) $ReKeyUpdate(sk_{DCT_i}, param)$*:* It updates the specific public-private key pair and re-encryption key.

5) $Encrypt_{DCT}(pk_i, m)$*:* DCT uses its public key to personalize the collected raw data m to obtain the personalized ciphertext.

6) $ReEncrypt_{ECL}(ReKey_{i\rightarrow p}, PC_i)$*:* $ECServer$ in ECL re-encrypts the personalized ciphertext to obtain the normalized ciphertext.

7) Distributed Trusted Storage of Ciphertext: DML slices the normalized ciphertext and distributes the fragments to other $DMServer$, so that each $DMServer$ stores only one piece of ciphertext, forming distributed trusted storage of ciphertext.

8) Secure Data Access for DR*:* According to the sharing request of DR, DML completes the verification of DR, the retrieval and aggregation of ciphertext with the assistance of $TKML$, and finally DR completes the decryption of ciphertext to realize the secure sharing of healthcare data.

C. Security Model

The confidentiality of data includes the confidentiality of personalized terminal encryption and the confidentiality of normalized re-encryption, defined as the chosen plaintext security of personalized terminal encryption and the chosen plaintext security of normalized re-encryption, respectively.

1) The Chosen Plaintext Security of Personalized Terminal Encryption: For an arbitrary personalized terminal encryption algorithm Φ, the choice of plaintext security for Φ is described by constructing the game between challenger C and adversary A. The game consists of the following phases.

1.1) Initialization phase: Challenger C runs $Setup_1$, obtains the system parameters $param$, and returns the system parameters to adversary A.

1.2) Query phase 1: Adversary A can execute the following queries.

- $q_{kc}(i)$. Query whose key has not been compromised. Challenger C runs $KeyGen_1$, obtains (sk_i, pk_i), and returns pk_i to adversary A.
- $q_{knc}(i)$. Query with compromised key. Challenger C runs $KeyGen_1$, obtains (sk_i, pk_i), and returns sk_i and pk_i to adversary A.

1.3) Challenge phase: Adversary A selects two messages m_0, m_1 of the same length to challenger C. Challenger C selects $b \in \{0, 1\}$ randomly, encrypts the message m_b into challenge ciphertext PC^* according to the encryption algorithm $Encrypt_{DCT}$, and returns the challenge ciphertext PC^* to adversary A.

1.4) Query phase 2: Adversary A executes the same query as in query phase 1.

1.5) Guessing phase: Adversary A makes a guess on b and outputs $b' \in \{0, 1\}$. If $b' = b$, adversary A is said to have won the game.

The advantage of adversary A in winning the game is denoted as $Adv_A = |\Pr[b' = b] - \frac{1}{2}|$. If no polynomial time algorithm can break the personalized terminal encryption algorithm Φ with advantage Adv_A, Φ is considered secure under the chosen-plaintext attack.

2) The Chosen Plaintext Security of Normalized Re-Encryption: For an arbitrary normalized re-encryption algorithm Φ, the choice of plaintext security for Φ is described by constructing the game between challenger C and adversary A. The game consists of the following phases.

2.1) Initialization phase: Challenger C runs $Setup_1$, obtains the system parameters $param$, and returns the system parameters to adversary A.

2.2) Query phase 1: Adversary A can execute the following queries.

- $q_{kc}(i)$. Query whose key has not been compromised. Challenger C runs $KeyGen_1$, obtains (sk_i, pk_i), and returns pk_i to adversary A.
- $q_{knc}(i)$. Query with compromised key. Challenger C runs $KeyGen_1$, obtains (sk_i, pk_i), and returns sk_i and pk_i to adversary A.
- $q_{rkc}(pk_i, pk_j)$. Query for re-encryption key. Challenger C runs $ReKeyGen$, obtains the re-encryption key $ReKey_{i\rightarrow j}$, and returns $ReKey_{i\rightarrow j}$ to adversary A.

2.3) Challenge phase: Adversary A selects two messages m_0, m_1 of the same length to challenger C. Challenger C selects $b \in \{0, 1\}$ randomly, encrypts the message m_b into challenge ciphertext UC^* according to the encryption algorithm $ReEncrypt_{ECL}(ReKey_{i\rightarrow p}, PC_i)$, and returns the challenge ciphertext UC^* to adversary A.

2.4) Query phase 2: Adversary A executes the same query as in query phase 1.

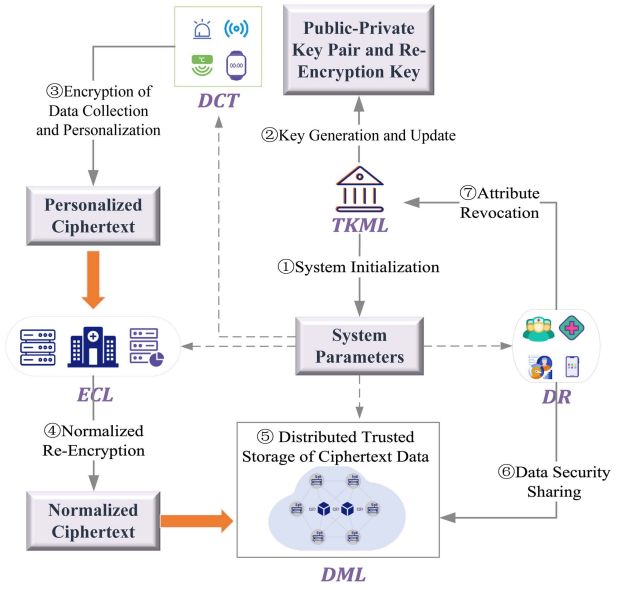


Fig. 2. Framework diagram of the proposed secure IoMT data sharing scheme.

2.5) Guessing phase: Adversary A makes a guess on b and outputs $b' \in \{0, 1\}$. If $b' = b$, adversary A is said to have won the game.

The advantage of adversary A in winning the game is denoted as $Adv_A = |\Pr[b' = b] - \frac{1}{2}|$. If no polynomial time algorithm can break the normalized re-encryption algorithm Φ with advantage Adv_A, Φ is considered secure under the chosen-plaintext attack.

V. Details of the Proposed Scheme

The framework of the proposed secure IoMT data sharing scheme is depicted in Fig. 2, which consists of seven components, namely, 1) System Initialization, 2) Key Generation and Update, 3) Data Collection and Personalized Terminal Encryption, 4) Normalized Re-encryption, 5) Distributed Trusted Storage of Ciphertext Data, 6) Data Security Sharing, and 7) Attribute Revocation. We now detail them one by one. The commonly used symbols in the scheme are shown in Table I.

A. System Initialization

$TKML$ needs to implement $Setup_1(1^\lambda)$ algorithm to provide a secure foundation for IoMT data sharing. Specifically, in the system initialization, $TKML$ uses $Setup_1(1^\lambda)$ to generate system parameters $param$ for personalized terminal encryption and normalized re-encryption. It sets the security parameter λ. For cyclic groups G_1 and G_2 of prime order q, where $q \geq 2^\lambda$, and bilinear mapping $e : G_1 \times G_1 \to G_2$, it randomly chooses $g_0, g_1, g_2, g_3 \in G_1$, and computes $I_0 = e(g_0, g_1)$ and $I_1 = e(g_0, g_3)$. The system parameters are $param = \{g_0, g_1, g_2, g_3, I_0, I_1\}$.

In addition, $TKML$ needs to run $Setup_{CP-ABE}$ algorithm in attribute-based encryption to generate the required public-private key and system parameters for the attribute-based encryption phase.

TABLE I
Symbols and Descriptions

Symbol	Description
DCT	IoMT Data Collection Terminal
ECL	Edge Computing Layer
$ECServer$	Edge Computing Server
$ECServer_0$	Nearest $ECServer$ in ECL to DCT
DML	Data Management Layer
$DMServer$	Data Management Server
$DMServer_0$	Nearest $DMServer$ in DML to ECL
DR	Data Requester
$TKML$	Trusted Key Management Layer
(sk_{DTC_i}, pk_{DTC_i})	DCT's public-private key pair
$(sk_{peculiar}, pk_{peculiar})$	The peculiar public-private key pair
$ReKey_{i\to p}$	Re-encryption key
m	Plain text
kws	Keywords
AS	Attribute set
$Tree$	Access control tree
δ	Maximum number of accesses
PC	Personalized ciphertext
UC	Normalized ciphertext

B. Key Generation and Update

With system parameters $param$ as input, $TKML$ executes $KeyGen(param)$ to generate DCT's public-private key pair and peculiar public-private key pair. This includes:

- Use $KeyGen_1(param)$ to generate public-private key pair $(sk_{DTC_i}, pk_{DTC_i}), i = 1, \ldots, n$ for DCT, where n is the number of DCT involved in data sharing. It randomly selects $sk^1_{DTC_i}, sk^2_{DTC_i} \in Z_q$ as the private key sk_{DTC_i} of DTC_i, i.e., $sk_{DTC_i} = (sk^1_{DTC_i}, sk^2_{DTC_i})$, calculates the public key $pk_{DTC_i} = (pk^1_{DTC_i}, pk^2_{DTC_i}) = (g_0^{sk^1_{DTC_i}}, g_2^{sk^2_{DTC_i}})$ of DTC_i, and distributes pk_{DTC_i} to DTC_i.
- Use $KeyGen_2(param)$ to generate peculiar public-private key pair $(sk_{peculiar}, pk_{peculiar})$. It randomly selects $sk^1_{peculiar}, sk^2_{peculiar} \in Z_q$ as $sk_{peculiar}$, i.e., $sk_{peculiar} = (sk^1_{peculiar}, sk^2_{peculiar})$, computes the public key $pk_{peculiar} = (pk^1_{peculiar}, pk^2_{peculiar}) = (g_0^{sk^1_{peculiar}}, g_2^{sk^2_{peculiar}})$, and distributes $pk_{peculiar}$ to ECL.

$TKML$ uses $ReKeyGe(sk_{DCT_i}, pk_{peculiar})$ to generate the re-encryption key. Re-encryption is that $ECServer$ in ECL encrypts the data again on the basis of the terminal personalized encryption. The re-encryption key $ReKey_{i\to p}$ is required to be generated from the private key sk_{DTC_i} of DTC_i and the peculiar public key $pk_{peculiar}$. It computes the re-encryption key $ReKey_{i\to p} = (g_1 g_3 pk^2_{peculiar})^{\frac{1}{sk^1_{DCT_i}}}$, and sends $ReKey_{i\to p}$ to ECL.

$ReKeyUpdate(sk_{DCT_i}, param)$ is used to update the peculiar public-private key pair to $(sk^{new}_{peculiar}, pk^{new}_{peculiar})$ and the re-encryption key to $ReKey^{new}_{i\to p}$. This key update process only needs to update the key in ECL and does not need to update the key in DCT. The update steps are as follows. First evenly select $sk^{1\_new}_{peculiar}, sk^{2\_new}_{peculiar} \in Z_q$ as the new private key $sk^{new}_{peculiar} = (sk^{1\_new}_{peculiar}, sk^{2\_new}_{peculiar})$, then compute the new

peculiar public key $pk^{new}_{peculiar} = (pk^{1\_new}_{peculiar}, pk^{2\_new}_{peculiar}) = (g_0^{sk^{1\_new}_{peculiar}}, g_2^{sk^{2\_new}_{peculiar}})$, and finally compute the updated re-encryption key $ReKey^{new}_{i\to p} = (g_1 g_3 pk^{2\_new}_{peculiar})^{\frac{1}{sk^1_{DCT_i}}}$.

C. Data Collection and Personalized Terminal Encryption

DCT is responsible for extracting keywords from the collected plaintext data, performing preliminary personalized encryption to obtain personalized ciphertext, and uploading the keywords and personalized ciphertext to the nearest $ECServer$ in ECL based on the network topology.

Specifically, the IoMT data collection terminal DTC_i extracts the keywords kws from the healthcare data m, which are terms or phrases used to describe the content or subject of the data, facilitating data retrieval by data requesters. Assuming DTC_i collects blood pressure information for user A, such as "Blood Pressure: 100/70mmHg," the corresponding kws for this data could include: "User A, Blood Pressure" or "DTC_i, Blood Pressure". DTC_i then executes the personalized terminal encryption algorithm $Encrypt_{DCT}(pk_i, m)$ by using its public key pk_i to personalize the collected data m into a personalized ciphertext PC_i. The process is as follows. DTC_i randomly selects $r_a \in Z_q$, obtains the personalized ciphertext $PC_i = (C_0, C_1, C_2, C_3)$ by computing (2).

$$\begin{cases} C_0 = m{I_0}^{r_a}, \\ C_1 = g_2^{r_a}, \\ C_2 = {I_1}^{r_a}, \\ C_3 = \left(pk^1_{DTC_i}\right)^{r_a}. \end{cases} \tag{2}$$

After obtaining the personalized ciphertext PC_i, DTC_i uploads $\{kws, PC_i\}$ to the nearest $ECServer$ in ECL based on the network topology, denoted as $ECServer_0$.

D. Normalized Re-Encryption

$ECServer$ makes access control policies and performs normalized re-encryption on the personalized encryption results. After $ECServer_0$ in ECL receives the key words and ciphertext of the data, the attribute-based access control policy of the data is formulated, and the access control tree $Tree$ is constructed based on this access control policy. Then $ECServer_0$ computes the hash value $H(kws)$ of the keywords kws and re-encrypts the personalized ciphertext PC using the re-encryption key $ReKey_{i\to p}$ to obtain the normalized ciphertext UC. This ciphertext can be decrypted by the peculiar private key $sk_{peculiar}$. More specifically, after receiving the personalized ciphertext PC_i, $ECServer_0$ uses the re-encryption key $ReKey_{i\to p}$ generated by $TKML$ to run the re-encryption algorithm $ReEncrypt_{ECL}(ReKey_{i\to p}, PC_i)$ to obtain the normalized ciphertext UC, by calculating (3), and forming $UC = (C'_0, C'_1, C'_2)$.

$$\begin{cases} C'_0 = C_0, \\ C'_1 = C_1, \\ C'_2 = \frac{e\left(C_3, ReKey_{i\to p}\right)}{C_2}. \end{cases} \tag{3}$$

When the data requester has the normalized ciphertext UC and the peculiar private key $sk_{peculiar}$, it can run the decryption algorithm $Decrypt(sk_{peculiar}, UC)$ with the input of the normalized ciphertext UC and the peculiar private key $sk_{peculiar}$ to get the data by (4). The correctness proof of this data decryption can be found in Subsection .

$$m = \frac{C'_0 \cdot e\left(g_0, C'_1\right)^{sk^2_{peculiar}}}{C'_2} \tag{4}$$

$ECServer_0$ uploads $H(kws)$, $Tree$ and UC to the nearest $DMServer$ in DML, denoted as $DMServer_0$.

E. Distributed Trusted Storage of Ciphertext Data

$DMServer$ sets the threshold of data access times, splits the normalized ciphertext, and distributes fragments to other $DMServer$s, so that each $DMServer$ only stores one piece of ciphertext. The purpose of setting the threshold is to prevent the risk of users having unlimited access to data once authorized. The size of the threshold can be determined based on, factors such as data management policies, data privacy levels, cost-effectiveness considerations. Splitting the ciphertext serves the purpose of ensuring effective recovery of user-stored data in case of a storage server failure, so as to prevent the user's data from being lost or corrupted. We now describe this distributed storage.

After receiving the normalized ciphertext UC from ECL, $DMServer_0$ first sets the upper limit δ for the number of accesses to this data, i.e., the same user can access this data at most δ times. Then, the normalized ciphertext UC is stored in each server of DML in a distributed manner, that is, UC is distributed as a secret parameter to other $DMServer$s in DML. The detailed steps are as follows.

Step 1: Polynomials construction. $DMServer_0$ constructs two t-order polynomials $f_1(x)$ and $f_2(x)$. Specifically, the binary string corresponding to UC is cut into t copies as the coefficients of $f_1(x)$, which are written in the order of $a_0, a_1, \ldots, a_{t-1}$, i.e., $f_1(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1}$. Randomly choose $b_0, b_1, \ldots, b_{t-1} \in \mathbb{Z}_q$ to construct $f_2(x) = b_0 + b_1 x + \cdots + b_{t-1}x^{t-1}$. $DMServer_0$ computes $V_k = g_0^{a_k} g_1^{b_k}$ for other $DMServer$s to verify that the ciphertext fragmentation, where $0 \le k < t$.

Step 2: Ciphertext splitting. $DMServer_0$ chooses n nonzero mutually unequal elements $x_1, x_2, \ldots, x_n \in Z_q$, computes $y_i = (f_1(x_i), f_2(x_i))$, $1 \le i \le n$, and assigns (x_i, y_i, V_k) to the ith $DMServer$ in DML, denoted as $DMServer_i$.

Step 3: Ciphertext verification. After receiving (x_i, y_i, V_k), $DMServer_i$ verifies if $g_0^{f_1(x_i)} g_1^{f_2(x_i)} == \prod_{k=0}^{t-1} (V_k)^{{x_i}^k}$ holds. Ciphertext fragment correctness can be proved in Subsection . If it holds, the ciphertext fragment $UC_i = (x_i, f_1(x_i))$ is valid, and $DMServer_i$ returns the message that the ciphertext fragment was received to $DMServer0$ and stores UC_i. Otherwise, $DMServer_i$ requests $DMServer_0$ to resend the correct fragment of the ciphertext.

The structure stored in $DMServer_i$ is shown in Fig. 3. Each $DMServer$ in DML stores not only a unique ciphertext fragment UC_i but also the keywords hash $H(kws)$ corresponding

UC_i	$H(kws)$	$Tree$	δ

Fig. 3. Data storage structure of $DMServer_i$ in DML.

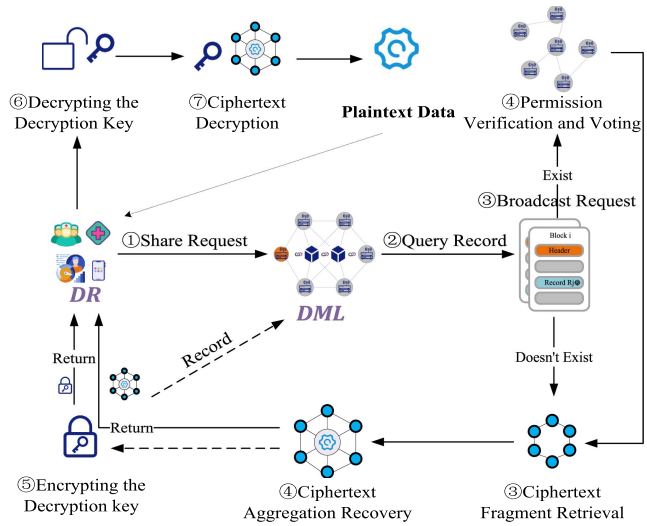


Fig. 4. The process of DR requesting shared data.

to the complete data, the access control structure $Tree$ and the access limit δ for data retrieval and verification.

To perform ciphertext recovery, it is sufficient to combine any at least t ciphertext fragments $UC_i = (x_i, f_1(x_i))$ to construct the following system of t linear equations

$$\begin{cases} a_0 + a_1 x_1 + a_2 {x_1}^2 + \cdots + a_{t-1} {x_1}^{t-1} = f_1(x_1), \\ a_0 + a_1 x_2 + a_2 {x_2}^2 + \cdots + a_{t-1} {x_2}^{t-1} = f_1(x_2), \\ \vdots \\ a_0 + a_1 x_t + a_2 {x_t}^2 + \cdots + a_{t-1} {x_t}^{t-1} = f_1(x_t). \end{cases} \tag{5}$$

By solving the above equations to obtain $a_0, a_1, \ldots, a_{t-1}$, we can get the complete ciphertext UC.

F. Data Security Sharing

DR retrieves the unique identifier, attribute set $AS = \{att_1, att_2, \ldots, att_{N_{as}}\}$, and attribute key sk_{AS} from $TKML$. When DR sends a data access request to its neighboring $DMServer$ in DML, the $DMServer$ verifies whether DR has permission to access the requested data. If permission is granted, the ciphertext fragments are retrieved and aggregated to form the normalized ciphertext. Simultaneously, $TKML$ is contacted to encrypt the decryption key of the normalized ciphertext, based on the attributes of DR. The resulting normalized ciphertext and the encrypted decryption key are then transmitted to DR. Subsequently, DR decrypts the received decryption key and utilizes it to decrypt the normalized ciphertext, thereby obtaining the plaintext data. Fig. 4 shows the data request process of DR from the nearest server $DMServer_0$ in DML. If related conflicts or intentional denial of access requests are involved, conflict resolution mechanisms and error handling mechanisms can be introduced, which will not be studied in depth in this paper. The specific steps are as follows.

Step 1: The DR sends a data request to $DMServer_0$, containing identity information, attribute hash $HAS = \{H(att_1), H(att_2), \ldots, H(att_{N_{as}})\}$, the hash value $H(kws)$ of the keywords kws for the data to be accessed. For the authenticated DR, it includes the following three situations.

- DR exists in the attribute change chain: Use the latest set of attributes in the attribute change chain as new HAS. Then go to Step 2.
- DR does not exist in the attribute change chain, and there are records of DR accessing the data in the data sharing chain: When the number of accesses is less than δ, $DMServer_0$ agrees to let DR access the data, and executes Step 3; Otherwise the access is directly denied.
- DR does not exist in the attribute change chain and the data sharing chain: Go to Step 2.

Step 2: $DMServer_0$ broadcasts $message = \{H(ID), HAS, H(kws)\}$ to other servers $DMServer_i$ in DML. For each "$DMServer_i$," the following steps are taken:

- $DMServer_i$ looks up and retrieves the data fragments, access count threshold, and the corresponding access control tree based on the hash value of the keyword set $H(kws)$.
- If the number of accesses is less than δ and the hash value of the attribute set matches $Tree$ successfully, $DMServer_i$ approves DR's data access and sends the data fragments to "$DMServer_0$". Otherwise, it rejects the data access request from DR.
- When the number of votes exceeds half, $DMServer_0$ agrees to let DR access the data and executes Step 3.

Step 3: $DMServer_0$ sends the access control tree $Tree$ and the information of DR to $TKML$.

Step 4: Based on the AS of DR, $TKML$ executes $Encrypt_{CP-ABE}(param_{CP-ABE}, sk_{peculiar}, Tree)$ algorithm to encrypt $sk_{peculiar}$ of into the ciphertext KC, and sends KC to $DMServer_0$. Here, $skpeculiar$ can decrypt UC.

Step 5: $TKML$ encrypts this shared record based on its own private key, generates signature, and sends the signature to the blockchain composed of $DMServer_i$ in $TKML$ and DML. When $DMServer_i$ receives the message, it determines whether the shared record comes from $TKML$ by verifying the signature. If so, store the record. That is, the data sharing record is stored in the data sharing chain and cannot be tampered with later, and the usage of the data can be tracked for further audit.

Step 6: $DMServer_0$ selects t of these $DMServer$s, constructs linear polynomial groups from the data fragments they sent, and derives $f_1(x)$. $DMServer_0$ uses the remaining ciphertext fragments to verify $f_1(x)$. The normalized ciphertext UC is derived after the verification passes.

Step 7: $DMServer_0$ sends the normalized ciphertext UC and the encrypted key KC to DR. DR decrypts KC using the attribute key sk_{AS}, and executes $Decrypt_{CP-ABE}(param_{CP-ABE}, KC, sk_{AS})$ algorithm to obtain the correct $sk_{peculiar}$. Then DR executes $Decrypt(sk_{peculiar}, UC)$ algorithm, and computes $m = \frac{C'_0 \cdot e(g_0, C'_1)^{sk^2_{peculiar}}}{C'_2}$, so that UC can be decrypted and plaintext m can be obtained. The proof process that the plaintext obtained by decryption is the original plaintext is in Subsection .

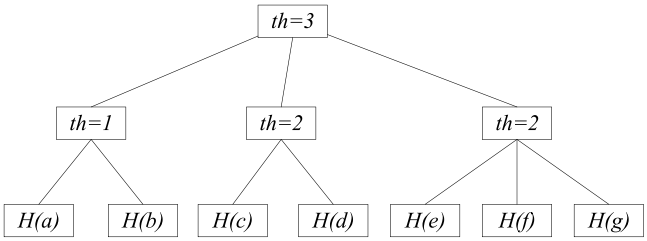


Fig. 5. Access control tree based on attribute hash.

For ease of understanding the matching process between the attribute set hash and the access control tree, we provide illustrative examples. Let's consider the current access control tree $Tree$ depicted in Fig. 5. In this representation, leaf nodes denote the hash values of attributes, specifically $H(a)$, $H(b)$, $H(c)$, $H(d)$, $H(e)$, $H(f)$, $H(g)$. Non-leaf nodes represent the minimum threshold values. Once the sum of its child nodes' values exceeds or equals the minimum threshold, the node is set to 1. This process is recursively applied upwards until it is determined whether it satisfies the threshold at the root node. If satisfied, it indicates that the attribute set of the corresponding DR complies with the requirements of the given $Tree$.

Let DR_1 have an attribute set $\{b, c, d, e, f\}$ and DR_2 have an attribute set $\{a, c, e, f, g\}$. According to the aforementioned conditions, DR_1 satisfies the access structure depicted in Fig. 5, while DR_2 does not. Through this mechanism, $DMServer$ can rapidly ascertain whether DR possesses the authorization to access the data.

G. Attribute Revocation

The algorithm 1 formalizes the process of managing attribute changes for a DR and ensuring data access authorization based on the latest attribute set hash.

When the attributes of DR change, the first step is for DR to request the latest attribute key from $TKML$. Subsequently, $TKML$ adds an attribute change record for DR to the attribute change chain, which is jointly maintained by $TKML$ and $DMServer$ in DMC. This record contains identity information of DR, the latest attribute set hash, and the attribute change time. After creating the record, $TKML$ encrypts it with the private key to form a signature and transmits this signature to the attribute change chain. Upon receiving the signature, $DMServer$ in DML verifies its authenticity to ensure it originates from $TKML$. If the verification is successful, $DMServer$ stores the attribute change record and maintains synchronization with the attribute change chain managed by $TKML$.

When a DR with changed attributes requests shared data from $DMServer_0$ in DML, the server first looks up the latest attribute set hash for DR in the attribute change chain. Following this lookup, the latest attribute set hash is compared against the access control structure to determine whether DR is authorized to access the requested data.

Algorithm 1: Attribute Change Management and Data Access Authorization.

ProcedureAttributeChange
  Input: Attributes of DR
  Output: Updated attribute change chain
  $key \leftarrow$ RequestLatestKey(DR)
  $record \leftarrow$ CreateRecord(DR, key)
  $signature \leftarrow$ Encrypt($record$, $privateKey$)
  SendToChain($signature$)
  $verified \leftarrow$ ChainVerifySignature($signature$)
  If $verified$ **then**
    ModifyChain(DR, $record$)
  else
    DenyModify()
  ‘
procedureAuthorizeAccess
  Input: DR, $sharedData$
  Output: Access decision
  $latestHash \leftarrow$ LookupHash(DR)
  $authorized \leftarrow$ MatchHash($latestHash$, $accessControl$)
  if $authorized$ **then**
    GrantAccess($sharedData$)
  else
    DenyAccess()

VI. Proof of Correctness and Security Analysis

A. The Correctness of the Data Sharing Model

1) Ciphertext Fragment Correctness Proof: In the distributed trusted storage phase of ciphertext data, $DMServer_i$ receives the data fragment (x_i, y_i) sent by $DMServer_0$ and verifies $g_0^{f_1(x_i)} g_1^{f_2(x_i)} == \prod_{k=0}^{t-1} (V_k)^{{x_i}^k}$ holds or not to decide whether the ciphertext fragment is valid or not. The correctness proof is as follows:

$$
\begin{aligned}
& g_0^{f_1(x_i)} g_1^{f_2(x_i)} = \left(g_0^{a_0 + a_1 x_i + a_2 {x_i}^2 + \cdots + a_{t-1} {x_i}^{t-1}}\right) \\
& \times \left(g_1^{b_0 + b_1 x_i + b_2 {x_i}^2 + \cdots + b_{t-1} {x_i}^{t-1}}\right) \\
& = \left(g_0^{a_0} g_1^{b_0}\right) \left(g_0^{a_1} g_1^{b_1}\right)^{x_i} \left(g_0^{a_2} g_1^{b_2}\right)^{{x_i}^2} \cdots \left(g_0^{a_{t-1}} g_1^{b_{t-1}}\right)^{{x_i}^{t-1}} \\
& = V_0 V_1^{x_i} V_2^{{x_i}^2} \cdots V_{t-1}^{{x_i}^{t-1}} = \prod_{k=0}^{t-1} (V_k)^{{x_i}^k}. \qquad (6)
\end{aligned}
$$

2) Data Decryption Correctness Proof: When DR has the normalized ciphertext UC and the peculiar private key $sk_{peculiar}$, it runs $Decrypt(sk_{peculiar}, UC)$ algorithm. The data m is calculated as $m = \frac{C_0' e(g_0, C_1')^{sk_{peculiar}^2}}{C_2'}$, and the correctness proof is as follows:

$$
\frac{C_0' e\left(g_0, C_1'\right)^{sk_{peculiar}^2}}{C_2'} = \frac{m I_0^{r_a} e\left(g_0, g_2^{r_a}\right)^{sk_{peculiar}^2}}{e\left(C_3, ReKey_{i \to p}\right) / C_2}
$$

$$= \frac{mI_0^{r_a} e\left(g_0, g_2^{r_a}\right)^{sk^2_{peculiar}}}{e\left(\left(pk^1_{DTC_i}\right)^{r_a}, \left(g_1 g_3 pk^2_{peculiar}\right)^{\frac{1}{sk^1_{DCT_i}}}\right)/I_1^{r_a}}$$

$$= \frac{mI_0^{r_a} e\left(g_0, g_2^{r_a}\right)^{sk^2_{peculiar}}}{e\left(\left(g_0^{sk^1_{DTC_i}}\right)^{r_a}, \left(g_1 g_3 pk^2_{peculiar}\right)^{\frac{1}{sk^1_{DCT_i}}}\right)/e(g_0, g_3)^{r_a}}$$

$$= \frac{mI_0^{r_a} e\left(g_0, g_2^{r_a}\right)^{sk^2_{peculiar}}}{e\left(g_0, \left(g_1 g_3 pk^2_{peculiar}\right)\right)^{r_a}/e(g_0, g_3)^{r_a}}$$

$$= \frac{mI_0^{r_a} e\left(g_0, g_2^{r_a}\right)^{sk^2_{peculiar}}}{e(g_0, g_1)^{r_a} e(g_0, g_3)^{r_a} e\left(g_0, pk^2_{peculiar}\right)^{r_a}/e(g_0, g_3)^{r_a}}$$

$$= \frac{mI_0^{r_a} e(g_0, g_2^{r_a})^{sk^2_{peculiar}}}{e(g_0, g_1)^{r_a} e\left(g_0, pk^2_{peculiar}\right)^{r_a}} = \frac{mI_0^{r_a} e(g_0, g_2^{r_a})^{sk^2_{peculiar}}}{I_0^{r_a} e\left(g_0, pk^2_{peculiar}\right)^{r_a}}$$

$$= \frac{m \cdot e\left(g_0, g_2^{r_a}\right)^{sk^2_{peculiar}}}{e\left(g_0, g_2^{sk^2_{peculiar}}\right)^{r_a}} = m. \tag{7}$$

B. The Security of the Data Sharing Model

1) Confidentiality of Data: The confidentiality of the data sharing scheme involves the personalized terminal encryption and the normalized re-encryption. Therefore, we prove that the scheme can guarantee the confidentiality of data through the security of personalized terminal encryption algorithm and the security of normalized re-encryption algorithm, respectively.

Theorem 1: Under the DBDH assumption, the personalized terminal encryption algorithm is consistent with the ciphertext indistinguishability under the choice of plaintext attack.

Proof: If there is an adversary A who can break the personalized terminal encryption algorithm with a non-negligible advantage Adv_A, then construct a challenger C to solve the DBDH problem with the advantage Adv_A.

T1.1) Initialization phase: Randomly select $u \in Z_q$, let $g_0 = g^a$, $g_1 = g^b$, $g_2 = g$, $g_3 = g^u$, $I_0 = e(g^a, g^b)$, $I_1 = e(g^a, g^u)$, and output $param = \{g_0, g_1, g_2, g_3, I_0, I_1\}$ as the system parameters.

T1.2) Query phase 1: Challenger C answers the query as follows.

- $q_{kc}(i)$: Randomly choose $sk_i^1, sk_i^2 \in \mathbb{Z}_q$. If it is the kth key query, let $i^* = i$, calculate $pk_{i^*} = (pk^1_{i^*} = g^{sk^1_{i^*}}, pk^2_{i^*} = g^{sk^2_{i^*}})$, let $sk^*_i = (\frac{sk^1_{i^*}}{a}, sk^2_{i^*})$, and return pk_{i^*} to adversary A. Otherwise, calculate $pk_i = (pk_i^1 = g_0^{sk_i^1}, pk_i^2 = g_1^{-1} g^{sk_i^2})$, let $sk_i = (sk_i^1, sk_i^2 - b)$, and return pk_i to adversary A.
- $q_{knc}(i)$: Randomly choose $sk_i^1, sk_i^2 \in \mathbb{Z}_q$, let $sk_i = (sk_i^1, sk_i^2)$, calculate $pk_i = (pk_i^1 = g_0^{sk_i^1}, pk_i^2 = g^{sk_i^2})$, and return (sk_i, pk_i) to adversary A.

T1.3) Challenge phase: When adversary A completes the query of phase 1, it outputs m_0, m_1 and the target public key pk^*. If $pk_{i^*} \neq pk^*$, challenger C outputs a random bit and rejects the response of adversary A. Otherwise, challenger C randomly chooses $b \in \{0, 1\}$, and calculates $C_0 = T \cdot m_b$, $C_1 = g_2^{r_a} = g^c$, $C_2 = I_1^{r_a} = e(g^a, g^c)^u$, $C_3 = (pk^1_{DTC_i})^{r_a} = (g^c)^{sk^1_{i^*}}$, where $r^a = c$. Let adversary A perform $q_{kc}(i)$ at most $n_{q_{kc}}$ times. Then the probability that challenger C guesses correctly is at least $\frac{1}{n_{q_{kc}}}$.

T1.4) Query phase 2: Adversary A executes the same query as in query phase 1.

T1.5) Output phase: Adversary A guesses b and outputs $b' \in \{0, 1\}$. If $b' = b$, challenger C outputs 1, otherwise outputs 0.

The probability that challenger C solves the DBDH problem is at least $\frac{Adv_A}{n_{q_{kc}}}$. Therefore, under the DBDH assumption, the personalized terminal encryption algorithm is indistinguishable from ciphertext under chosen plaintext attack.

Theorem 2: Under the DBDH assumption, the normalized re-encryption algorithm is consistent with the ciphertext indistinguishability under the choice of plaintext attack.

Proof: If there is an adversary A who can break the normalized re-encryption algorithm with a non-negligible advantage Adv_A, then construct a challenger C to solve the DBDH problem with the advantage Adv_A.

T2.1) Initialization phase: Randomly select $u \in \mathbb{Z}_q$, let $g_0 = g^a$, $g_1 = g^b$, $g_2 = g$, $g_3 = g^u$, $I_0 = e(g^a, g^b)$, $I_1 = e(g^a, g^u)$, and output $param = \{g_0, g_1, g_2, g_3, I_0, I_1\}$ as the system parameters.

T2.2) Query phase 1: Challenger C answered the query as follows.

- $q_{kc}(i)$: Randomly choose $sk_i^1, sk_i^2 \in \mathbb{Z}_q$. If it is the kth key query, let $i^* = i$, calculate $pk_{i^*} = (pk^1_{i^*} = g_0^{sk^1_{i^*}}, pk^2_{i^*} = g_1^{-1} g^{sk^2_{i^*}})$, let $sk_{i^*} = (sk^1_{i^*}, sk^2_{i^*} - b)$, and return pk_{i^*} to adversary A. Otherwise, calculate $pk_i = (pk_i^1 = g^{sk_i^1}, pk_i^2 = g^{sk_i^2})$, let $sk_i = (\frac{sk_i^1}{a}, sk_i^2)$, and return pk_i to adversary A.
- $q_{knc}(i)$: Randomly choose $sk_i^1, sk_i^2 \in \mathbb{Z}_q$, let $sk_i = (sk_i^1, sk_i^2)$, calculate $pk_i = (pk_i^1 = g_0^{sk_i^1}, pk_i^2 = g^{sk_i^2})$, and return (sk_i, pk_i) to adversary A.
- $q_{rkc}(pk_i, pk_j)$: Given pk_i and pk_j queried from $q_{kc}(i)$ and $q_{knc}(i)$, challenger C runs $ReKeyGen$ and returns the output.

T2.3) Challenge phase: When adversary A completes the query of phase 1, it outputs m_0, m_1 and the target public key pk^*. If $pk_{i^*} \neq pk^*$, challenger C outputs a random bit and rejects the response of adversary A. Otherwise, challenger C randomly chooses $b \in \{0, 1\}$, and calculates $C_0' = T \cdot m_b$, $C_1' = g_2^{r_a} = g^c$, $C_2' = I_0^{r_a} e(g_0, pk_{i^*}^2)^{r_a} = e(g^a, g^c)^{pk^2_{i^*}}$. Let adversary A perform $q_{kc}(i)$ query at most $n_{q_{kc}}$ times. Then the probability that challenger C guesses correctly is at least $\frac{1}{n_{q_{kc}}}$.

T2.4) Query phase 2: Adversary A executes the same query as in query phase 1.

T2.5) Output phase: Adversary A guesses b and outputs $b' \in \{0, 1\}$. If $b' = b$, challenger C outputs 1; otherwise outputs 0.

The probability that challenger C solves the DBDH problem is at least $\frac{Adv_A}{n_{q_{kc}}}$. Therefore, under the DBDH assumption, the normalized re-encryption algorithm is indistinguishable from ciphertext under chosen plaintext attack.

2) Prevent Storage Server From Single Point of Failure: To prevent the loss of shared data, we a distributed approach is used

to store the data. ciphertext is cut into n pieces by constructing t-order polynomials, and the ciphertext fragments are distributed to servers. Each storage server in the data management center stores only one unique fragment of the ciphertext and the hash of the keywords used for fragment retrieval.

When the data requester accesses the data, the data sharing is realized by retrieving and aggregating the data. In data aggregation, we do not need to aggregate all the ciphertext fragments but only need to combine any t of the data fragments to recover the original ciphertext. After the aggregator receives the ciphertext fragments, the remaining data fragments are used for verification to prevent a single point from being hijacked or data recovery errors. This approach not only greatly reduces the storage overhead of each server, but also prevents a situation where the ciphertext data cannot be recovered due to a failure or attack on one of the servers.

3) Resistance to Collusion Attacks: Our scheme can combat collusion attacks between storage servers in the data management center to obtain the access to stored shared data, which leads to data leakage. The ciphertext data in the data management center is encrypted at the personalized terminal encryption by DCT, and then normalized re-encryption by the edge computing center. The ciphertext is cut into n pieces by constructing t-order polynomials, and then the ciphertext fragments are distributed to servers. Only one of the ciphertext fragments is stored in each storage server of the data management center. It is impossible to obtain the complete ciphertext data without collusion among at least t storage servers.

Even in the extreme case where more than t storage servers collude to obtain the complete ciphertext, the ciphertext still needs to be decrypted by the specific key. The specific key is encrypted by the key management center based on the attributes of the data requester, and the storage server in the data management center cannot decrypt the plaintext of the shared data. Therefore, the shared private data cannot be accessed through the storage server of the data management center in our scheme.

4) Controlling Number of Accesses to Shared Data: To ensure that the data can be used reasonably and normally, the data management center sets the threshold δ for access times when the ciphertext data is received, and the same data can only be accessed by the same data requestor δ times. Each storage server in the data management center stores a ciphertext fragment and the corresponding threshold of access times for the ciphertext fragment. When a data requester makes a data sharing request, $DMServer$ in DML verifies if the number of accesses exceeds δ. If it does, access is denied. This prevents malicious users who have the corresponding permission from accessing the data indefinitely, avoiding the risk of data leakage. This prevents data requester from repeatedly making the data sharing request, thereby occupying communication bandwidth and wasting resources. This access control therefore prevents malicious users who have the corresponding permission from accessing the data indefinitely, avoiding the risk of data leakage.

5) Data Access Record Traceability: When the data requester DR makes a data request, $DMServer$ in DML needs to verify whether DR has shared data access authority. When the verification is passed, the key management center $TKML$ encrypts this data sharing record based on its own private key and generates the associated signature. Other participants in the data sharing chain determine whether the shared record comes from $TKML$ by verifying the signature. If so, the record is stored. The data sharing records are stored in the data sharing chain, enhancing the tamperability of the records. The blockchain structure ensures that each data sharing record is linked to the preceding one, forming a chain-like structure. Once a record is written into the chain, tampering with any individual record necessitates modifying all subsequent records, a formidable task on a blockchain. This guarantees the tamperability of shared record data. The data sharing chain records the access history of the data, and each data sharing record is linked to the previous record, making it easy to track data usage and enhancing the traceability of data access.

6) Resistance to EDoS Attacks: Some malicious data requesters who do not have access to the shared data may try to maliciously take up the communication overhead by continuously making data sharing requests and downloading large amounts of data, which causes the data sharing system to fail to function properly. The proposed IoMT data sharing system is resistant to this type of attacks.

When DR makes a data request, $TKML$ first needs to determine whether there is a record of DR accessing this data in the data sharing chain. If relevant records exist, it will further determine whether DR has access permission according to the data access threshold. If there is no access record, the storage server of the data management center first judges whether the attribute set of DR can construct a data access control tree, and then decides whether it has access permission based on the voting consensus mechanism. If the data access control tree is not satisfied or the data access threshold is exceeded, the requested data cannot be downloaded. Therefore, EDoS attacks cannot be launched by continuously maliciously downloading large amounts of data. If DR persistently sends numerous unauthorized data sharing requests, they will be blacklisted, resulting in the rejection of all subsequent access requests. This proactive measure effectively curtails resource wastage. To manage the subsequent access requests efficiently, automated processes or manual intervention can be implemented for periodic review and potential removal of flagged entities, ensuring continuous system security and resource optimization.

7) Forward and Backward Security: During the key generation phase, the scheme employs the key generation algorithm $KeyGen$ based on system parameters to generate public-private key pairs of DCT and special public-private key pairs, ensuring the randomness and security of the generated key. The key generation process utilizes the discrete logarithm problem within group theory as a security foundation, involving steps such as randomly selecting private keys and computing public keys. These steps, rooted in mathematical complexities, enhance forward security by making it challenging for attackers to compromise keys during generation.

In the re-encryption process, $ReKeyGen$ is utilized to generate re-encryption keys, ensuring the security of data during transmission. Within the $ReKeyGen$ process, public-private key pairs and special public-private key pairs are employed to guarantee the secure and reliable generation of re-encryption keys. When updating the special public-private key pairs, the

system enhances backward security by selecting new private keys and recalculating new special public-private key pairs and re-encryption keys. This updating process only affects the keys in ECL, avoiding the need to update keys in DCT and minimizing the impact of system updates on the overall system, thereby enhancing backward security.

8) Resistance to Man-in-The-Middle Attacks: Our scheme establishes a hierarchical security protection system through the collaborative operation of CP-ABE and proxy re-encryption. For access permission verification, the scheme adopts the CP-ABE, embedding access control policies directly into the ciphertext structure. This ensures that each legitimate user's decryption key is bound to their specific set of attributes. This design not only achieves fine-grained data access control but, more importantly, establishes an intrinsic correlation between the key and the user's identity, fundamentally eliminating the possibility of attackers obtaining valid information through simple eavesdropping. When a man-in-the-middle attempts to intercept communication content, they first encounter this layer of attribute verification—attackers lacking compliant attributes cannot effectively decrypt the encrypted data even if they obtain it.

On the dynamic protection level, the scheme further enhances system security through a key management mechanism. The periodic key update algorithm introduced in the proxy re-encryption phase ensures the time-constrained validity of encryption keys. The re-encryption key update algorithm in the scheme can securely perform key rotation while maintaining continuous system operation. This dynamic feature strictly limits the validity window of decryption keys, even if attackers obtain them through certain means during a specific period. More critically, all data access behaviors are recorded on the data-sharing blockchain, forming an immutable and complete audit trail. The dual safeguards of attribute verification and dynamic key management make man-in-the-middle attacks theoretically and practically insurmountable obstacles.

VII. Performance Evaluation

This section provides further performance evaluation for our proposed data sharing scheme.

A. Functionality Comparison

Table II compares our proposed data sharing scheme with several existing data sharing schemes, in terms of data sharing functionality. It can be seen that our scheme provides the most functionality, and it can implement dynamic multi-user fine-grained access control, distributed secure storage and recovery of ciphertext. Therefore, our scheme is more suitable for data sharing in IoMT edge computing and storage systems.

B. Experimental Evaluation

The proposed scheme is implemented on a user terminal device. We invoke the Java Pairing-Based Cryptography(JPBC) Library 2.0.0 [60] to implement the basic cryptographic operations in our scheme. In the experiment, we choose an elliptic curve of Type A on a 512-bit finite prime field, and the order of the multiplicative cyclic group on the elliptic curve is 160-bit. The hardware environment where the experimental program runs is Intel(R) Core(TM) i5-6500 CPU, 3.20 GHz CPU frequency, 12.0 GB memory. We use IntelliJ IDEA 2020.1.2 compilation tool on Windows 10 operating system for programming. The execution times required for basic cryptographic operations in this environment are listed in Table III.

Table IV compares the computational complexity of our proposed scheme with those of the existing schemes [25], [35], [48], [57], in terms of the computational overhead of each entity as well as execution time. It can be seen that our scheme imposes the lowest total execution time.

Next we carry out some experiments to evaluate the performance of the proposed model, in terms of the time consumption in the four phases of the encryption system, initialization, re-encryption key generation, plaintext encryption and ciphertext decryption as well as in the distributed storage.

1) Fixed Data Length: IoMT data are collected and uploaded in real-time, and often the data uploaded to the server in real-time is of the same type and the same length. Therefore, we set the constant data length of 128 bytes and investigate the execution time of each stage of the system as the number of data increases. The results obtained are depicted in Fig. 6.

TABLE II
Functionality Comparison

Scheme	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
[25]	✓	×	×	×	×	×	×	✓
[35]	✓	✓	×	×	×	×	✓	×
[48]	✓	✓	×	✓	✓	×	×	×
[49]	✓	✓	✓	×	×	×	✓	✓
[50]	×	✓	×	×	×	×	✓	×
[53]	✓	×	×	×	×	×	×	✓
[54]	✓	✓	×	×	×	×	✓	×
[57]	✓	✓	×	×	✓	✓	✓	✓
Our	✓	✓	✓	✓	✓	✓	✓	✓

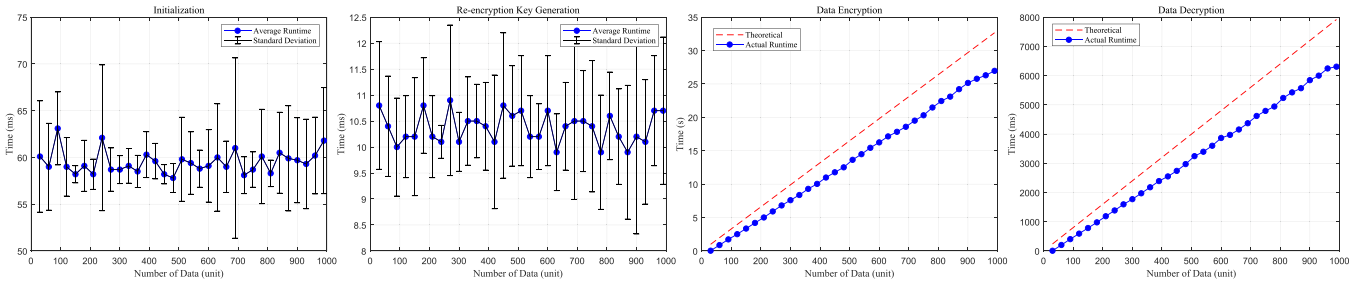
F_1: Data Confidentiality; F_2: Multi-User Access and Authentication; F_3: Limited Access Control; F_4: Data Access Traceability; F_5: Distributed Data Storage; F_6: Data Recoverability; F_7: Attribute Revocation; F_8: Resistance to EDoS Attacks.
Functionality exists: ✓ ; Functionality does not exist: × .

TABLE III
Execution Times of Basic Cryptography Operations

Symbol	Description	Execution time (ms)
T_{add}	Execution time for an elliptic curve point addition	0.129
T_{mul}	Execution time for an elliptic curve point multiplication	0.114
T_{exp}	Execution time for an elliptic curve point exponent	12
T_{inv}	Execution time for inversion operations on finite fields	0.033
T_h	Execution time for the hash operation	0.009
T_{pair}	Execution time of a bilinear pairing	25
T_{EnAtt}	Execution time of property-based encryption	84
T_{DeAtt}	Execution time of property-based decryption	36
T_{EnPk}	Execution time of public key encryption	43
T_{DeSk}	Execution time of private key decryption	9
T_{Sym}	Execution time of symmetric encryption/decryption	6

TABLE IV
COMPUTATION OVERHEADS OF FIVE SCHEMES IN TERMS OF NUMBERS OF CRYPTOGRAPHY OPERATIONS REQUIRED AND EXECUTION TIMES

Scheme	[25]	[35]	[48]	[57]	Our
Terminal	$8T_{exp}$+$2T_{pair}$ +$3T_{mul}$ (146.342 (ms))	T_{EnAtt}+T_{Sym} (90 (ms))	(2m+6)T_{mul}+T_{inv}+$2T_{pair}$ (0.228m+50.717 (ms)) (m is number of attributes)	T_{EnPk}+T_{EnAtt} (79 (ms))	$4T_{exp}$+T_{mul} (48.114 (ms))
Edge Computing Layer	T_{pair}+T_{inv} +T_{mul} (25.147 (ms))	$2T_{pair}$+$3T_{mul}$ (50.342 (ms))	$2T_{pair}$+(2m+2)T_{mul} +T_{inv} (0.228m+50.261 (ms))	-	T_{pair}+T_{inv}+T_{mul} (25.147 (ms))
Data Management Layer	-	-	$3T_{mul}$+$2T_{pair}$+T_{inv} (50.375 (ms))	T_{EnAtt}+$2nT_{Sym}$ (12n+36 (ms)) (n is number of servers)	$2T_{exp}$+T_{mul} (12.114 (ms))
Data Requester	$2T_{pair}$+$3T_{mul}$ +$2T_{inv}$ (50.408 (ms))	$2T_{exp}$+$2T_{mul}$ +T_{Sym} (30.228 (ms))	$2(m-1)T_{mul}$ +T_{add}+mT_{inv} (0.228m-0.066 (ms))	T_{EnPk}+T_{DeAtt} +T_{Sym} (85 (ms))	T_{exp}+T_{pair}+T_{inv} +$2T_{mul}$+T_{DeAtt} (73.261 (ms))
Total Execution Time	221.897 (ms)	170.57 (ms)	0.684m+151.287 (ms)	12n+200 (ms)	158.636 (ms)



(a) Initialization (b) Re-encryption Key Generation (c) Data Encryption (d) Data Decryption

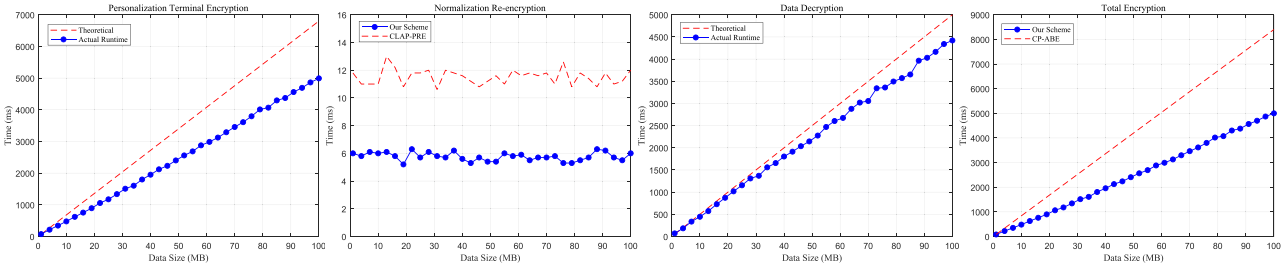
Fig. 6. Relationship between system runtime and the number of data. Each data has the fixed length of 128 bytes.

The system initialization is executed only once. The time consumption of each initialization does not have clear mathematical relationship with the amount of plaintext but is related to the algorithm design and more importantly to the specific experimental environment. In other words, the time consumption of system initialization is random. During the experiments, to test the runtime of the data encryption and decryption algorithms more accurately, we generated the key pairs in the initialization phase and hence the re-encryption key generation phase is also executed only once. At each execution, since the environment is not guaranteed to be exactly the same as the previous environment, the runtime fluctuates. Fig. 6(a) and (b) plot the mean runtime values and the associated standard deviations averaged over multiple runs in system initialization and re-encryption key generation phases, respectively. It can be seen that the time consumptions of system initialization and re-encryption key generation are random and do not depend on the number of data packets.

Fig. 6(c) and (d) depict the time consumptions in the data encryption and decryption phases, respectively, where it can be seen that the runtime increases linearly with the number of plaintext data. Our experimental results indicate that the encryption and decryption times of a single plaintext data are 33 ms and 8 ms, respectively. By multiplying the number of plaintext data with 33 ms and 8 ms, we obtain the theoretical runtimes of encryption and decryption phases, respectively, plotted in Fig. 6(c) and (d) as the red dashed lines. However, in practice, it is found that the actual runtimes are better than the theoretical ones. The reason is that we can pre-calculate the computations of the bilinear pairs during the actual operation so that the number of calculations of the bilinear pairs is reduced, which in turn reduces the time consumption.

2) Varible Data Length: In some scenarios, healthcare data are accumulated locally, and the data accumulated over a certain period of time are then uploaded. This forms the data of different sizes that need to be encrypted and decrypted. We set the data size from 1 MB to 100 MB and perform the personalized terminal encryption, normalized re-encryption and ciphertext decryption operations, respectively, to obtain the results shown in Fig. 7. Since the runtimes required in the system initialization and re-encryption key generation phases are random and do not have a clear relationship with the size of data, they are not included in Fig. 7.

It can be seen from Fig. 7 that the runtimes of the personalized terminal encryption and ciphertext decryption increase linearly with the data size, while the runtime of the normalized re-encryption phase is random. The reason is that the personalized terminal encryption and ciphertext decryption depend on the size of the plaintext data but the normalized re-encryption does not have clear relationship with the data size. For 1 MB of data, the runtimes required for personalized encryption and ciphertext decryption are about 68 ms and 50 ms, respectively. We add the theoretical runtimes of personalized terminal encryption and ciphertext decryption based on the



(a) Personalized Terminal Encryption (b) Normalization Re-encryption (c) Data Decryption (d) Total Encryption

Fig. 7. Relationship between system running time and individual data size.

results of 1 MB data as the red dashed lines in Fig. 7(a) and (c), respectively, where again it can be seen that the actual runtimes are better than the theoretical ones.

We compared the re-encryption phase of our scheme with that of CLAP-PRE [40], as illustrated in Fig. 7(b). It can be observed that the required re-encryption computation time remains nearly constant despite variations in plaintext data packet sizes, with our scheme consuming approximately half the time compared to CLAP-PRE. Since the re-encryption phase is independent of plaintext data, the consumed time remains relatively constant. Moreover, in designing the re-encryption phase, we aimed to minimize computational complexity by employing simple transformations, thereby avoiding multiple bilinear pairing operations as in CLAP-PRE.

In Fig. 7(d), the total encryption time in our scheme is compared with the CP-ABE encryption method. It is evident that, with an increase in data packet size, our scheme outperforms CP-ABE significantly. This is attributed to the fact that CP-ABE necessitates encrypting plaintext data based on access control structures, making the encryption process more intricate. Therefore, it is not well-suited for applications with substantial data requirements. However, in our proposed scheme, we employ attribute-based encryption for decryption keys associated with smaller data volumes, leveraging the advantages of attribute-based encryption algorithms.

3) Multi-Data Requester: In order to verify the efficiency of our scheme in a multi-user scenario, we introduced a simulation experiment featuring multiple data requesters interacting with the data storage center. In these experiments, we maintained a fixed packet size of 1 MB and measured the data encryption time statistics, presenting the results in Fig. 8. Where A-PRE is a data sharing scheme implemented using the proxy re-encryption method, proposed in [25]. It can be seen that the time of data encryption in our scheme remains almost the same as the number of users increases, while A-PRE is increasing. It is important to note that due to significant numerical differences, the encryption time for our proposed solution may appear to be 0 ms; however, the actual value is around 75 ms. This is because the proxy re-encryption scheme needs to re-encrypt the data once for each user if there are multiple data users when sharing the data, resulting in a linear increase in the encryption time with the number of users. Our scheme addresses this limitation of the proxy re-encryption approach by normalizing the ciphertext to the ciphertext decrypted by a unified key. This normalization enables a unified encryption process for multiple users, decoupling

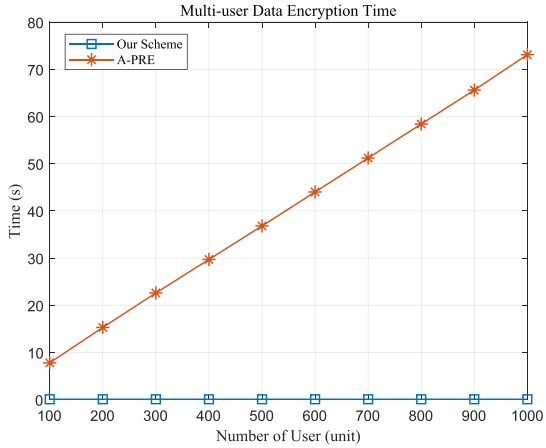


Fig. 8. Multi-user data encryption.

the data encryption time from the number of data requesters. As a result, our scheme demonstrates resilience to variations in the number of users, ensuring consistent data encryption performance.

4) Performance of Distributed Trusted Storage: We carry out some experiments to evaluate the proposed distributed trusted storage. As explained in Subsection V-E, in our scheme, after the data management center receives the ciphertext data uploaded from the edge processing center, it splits the ciphertext into n fragments by constructing t-order polynomials and distributes the ciphertext fragments to other servers to realize the distributed storage of data, where n is the number of servers and t is also called the ciphertext reconstruction threshold. When a data requester made a data sharing request, the data management center uses the ciphertext recovery method of Subsection V-E to reconstruct the ciphertext by combining t fragments of the ciphertext.

Fig. 9 depicts the computation times required for ciphertext splitting and reconstruction as the functions of the number of servers n. It can be seen that both the times required for ciphertext splitting and ciphertext reconstruction increase with n. The reason is that increasing the number of servers is equivalent to increasing the number of fragments, which increases the total number of polynomial coefficients to be constructed, thereby increasing the required ciphertext splitting and ciphertext reconstruction times. It can also be seen that the required ciphertext splitting and ciphertext recovery times for the threshold $t = \frac{2}{3}n$ are higher than the corresponding times for

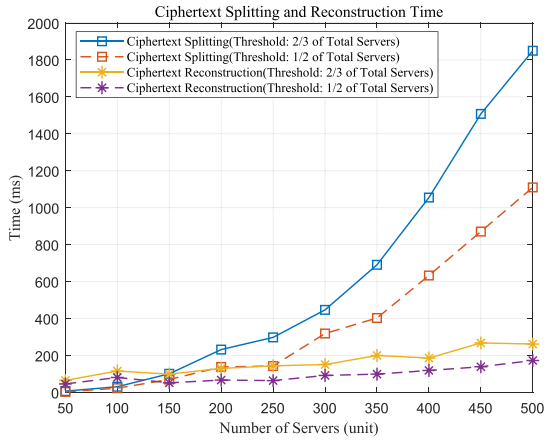


Fig. 9. Computation times for ciphertext splitting and reconstruction versus number of servers given two values of ciphertext reconstruction threshold t.

the threshold $t = \frac{1}{2}n$. The reason is that the larger the threshold value, the more polynomial coefficients need to be constructed and the more computation time is required. Moreover, under the same condition, ciphertext splitting takes much more time than ciphertext reconstruction.

5) Communication Cost: In our data sharing scheme, the health data undergoes a process of collection, encryption, storage, and then retrieval by authenticated data requesters. This process involves two main data flows: first, the data collection and storage phase "$DCT \rightarrow ECServer \rightarrow DMServer$", and second, the data sharing phase "$DMServer \rightarrow DR$". Let's analyze the data communication overhead in detail for each step of these data flows. Among them, the length of identity information is 128 bits, the length of keywords is 160 bits, the length of hash digest is 160 bits, the length of ciphertext is 1024 bits, and the maximum length of access control tree structure is 320 bits.

$DCT \rightarrow ECServer$: After DCT collects plaintext data with keywords $\{kws\}$, it performs personalized encryption to obtain PC_i, then uploads $\{kws, PC_i\}$ to $ECServer$. Therefore, the data transmission size is 160+1024=1184 bits.

$ECServer \rightarrow DMServer$: ECServer constructs an access control tree $Tree$, manages and encrypts the ciphertext to obtain normalized ciphertext UC, then uploads $\{H(kws), Tree, UC\}$ to $DMServer$. Therefore, the transmission size for this part is 160+320+1024=1504 bits.

$DMServer \rightarrow DR$: DR sends data request $\{Identity, HAS, H(kws)\}$ to DMServer. DMServer communicates with $TKML$ to verify DR's information and generates ciphertext for the decryption key if verification passes. The messages exchanged between them are $\{Identity, Tree\}$ and $\{KC\}$. Finally, $DMServer$ sends $\{UC, KC\}$ to DR. In this process, the total transmission size is {128+160+160}+{128+320}+{1024}+{1024+1024}=3968 bits.

VIII. Conclusion

Medical data sharing is integral to collaborative medical decision-making, research, and innovation. However, within the context of IoMT, the sharing of sensitive information poses risks of privacy breaches. To address this challenge, our research has focused on employing a dual encryption mechanism that combines proxy re-encryption and attribute-based encryption. This approach enables fine-grained access control and simplifies permission management. Additionally, we introduce a decentralized ciphertext storage and recovery mechanism based on Shamir secret sharing, enhancing data reliability and stability. Utilizing data sharing chains and attribute change chains streamlines the user permission determination process and ensures the security of shared records. Our proposed scheme is validated for correctness and confidentiality under the random oracle model, with performance evaluations demonstrating its suitability and efficiency in IoMT environments. The dual cryptographic mechanism proposed in the scheme is theoretically sound, but subsequent research and practical deployment must prioritize resolving interoperability issues between different healthcare institution information systems. Attribute-based encryption requires a unified attribute namespace and policy expression standards, which are currently lacking in the healthcare industry. The proxy re-encryption component necessitates establishing a unified key management framework for cross-institutional sharing, presenting significant challenges for implementation in distributed healthcare environments. Additionally, compatibility considerations with emerging standards must be addressed.

References

[1] S. Messinis et al., "Enhancing internet of medical things security with artificial intelligence: A comprehensive review," *Comput. Biol. Med.*, vol. 170, 2024, Art. no. 108036.
[2] V. Puri, A. Kataria, and V. Sharma, "Artificial intelligence-powered decentralized framework for Internet of Things in healthcare 4.0," *Trans. Emerg. Telecommun. Technol.*, vol. 35, no. 4, 2024, Art. no. e4245.
[3] X. Liu, P. Liu, B. Yang, and Y. Chen, "One multi-receiver certificateless searchable public key encryption scheme for IoMT assisted by LLM," *J. Inf. Secur. Appl.*, vol. 90, 2025, Art. no. 104011.
[4] G. Xu et al., "Anonymity-enhanced sequential multi-signer ring signature for secure medical data sharing in IoMT," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 5647–5662, 2025.
[5] S. F. Ahmed et al., "Insights into internet of medical things (iomt): Data fusion, security issues and potential solutions," *Inf. Fusion*, vol. 102, 2024, Art. no. 102060.
[6] S. Rani, S. Kumar, A. Kataria, and H. Min, "Smarthealth: An intelligent framework to secure IoMT service applications using machine learning," *ICT Exp.*, vol. 10, no. 2, pp. 425–430, 2024.
[7] B. Bhushan, A. Kumar, A. K. Agarwal, A. Kumar, P. Bhattacharya, and A. Kumar, "Towards a secure and sustainable internet of medical things (IoMT): Requirements, design challenges, security techniques, and future trends," *Sustainability*, vol. 15, no. 7, 2023, Art. no. 6177.
[8] O. J. Akindote et al., "Evaluating the effectiveness of it project management in healthcare digitalization: A review," *Int. Med. Sci. Res. J.*, vol. 4, no. 1, pp. 37–50, 2024.
[9] H. Jiang, P. Ji, T. Zhang, H. Cao, and D. Liu, "Two-factor authentication for keyless entry system via finger-induced vibrations," *IEEE Trans. Mobile Comput.*, vol. 23, no. 10, pp. 9708–9720, Oct. 2024.
[10] J. Hu et al., "WiShield: Privacy against Wi-Fi human tracking," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 10, pp. 2970–2984, Oct. 2024.
[11] Raghav et al., "Proactive threshold-proxy re-encryption scheme for secure data sharing on cloud," *J. Supercomputing*, vol. 79, no. 13, pp. 14117–14145, 2023.
[12] V. Muthukumaran and D. Ezhilmaran, "A cloud-assisted proxy re-encryption scheme for efficient data sharing across IoT systems," in *Research Anthology on Convergence of Blockchain, Internet of Things, and Security*. Hershey, PE, USA: IGI Global, 2023, pp. 626–646.

[13] F. Hu, H. Yang, L. Qiu, S. Wei, H. Hu, and H. Zhou, "Spatial structure and organization of the medical device industry urban network in China: Evidence from specialized, refined, distinctive, and innovative firms," *Front. Public Health*, vol. 13, 2025, Art. no. 1518327.

[14] C. Chen and J. Pan, "The effect of the health poverty alleviation project on financial risk protection for rural residents: Evidence from chishui city, China," *Int. J. Equity Health* vol. 18, no. 1, 2019, Art. no. 79.

[15] S. Das and S. Namasudra, "Macpabe: Multi-authority-based cp-abe with efficient attribute revocation for IoT-enabled healthcare infrastructure," *Int. J. Netw. Manage.*, vol. 33, no. 3, 2023, Art. no. e2200.

[16] M. Waqas et al., "The role of artificial intelligence and machine learning in wireless networks security: Principle, practice and challenges," *Artif. Intell. Rev.* vol. 55, pp. 5215–5261, 2022.

[17] A. Ometov, O. L. Molua, M. Komarov, and J. Nurmi, "A survey of security in cloud, edge, and fog computing," *Sensors*, vol. 22, no. 3, 2022, Art. no. 927.

[18] Y. Zhou, K. Liu, and P. Vijayakumar, "FTPS: Efficient fault-tolerant dynamic phrase search over outsourced encrypted data with forward and backward privacy," *Concurrency Computation: Pract. Experience*, vol. 34, no. 28, 2022, Art. no. e7360.

[19] M. Haus, M. Waqas, A. Y. Ding, Y. Li, S. Tarkoma, and J. Ott, "Security and privacy in device-to-device (D2D) communication: A review," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1054–1079, Secondquarter 2017.

[20] T. Wang, J. Zhou, X. Chen, G. Wang, A. Liu, and Y. Liu, "A three-layer privacy preserving cloud storage scheme based on computational intelligence in fog computing," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 3–12, Feb. 2018.

[21] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[22] A. Mohammadali and M. S. Haghighi, "A privacy-preserving homomorphic scheme with multiple dimensions and fault tolerance for metering data aggregation in smart grid," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 5212–5220, Nov. 2021.

[23] Y. Su, Y. Li, J. Li, and K. Zhang, "LCEDA: Lightweight and communication-efficient data aggregation scheme for smart grid," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15639–15648, Oct. 2021.

[24] Y. Su, J. Li, Y. Li, and Z. Su, "Edge-enabled: A scalable and decentralized data aggregation scheme for IoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 1854–1862, Feb. 2023.

[25] H. Guo et al., "Accountable proxy re-encryption for secure data sharing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 1, pp. 145–159, Jan./Feb. 2021.

[26] H. Pei et al., "Proxy re-encryption for secure data sharing with blockchain in Internet of Medical Things," *Comput. Netw.*, vol. 245, 2024, Art. no. 110373.

[27] S. Das and S. Namasudra, "Multiauthority CP-ABE-based access control model for IoT-enabled healthcare infrastructure," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 821–829, Jan. 2023.

[28] Y. Zhou et al., "TRE-DSP: A traceable and revocable CP-ABE based data sharing scheme for iov with partially hidden policy," *Digit. Commun. Netw.*, vol. 11, pp. 455–464, 2024.

[29] M. Xie, Y. Ruan, H. Hong, and J. Shao, "A CP-ABE scheme based on multi-authority in hybrid clouds for mobile devices," *Future Gener. Comput. Syst.*, vol. 121, pp. 114–122, Aug. 2021.

[30] Z. Ren, E. Yan, T. Chen, and Y. Yu, "Blockchain-based CP-ABE data sharing and privacy-preserving scheme using distributed kms and zero-knowledge proof," *J. Saud Univ.-Comput. Inf. Sci.*, vol. 36, 2024, Art. no. 101969.

[31] Y. Yang, J. Sun, Z. Liu, and Y. Qiao, "Practical revocable and multi-authority CP-ABE scheme from RLWE for cloud computing," *J. Inf. Secur. Appl.*, vol. 65, Mar. 2022, Art. no. 103108.

[32] S. Zhang, F. Guo, C. Jing, and C. Wu, "Electronic medical record privacy protection scheme based on attribute encryption technology," in *Proc. IAEAC 2024* Chongqing, China, Mar. 2024, vol. 7, pp. 402–412.

[33] M. Zhang, E. Wei, R. Berry, and J. Huang, "Age-dependent differential privacy," *IEEE Trans. Inf. Theory*, vol. 70, no. 2, pp. 1300–1319, Feb. 2024.

[34] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 8, pp. 2062–2074, Aug. 2018.

[35] J. Zhang et al., "Enabling efficient data sharing with auditable user revocation for IoV systems," *IEEE Syst. J.*, vol. 16, no. 1, pp. 1355–1366, Mar. 2022.

[36] Y. Yang, X. Chen, H. Chen, and X. Du, "Improving privacy and security in decentralizing multi-authority attribute-based encryption in cloud computing," *IEEE Access*, vol. 6, pp. 18009–18021, 2018.

[37] N. Vaanchig, W. Chen, and Z. Qin, "Ciphertext-policy attribute-based access control with effective user revocation for cloud data sharing system," in *Proc. CBD 2016*, Chengdu, China, Aug. 2016, pp. 186–193.

[38] L. Zhang, S. Xie, Q. Wu, and F. Rezaeibagha, "Enhanced secure attribute-based dynamic data sharing scheme with efficient access policy hiding and policy updating for iomt," *IEEE Internet Things J.*, vol. 11, no. 16, pp. 27435–27447, Aug. 2024.

[39] M. A. et al., "A secure and privacy preserved data aggregation scheme in IoMT," *Heliyon*, vol. 10, 2024, Art. no. e27177.

[40] C. Ren et al., "Clap-pre: Certificateless autonomous path proxy re-encryption for data sharing in the cloud," *Appl. Sci.*, vol. 12, no. 9, 2022, Art. no. 4353.

[41] B. Gong et al., "SLIM: A secure and lightweightmulti-authority attribute-based signcryption scheme for IoT," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 1299–1312, 2024.

[42] S. Tu, M. Waqas, A. Badshah, M. Yin, and G. Abbas, "Network intrusion detection system (NIDS) based on pseudo-Siamese stacked autoencoders in fog computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 6, pp. 4317–4327, Nov./Dec. 2023.

[43] J. Li et al., "Multiauthority attribute-based encryption for assuring data deletion," *IEEE Syst. J.*, vol. 17, no. 2, pp. 2029–2038, Jun. 2023.

[44] W. Weng, J. Li, Y. Zhang, Y. Lu, J. Shen, and J. Han, "Efficient registered attribute based access control with same sub-policies in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 9, pp. 8441–8453, Sep. 2025.

[45] J. Li et al., "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 762–773, Apr.–Jun. 2022.

[46] J. Li, E. Zhang, J. Han, Y. Zhang, and J. Shen, "PH-MG-ABE: A flexible policy-hidden multigroup attribute-based encryption scheme for secure cloud ctorage," *IEEE Internet Things J.*, vol. 12, no. 2, pp. 2146–2157, Jan. 2025.

[47] S. Chen, J. Li, Y. Zhang, and J. Han, "Efficient revocable attribute-based encryption with verifiable vata integrity," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10441–10451, Mar. 2024.

[48] H. Sun et al., "A fine-grained and traceable multidomain secure data-sharing model for intelligent terminals in edge-cloud collaboration scenarios," *Int. J. Intell. Syst.*, vol. 37, no. 3, pp. 2543–2566, 2022.

[49] J. Ning et al., "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 1, pp. 94–105, Jan. 2018.

[50] S. Wang et al., "Attribute-based data sharing scheme revisited in cloud computing," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 8, pp. 1661–1673, Aug. 2016.

[51] C. Lan, C. Wang, H. Li, and L. Liu, "Comments on 'attribute-based data sharing scheme revisited in cloud computing'," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 2579–2580, 2021.

[52] M. Kumar and A. Singh, "Bloom filter empowered smart storage/access in IoMT [edge-fog-cloud] hierarchy for health-care data ingestion," *Concurrency Computation: Pract. Experience*, vol. 36, 2024, Art. no. e8012.

[53] L. Jiang and D. Guo, "Dynamic encrypted data sharing scheme based on conditional proxy broadcast re-encryption for cloud storage," *IEEE Access*, vol. 5, pp. 13336–13345, 2017.

[54] S. Wang, D. Zhang, Y. Zhang, and L. Liu, "Efficiently revocable and searchable attribute-based encryption scheme for mobile cloud storage," *IEEE Access*, vol. 6, pp. 30444–30457, 2018.

[55] B. Seth et al., "Integrating encryption techniques for secure data storage in the cloud," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 4, 2022, Art. no. e4108.

[56] M. O. U. Islam et al., "Lightweight medical-image encryption technique for iomt based healthcare applications," *Multimedia Tools Appl.*, vol. 84, pp. 8929–8964, 2025.

[57] Y. Pu, C. Hu, S. Deng, and A. Alrawais, "R^2PEDS: A recoverable and revocable privacy-preserving edge data sharing scheme," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8077–8089, Sep. 2020.

[58] D. Zeng, A. Badshah, S. Tu, M. Waqas, and Z. Han, "A security-enhanced ultra-lightweight and anonymous user authentication protocol for telehealthcare information systems," *IEEE Trans. Mobile Comput.*, vol. 24, no. 5, pp. 4529–4542, May 2025.

[59] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. SP 2007*. Berkeley, CA, USA, May 2007, pp. 321–334.

[60] A. D. Caro and V. Iovino, "jPBCJ: JAVA pairing based cryptography," in *Proc. ISCC*, 2011, Kerkyra, Greece, 2011, pp. 850–855.