A Sustainable Data Encryption Storage and Processing Framework via Edge Computing-Driven IoT

Qi Li, Jian Huang*, Sihan Li and Chenze Huang

Abstract—**Edge computing serves as a critical intermediary for secure data exchange between IoT devices and data centres in healthcare, where the protection of sensitive patient data is paramount. This study presents UdesMec, a comprehensive edge computing framework designed for efficient and secure data storage and processing in such contexts. Focusing on the challenge of implementing robust data security models in the presence of limited resources at edge nodes, UdesMec employs a unified encryption approach that is particularly suited for expert knowledge-based prediction of complex critical medical events. Using secret sharing and homomorphic encryption, it ensures the confidentiality and integrity of sensitive patient data transmitted by IoT devices, while enabling efficient computation of prediction algorithms on a cloud server. The experimental evaluation confirms the strong encryption performance and the ease of use of UdesMec, positioning it as a promising solution for the secure and reliable prediction of complex critical medical events in healthcare systems based on edge computing.**

Index Terms—**Network security, edge computing, homomorphic encryption, Internet of things**

I. Introduction

In the context of the rapid evolution of smart transport, location services, mobile pay and other innovative service paradigms, driven by disruptive advances in IoT technology and the widespread deployment of 5G networks. There is a growing need to leverage these advances to improve healthcare, particularly in the area of expert knowledge-based prediction of complex and critical medical events [1]. The exponential proliferation of smartphones, wearable devices, smart TVs, and various sensor-equipped gadgets has led to an unprecedented surge in data generation at IoT terminals [2], [3]. It is estimated that by 2020, global data transmission surpassed a staggering 15.3 quintillion bits [4]. Concurrently, the number of Internet of Things (IoT) devices continues its relentless ascent, reaching approximately 50 billion in the same year [4].

In this era of pervasive connectivity, the traditional centralized big data processing approach, reliant on the cloud computing model, is increasingly found wanting in addressing the pressing demands for real-time processing, data security, and energy efficiency when confronted with the deluge of data generated by network edge devices [5], particularly in the healthcare domain where time-sensitive decision-making can be lifesaving. This underscores the need for novel solutions that can effectively leverage IoT-generated data and expert medical knowledge to predict and manage complex, high-risk medical events, thereby revolutionizing patient care and outcomes in the era of digital health [6].

Our research is firmly anchored in this landscape, focusing on the development and application of advanced, expert knowledge-driven predictive models for identifying and mitigating complex, life-threatening medical events. By integrating cutting-edge IoT technologies, real-time data analytics, and deep domain expertise, we aim to bridge the gap between the voluminous data generated at the network edge and the urgent need for swift, accurate, and informed clinical decision-making in the face of critical health situations.

Edge computing is a novel service that enables the processing of data or tasks to occur at the network edge, in close proximity to the data storage location [7]. The first cloud computing facility relocates all or a portion of its computational tasks to the data origin for execution. The network edge refers to any device or system that performs tasks, ranging from the data source to the cloud computing center. These organizations possess edge computing systems that integrate the primary functionalities of networking, processing, storage, and applications. These platforms provide consumers with computer services that are real-time, dynamic, and intelligent. The concept of processing data locally also provides more efficient assistance for safeguarding privacy and data security [8]. Edge computing is becoming crucial in domains such as the Internet, industrial robotics, autonomous vehicles, intelligent transportation, and other related fields [9]. This innovative autonomous architecture integrates cloud computing's storage, computation, and network tools with the edge of the network, enhancing the performance of large, collaborative online applications.

Large terminal networks make terminal device data storage, sharing, and other security concerns complex. Once the central processing node is attacked, the edge computing network will receive incorrect control directives, which might damage persons and property [6]. Edge computing applications commonly use real-time data to operate equipment. Mithun et al. [10] detailed edge network privacy security. Lu et al. [5] advocated IoT data privacy and aggregation. An integrated approach for secure data management and computation in edge computing environments is proposed here. Considering the expansive array of sensor nodes, diverse device types, and the network's inherent diversity, a standardized protocol for encrypting data at the edge nodes has been devised, designed to be seamlessly integrated with various

Manuscript received December 5, 2023; revised July 5, 2024.

Qi Li is a Lecturer of Shaoxing University, Shaoxing, Zhejiang 312000, China (e-mail: lsongru1106@163.com).

Jian Huang is an Associate Researcher of Shaoxing University, Shaoxing, Zhejiang 312000, China (e-mail: huangjian@usx.edu.cn).

Sihan Li is an undergraduate student of Shaoxing University, Shaoxing, Zhejiang 312000, China (e-mail: li.sihan.0815@gmail.com).

Chenze Huang is a PhD candidate of Research and Development Institute of Northwestern Polytechnical University, Shenzhen, Guangdong 518057, China (e-mail: yxy_0713@qq.com).

application contexts. Our proposed framework delineates the architecture of edge computing into distinct tiers: cloud services, edge processing, and application interfaces. Owing to the limited computational and storage capabilities inherent in edge node data handling, cryptographic techniques such as secret distribution and encryption that allows computation on ciphertexts are utilized to transmit data securely and offload the bulk of processing tasks to the cloud server. The cloud computes and analyzes the edge node ciphertext. The encryption approach protects user data. The innovations of this paper are summarised as follows:

- We designed a complete data encryption model for edge computing. The edge node and cloud server use this approach to securely transfer ciphertext IoT data to the edge layer, cloud service layer and application layer. The cloud server handles the operation and storage of the ciphertext, offloading the computational and storage load from the edge computing nodes. Data collection and processing under encryption is efficient.
- We engineered a security framework at the application level for user authentication and permission management, which facilitates the secure transmission of IoT sensor information to the cloud's computational infrastructure while mitigating the concerns regarding potential cyber threats. Cloud providers are capable of aggregating their server capacities to facilitate edge computing operations, serve a multitude of user communities through a unified hardware and software platform, and regulate the data access permissions for users.
- We evaluated the influence of the model on communication and computational performance through numerous experiments, analyse its suitability for real-world edge computing applications, and demonstrate the effectiveness of our proposed model.

II. Related Work

Currently, cloud computing has well-developed and comprehensive security protection methods and technological solutions [11]. Nevertheless, the evolving features of lightweight devices and the diverse architecture of edge computing render the security paradigms of conventional cloud computing obsolete [12]. The use of cryptographic technology in the edge computing environment is crucial for resolving data security concerns [13].

Edge computing design complicates key negotiation since cloud servers, edge servers, and users hold and handle key information differently. Key agreement technology is needed for edge node-cloud server communication [14]. For security, each edge computing connection needs a key. Authentication and session keys are needed by the device and communication layers, respectively. Comprehensive and secure key systems are necessary. An identity-based anonymous authentication key agreement system for mobile edges was developed by Jia et al [15]. The protocol assures user anonymity and non-traceability and verifies both participants' identities in one message exchange cycle. The protocol's implementation is constrained by device processing capacity, even if the calculation time is faster than other techniques. The authors' [16] three-party key negotiation approach for the CK (Canetti and Krawczyk) security model does not preserve device and server anonymity.

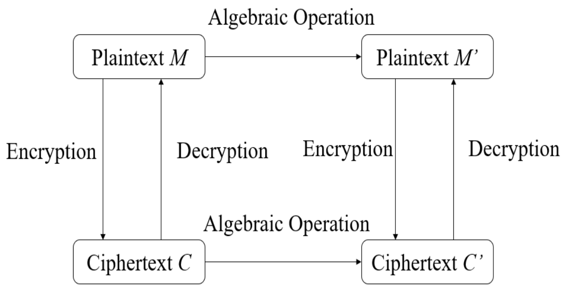

Fig. 1: Fully homomorphic encryption.

The most prevalent use of cryptography in edge computing is communication data encryption. Edge networks may store localized data shared by several users and have a lower data storage capacity than the cloud [17]. The edge computing framework's public key encryption algorithm is too difficult for terminal devices with limited resources to calculate for decryption [18]. Lightening the encryption algorithm remains a challenge. Most public-key encryption systems are identity-based [12]. Kim et al. [19] presented identity-based broadcast encryption for edge computing data encryption. It decreases terminal device processing and communication costs by offloading partial decryption to the edge. When edge devices require more sophisticated computing resources and services to process and analyze data, they outsource it to a third party, separating ownership and data control. Fully homomorphic encryption [2], [3] is a practical strategy for data outsourcing since it ensures that ciphertext algebra operations yield the same results as plaintext. Fig. 1 depicts completely homomorphic encryption.

Traditional cloud computing privacy protection techniques do not apply to edge devices, and edge nodes cannot execute algorithms [14]. Existing related efforts cannot combine cloud and edge computing effectively [4], [16]. This study proposes a hierarchical security integration paradigm for end-cloud integration that securely stores data in the cloud and edge nodes according to application needs. The model may also reliably interface with end-cloud data, deliver suitable apps, and fully integrate cloud and edge computing.

III. UdesMec Model

The full hierarchical security integration model is shown in Fig. 2, which is comprised of three layers: the application layer, the edge layer, and the cloud service layer.

Edge Layer: It is comprised of interconnected networks of IoT devices and nodes specifically designed for edge computing. Every terminal network consists of a designated quantity of edge computing nodes, which are charged with the collection and processing of network information. Within the structure of every terminal network, a precise number of edge computing nodes is designated to undertake the critical tasks of gathering and meticulously analyzing the pertinent network data. Each terminal network is equipped with a set number of edge computing nodes that are entrusted with the responsibility of amassing and scrutinizing the network data. These equipments, such as personal computers, servers, or base stations, are characterized by their computational capabilities. They collect data from IoT devices and provide specialized capabilities for calculating and storing data.

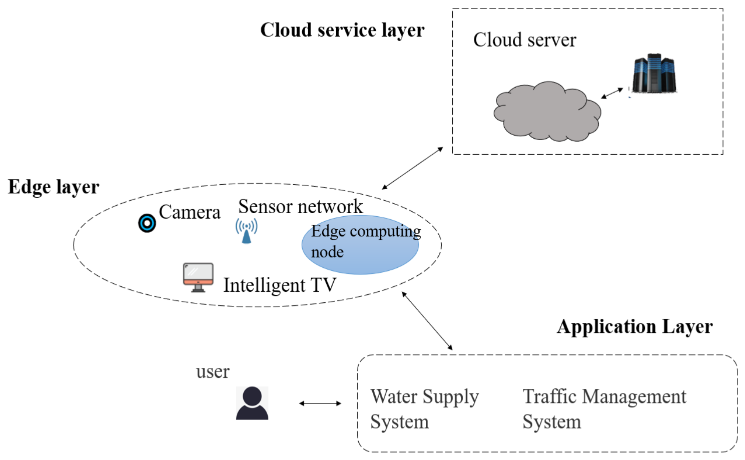

Fig. 2: Hierarchical security integration model diagram.

The cloud service layer and application layer's network architecture is resilient to the fluctuating numbers of IoT devices and edge computing nodes within the edge layer. The expandability ensures that the edge layer devices are adept at handling extensive decentralized networks. The network's design is such that it accommodates changes in the IoT ecosystem without disruption, thereby enhancing its suitability for expansive and distributed networks. The architecture's inherent flexibility ensures that the edge layer remains operational and efficient, even with the ebb and flow of IoT devices and computing nodes.

Cloud Service Layer: Cloud servers from service providers receive encrypted, pre-processed data from edge computing nodes. Edge nodes homomorphically encrypt data, allowing for direct encryption on the cloud server without the need for cloud service layer decryption, thereby safeguarding the privacy of IoT device data.

Application Layer: Data owners use service provider application software to link many IoT devices to the edge computing network. For cloud storage, IoT devices transfer data to edge nodes. Data viewing and analysis must be available in the service provider's application system. Service providers must optimize server resources to serve clients. Users may access diverse data using a sophisticated user authentication and access control method to secure data and prevent hostile attackers from accessing it. Providers furnish a comprehensive edge computing framework enabling user collectives to interface with a centralized system aligned with their credentials, for the purpose of aggregating and analyzing data. The edge computing framework offered by service providers is designed to cater to various user groups, each with access to a centralized system tailored to their specific identities, facilitating the aggregation and analysis of data.

A. *Edge Layer Ciphertext Model*

The IoT connects tangible things to the digital world using integrated circuits and detectors. A large, multi-geographical sensor network can collect a significant amount of valuable data to improve people's understanding of their environment. Visual data, the primary type of data collected by surveillance cameras, is used extensively in autonomous monitoring tasks such as road surveillance, pedestrian flow identification and anomalous activity alerts. Advancements in image processing and data analysis technologies allow anyone to utilize surveillance camera footage for various data analysis tasks, including quantity statistics, face recognition, license plate recognition, and real-time tracking.

This section primarily uses the experimentally developed system as a case study to clarify the concept of the edge layer cipher. The IoT terminals refer to cameras that are part of a distributed network. These cameras communicate video data to the edge computer node. The edge computing node analyzes the picture to detect people, producing their coordinates, color histograms, and other pertinent data. Because of the vast volume of data, the character color histogram is classified as unstructured data and sent as a file inside the integrated security system. In order to establish the data model of the system, it is crucial to generate an individual meta-database for each edge node. This meta-database will store distinct keys and supplementary data.

For edge node a, randomize two prime numbers p_1 and p_2 (typically 1024 bits) to get their product n using the RSA technique. Formula (1) generates two prime numbers (p_1 = 53, p_2 = 59) for node a, and n = 3127, yielding $\varphi(n)$= 3016. We use smaller keys to enhance readability and simplify computation. We must produce a random number p, which is a positive integer that is relatively prime to n.

$$\varphi(n) = (p_1 - 1)(p_2 - 1) \tag{1}$$

After the edge node's metadata has been set up, the node initiates and is equipped to accept the raw video footage obtained from the cameras. Upon the completion of the recognition processing for each frame of the camera data, the derived structured data, which is illustrated in Table I. The data consists of the frame sequence number, identified as *id*, the camera number, identified as *cam_id*, the time stamp, identified as *timestamp*, the coordinates of the person's square, identified as x_1, y_1, x_2, y_2, and the RGB histogram file of the person. The symbols "S" and "PRI" are designated as reserved fields for ciphertext operation. Each field in the edge node is assigned a distinct column key, denoted as *ck*. The edge node then uses the produced n to acquire a ciphertext value, represented as V_e, for each field. The value of V_{key} is produced by the combination of ck_A and r_i, as shown in Formula (2).

TABLE I: COMPARISON OF COMMUNICATION OVERHEAD OF THE CIPHERTEXT MODEL.

Field	V	*ck*	V_e
id	1	NULL	NULL
cam_id	1	(1,3)	389
timestamp	00001	(2,2)	1642
x_1	21	(3,3)	2549
y_1	93	(2,5)	1041
x_2	172	(2,7)	811
y_2	469	(9,10)	495
histogram	*xxx,npy*	NULL	NULL
S	1	(2,3)	2514
PRI	3	(1,5)	387

$$V_{key} = g\left(r, (x, y)\right) = xp^{ry \bmod \varphi(n)} \bmod n \tag{2}$$

Next, V_e is produced by combining V_{key} with the plaintext V. V_e represents the encrypted data value, as shown by Formula (3).

$$V_e = E\left(V, V_{key}\right) = VV_{key}^{-1} \bmod n \tag{3}$$

The term V_{key}^{-1} refers to the modular inverse of V_{key}.

$$V_{key}V_{key}^{-1} \bmod n = 1 \tag{4}$$

The secured structured data includes the derived structured information, including a person's coordinates. The histogram provides the essential data for identifying individuals, presented as a document. The edge node must transmit solely the encrypted value V_e along with the individual's histogram document to the cloud service layer for subsequent storage and processing by the cloud server. In scenarios where edge nodes have limited storage capabilities, they may only be able to retain minimal original video data, or in some instances, may not be able to store any original video data at all. Moreover, by utilizing the RabbitMQ message queue for the transfer of data between the cloud service layer and the edge layer, the data is retained at the edge node during periods of weak or absent network connectivity. As soon as network connectivity is reestablished, the data transmission will proceed without incurring any losses attributed to network interruptions.

B. Ciphertext Model of Cloud Service Layer

Each edge node supplies frame data for the cloud server's use. Owing to delays inherent in network transmission, the identical frame data from different nodes is not simultaneously deliverable to the cloud server. Given the inherent delays in network transmission, frame data from multiple edge nodes cannot be simultaneously conveyed to the cloud server. Consequently, due to the network latency, the frame data from disparate edge nodes cannot be synchronized in their transmission to the cloud server. Frame sequence numbers are used to synchronize frame data across devices. TABLE I shows that the frame sequence number is saved in plaintext, which simplifies cloud service data synchronization and avoids decrypting data each time.

Once the cloud server has synchronized the frames, the system matches the observed entities with the reference set, employing technology for re-identifying entities to acquire movement paths of subjects under surveillance. Upon synchronization of the frames by the cloud server, the system engages in a comparison of the observed entities with a reference set, applying re-identification techniques to ascertain the movement trajectories of individuals captured by the camera. Post the synchronization of frames on the cloud server, the system then leverages character re-identification technology [20] to extract the trajectory data of the individuals within the camera's view, which is subsequently encrypted and securely stored.

TABLE II: A SAMPLE OF THE DATA ORGANIZATION FOR ENTITY MOVEMENT RECORD IN THE CLOUD-BASED SYSTEM (n = 2907, p = 1).

Field	V	*ck*	V_e
id	1	NULL	NULL
cam_id	1	(1,3)	389
timestamp	00001	(2,2)	1642
x_1	21	(3,3)	2549
y_1	93	(2,5)	1041
x_2	172	(2,7)	811
y_2	469	(9,10)	495
histogram	*xxx,npy*	NULL	NULL
S	1	(2,3)	2514
PRI	3	(1,5)	387

The cloud service layer maintains the confidentiality and accuracy of the entity movement records by safely storing the encrypted data within the cloud infrastructure. Owing to the implementation of homomorphic encryption within the data framework, the data proprietor can promptly perform the requisite computations on the encrypted data without compromising the security of the information. When it becomes essential to access the data, the data owner may decode the data to acquire the corresponding plaintext information.

C. User Authentication and Key Transmission in Application Layer

The application layer comprises entities such as service providers and end-users. The service providers, who are the custodians of cloud server assets, adopt a security integration methodology at the edge nodes. Within the application layer, service providers, who possess cloud server assets, implement a security consolidation strategy on the edge nodes to ensure data protection and seamless operation. The database key (containing n, p, p_1, p_2, and column key *ck*) is sent and stored in a secure database, functioning as a trusted third party (TTP) to authenticate data owners accessing their own data. Data owners may also keep edge node key information and develop their own encryption and decryption client applications. This strategy also protects cloud service provider data. Due to expense, small data owners generally don't have a security key database or a powerful data firewall like cloud service providers. The model in this section solely considers the service provider keeping the key.

Users must receive the decryption key from the service provider to access plaintext data. User authentication verifies identity and provides keys. Cloud servers and TTP are SP resources. The user logs in to TTP and requests an access token (AT) with its identifying information. TTP returns user-generated AT after identification verification. The system requests database key information from SP when the user approves AT for the client application. Once AT is verified,

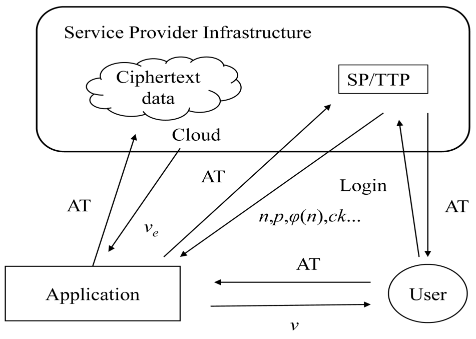

Fig. 3: User authentication and key transmission model diagram.

SP sends the database key information (e.g., *n*, *p*, $\varphi(n)$, *ck*) to the client application. The client application utilizes AT to request the ciphertext from SP's cloud server after obtaining the key information. The cloud server sends the database ciphertext to the client application after verifying AT. After decrypting using the database key, the client application delivers the plaintext data. Fig. 3 shows the full ciphertext acquisition procedure.

D. Data Analysis and Application

1) Ciphertext Data Operation in Cloud Service Layer: The strategy for integrated security entails initial data processing from IoT end-devices at the edge, followed by the secure transmission to the cloud for secure storage in encrypted form. The data, once uploaded, is stored securely in the cloud in an encrypted format, enabling the data owner to perform operations on the cloud-hosted data and receive corresponding encrypted results. Upon the cloud's successful re-identification of individuals in each frame of the transmitted data, it proceeds to calculate the individual's position relative to the camera, utilizing this data to generate a map illustrating the person's movement path. To perform this operation, the ciphertext values x_1 and x_2 of the coordinate data must be added together to produce the ciphertext result. Then, the ciphertext must be decrypted by the data owner to determine the character's relative location on the *x*-axis of the video. The UdesMec system is compatible with the majority of SQL statement operators. Initially, this text presents the implementation of encrypted query for addition, subtraction, and multiplication operators.

(1) multiplication operator

The data table *T* has two columns, *A* and *B*, which are encrypted. The column has secret keys $ck_A = \langle x_A, y_A \rangle$ and $ck_B = \langle x_B, y_B \rangle$, respectively. If the product of *A* and *B* yields column *C*, then the column key of column *C* may be represented as $ck_C = \langle x_C, y_C \rangle$. In order to determine the value of *C* based on the values of *A* and *B*, it is necessary to compute C_e and ck_C. The user-side protocols, *mul_x* and *mul_y*, from the Custom Protocol Stack are run to get ck_C in the following manner.

$$ck_C = \langle x_C, y_C \rangle = \langle x_A y_B, x_A + y_B \rangle \tag{5}$$

Perform the *mul_y_*c_e protocol on the cloud database in order to get the value of c_e in the following manner.

$$C_e = A_e B_e mod n \tag{6}$$

Formulas (2) to (4) provide the following equation.

$$C_{key} = x_C \cdot p^{r y_C} = A_{key} \cdot B_{key} \, (mod \; n) \tag{7}$$

Thus, it can be shown that

$$C = C_e \cdot C_{key} = A \cdot A_{key}^{-1} \cdot B \cdot B_{key}^{-1} \cdot A_{key} \cdot B_{key} = A \cdot B \tag{8}$$

(2) Addition and subtraction operators

To perform multiplication, the result column may be acquired by multiplying the values of A_e and B_e. Similarly, ck_C can be produced by multiplying the values of ck_A and ck_B. In order to enable the addition and subtraction operators, the key update operation, denoted as *U*, as well as two auxiliary columns, *S* and *PRI*, must be inserted to complete the operation.

Each row in column *S* has a constant value of 1, whereas each value in *PRI* is a randomly generated prime number. The key update operation *U* may produce a new column, denoted as C, by applying the operation to the input column A and the coordinates $< x_C, y_C >$. The resultant column, denoted as *C*, is obtained by updating column *A* using the operator *U* (*C*=*A*). Additionally, the starting value of ck_C is set to a specified value based on the requirements of the operation. The inverse of S_{key}, denoted as S_e, may be derived using Formula (3). Declare two temporary variables, *j* and *k*, then instruct the user to do the specified actions to acquire the values of *j* and *k*.

$$j = y_S^{-1} \left(y_C - y_A \right) mod \; \varphi \left(n \right) \tag{9}$$

$$k = x_A x_S^j x_C^{-1} \tag{10}$$

Once the values of *j* and *k* are obtained, they are sent to the cloud database. The cloud database does the following operations.

$$C_e = k \cdot A_e \cdot S_e^j \tag{11}$$

Upon doing the computation, it is evident that the value in column *C* is equivalent to the value in column *A*, as shown by the following proof.

$$C = C_e C_{key} = x_A \cdot x_S^j \cdot A_e \cdot \left(S_{key}^{-1} \right)^j \cdot p^{r y_C} = A \tag{12}$$

When it comes to completing the operation, the key update operation *U* provides assistance to other operators. Despite the fact that the key update operation *U* is intended to assure the secure functioning of these operators, it is essential that its own security be maintained.

Consider *C*=*A*+*c*, where *c* is a constant, as with multiplication. Calculate $A + (S \cdot c)$, where *S* is a constant column. To finish the procedure, compute *C*= *A*+*B*. Metadata in the edge node database includes *n* = 3127, *p* = 2, p_1 = 53, and p_2= 59. The coordinate values x_1 = 23 (ck_{x_1} = $\langle$2,2$\rangle$) and x_2 = 194 (ck_{x_2} = $\langle$1,3$\rangle$) are encrypted in Formula (4).

$$V_{key} = g\left(r, (x, y)\right) = x p^{r y \bmod \varphi(n)} \bmod n \tag{13}$$

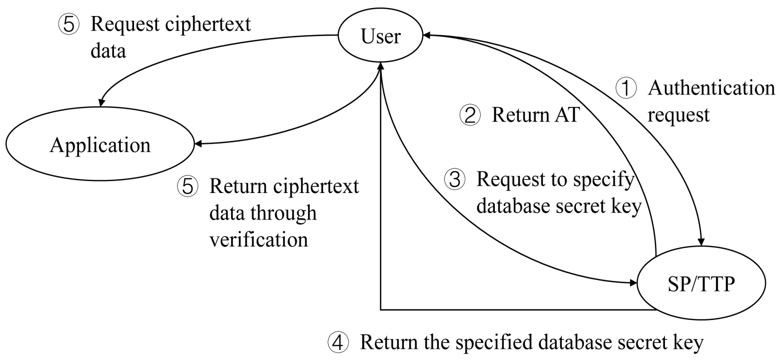
⑤ Request ciphertext data
User
Application
① Authentication request
② Return AT
③ Request to specify database secret key
⑤ Return ciphertext data through verification
SP/TTP
④ Return the specified database secret key

Fig. 4: Flow of ciphertext acquisition.

The ciphertext values, denoted as $v_{e_{x1}}$ = 2739 and $v_{e_{x2}}$ = 806, are derived from the same frame data. The pair of encrypted values is stored inside the cloud service layer. The cloud is not required to acquire the information but simply needs to execute the key update operation based on the provided formula.

$$j = y_S^{-1} \left(y_C - y_A \right) mod\varphi \left(n \right) \tag{14}$$

$$k = x_A x_S^j x_C^{-1} \tag{15}$$

Next, using Formula (16), the ciphertext is combined ($v_{e_{x1}}$ and $v_{e_{x2}}$) in the cloud, resulting in the temporary ciphertext v_e = 834.

$$v_e = v'_{e_{x1}} + v'_{e_{x2}} \tag{16}$$

Using Formula (17), the ciphertext value 834 corresponds to a plaintext value of 217, which matches the result obtained by directly adding the plaintext values.

$$V = D \left(V_e, V_{key} \right) = V_e V_{key} \text{mod} n \tag{17}$$

The transient encrypted data, denoted as v_e= 834, may be kept on the cloud server according to the user requirements, awaiting retrieval by the application layer or for further processing. The cloud is able to perform calculations on encrypted data without the need for decryption, thanks to the use of the homomorphic encryption technique. This capability effectively addresses the majority of data processing and analysis requirements.

2) User Layer Data Application: The data proprietor conducts an analysis of the encrypted data at the application tier, delegating computational and storage responsibilities to the cloud infrastructure. The individual who owns the data performs an in-depth analysis of the encrypted data at the application tier, while entrusting the cloud infrastructure with the execution of computational processes and data storage. After scrutinizing the encrypted data at the application layer, the data owner then delegates the associated computational and storage tasks to the cloud service layer. It just requires ciphertext retrieval and decryption. The edge layer's edge node keeps just its unique database key to prevent privacy data from leaking to other edge nodes if acquired by an attacker. However, the attacker cannot pass TTP user authentication but may obtain edge node database key data. Only TTP-authenticated users may access cloud service layer ciphertext.

A comprehensive model of the whole security integration model that has been implemented in the system application is shown in Fig. 4. A re-identification of the human being is performed, and then the encrypted trajectory data of person 1 is stored on the cloud server.

Video data from each camera is pre-processed and encrypted at the edge node. Data with extracted character features is encrypted. The cloud server collects and processes the secured data, which is then transformed into an encrypted format to yield the secured trajectory information for each entity, encompassing details such as the individual's coordinate location, relative positioning, and associated camera identifier. Once the data has been encrypted and stored on the cloud server, the data proprietor is required to verify their identity with the service provider to acquire the decryption key for the edge node, and subsequently either decrypt or encrypt the cloud data within the surveillance framework to extract the outcome data. The service provider ensures the secure management of decryption keys across all edge nodes, thereby eliminating the need for the data owner to retain all key details at the application layer. Offerings may encompass a centralized login mechanism and tailored application systems catering to diverse user cohorts, alongside hardware resource orchestration and allocation, empowering data owners to oversee edge nodes and IoT devices without the overhead of software application maintenance.

E. Security Proof

In this section, we mainly discuss the possible attack ways against UdesMec system. For the data model, the scope of the attack is shown in Fig. 6. The data base management system (DBMS) on the data owner (DO) side has fields *ck*, *n* and *p*, and the DBMS on the cloud data base (CDB) side has fields *E*(*r*) and V_e. The following describes two possible attack modes against UdesMec system.

1) Chosen-plaintext Attack: A known-plaintext attack (KPA) occurs when the attacker has access to both the ciphertext and associated plaintext values in the cloud database. Chosen-plaintext attack (CPA) allows the attacker to choose a specific plaintext and have the database encrypt it to obtain the corresponding ciphertext. Attackers can choose a specific plaintext to encrypt in CPA, making it more dangerous than KPA and able to extract more information about the key.

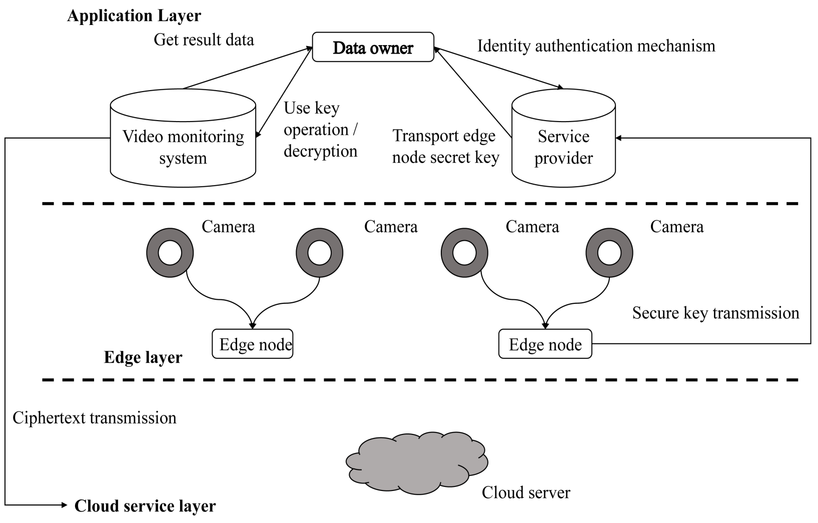

Fig. 5: Application system graph based on security integration model.

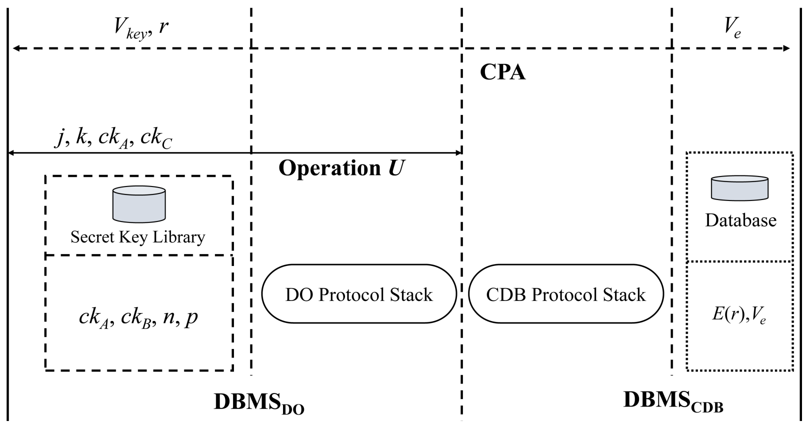

Fig. 6: Possible attack modes and data fields involved in the UdesMec model.

Assuming that the attacker can obtain the ciphertext value V_e stored on the CDB side, and the attacker has obtained the plaintext value V corresponding to its ciphertext V_e, the known-plaintext attack can be realized. The attacker pretends to be an ordinary UdesMec user, uploads the plaintext data value *V*, and observes their ciphertext value V_e, and can carry out chosen-plaintext attack.

Theorem: The column key of column *A* is ck_A=<x_A,y_A>. Given the row key set of column *A*, it contains several V_{key} and p^r corresponding to each row. The attacker cannot calculate y_A in column key ck_A=<x_A,y_A>.

Proof: When an attacker performs a CPA attack, he gets V_e and *V*. According to Formula (3), V_e can be calculated. In the UdesMec data model, the model generates a large prime number *n* through RSA algorithm. Its public key is defined as <*e*, *n*>. If the attacker has the knowledge obtained by CPA attack, he can obtain any ciphertext $V^{'}$ = *V* mod *n* of any plaintext value *V*. Then the attacker's goal is to obtain the private key <*d*, *n*>, $v = {v^{'}}^{d} \text{mod} n$. The attacker can design three steps to achieve this goal. First, a random value *p* is generated, which is called $\hat{P}$. Then, select a row at random $\hat{P}$ and $v = \hat{P}^{\hat{r}}$. Finally, $v^{'} = v^{e}$ is calculated. Assuming that *v* is the row key and the value of p^r is $p^{'}$. The attacker can repeat the second and third steps repeatedly until they have enough data pairs, and then crack UdesMec encryption algorithm according to the column key <*x*=*l*, *y*=*d*>. The attacker gets the value of *d*, and finally get the RSA private key and crack the RSA algorithm. It can be proved that the algorithm strength of UdesMec is the same as that of RSA.

Therefore, it can be concluded that even if the attacker obtains V_{key} and ciphertext column values, it still cannot obtain the plaintext value of the column.

2) Key Update Security: Suppose that the attacker can obtain the ciphertext value and *j*, *k*, ck_A in CDB, as shown in Fig. 13. It can be proved here that the attacker cannot infer ck_C and ck_S from these data.

Theorem: Given *j*, *k* and ck_A of key update operation, it can be guaranteed that there is no less than $\varphi(\varphi(n))$ possible combinations of ck_C and ck_S.

Proof: After the attacker has *j*, *k* and ck_A, the calculation is constructed first as follows.

$$j = y_S^{-1} \left(y_C - y_A \right) \text{mod} \varphi \left(n \right), \forall \hat{y}_S \in \left[1, \varphi \left(n \right) \right] \tag{18}$$

$$gcd \left(\hat{y}_S, \varphi \left(n \right) \right) = 1, \exists \hat{y}_C = y_A + p \hat{y}_S \text{mod} \varphi \left(n \right) \tag{19}$$

Since y_s and $\varphi(n)$ are mutually prime numbers, it can be concluded that there are more than $\varphi(\varphi(n))$ possibilities for $\hat{y}_S$ and $\hat{y}_C$. And $\varphi(n)$ is the Euler function, which is generated by two large prime numbers P_1 and P_2. It is difficult for attackers to infer the results, and its security is guaranteed by RSA.

IV. Experiments

The system with the suggested security model (UdesMec) stores and processes user privacy data in the form of ciphertext. We primarily assess the system's feasibility and then confirm that the proposed system can ensure data security and privacy. The experiment employs a cloud-based server equipped with an Intel Xeon processor e3-1270 operating at 3.40GHz for performing encrypted data computations, in conjunction with four separate personal computers acting in the capacity of edge computing units.

These machines are equipped with an Intel Core i5-6200U@2.3GHz CPU and an NVIDIA GeForce GTX 760 2GB graphics card. The camera records videos at a resolution of 640×480 and a maximum frame rate of 3 frames per second. We created a system for tracking and predicting human trajectories in a multi-camera environment. The system's prototype program is coded in Python.

A. Analysis of Communication Cost of System

This section is principally dedicated to scrutinizing the communicational load that comes with the execution of a security protocol. Furthermore, an analysis is conducted to compare the proposed encryption technique with other state-of-the-art techniques. The methodologies for comparison are detailed subsequently.

AggBPE [21]: The ciphertext is represented as $v_e = g^m r^n \bmod n^2$ and stored as two distinct ciphertext values, denoted as x_i and ${x_i}^2$. Suppose the experimental parameters encompass 1024 bits and the number of bits squared equals 2048, the necessary storage capacity for data generated by *N* IoT devices is 4096N bits.

LPDA [22]: The algorithm combines the values of x_i and ${x_i}^2$ into a ciphertext represented as $c_{is} = \left[1 + n \times a_j \times \left(x_i \times \alpha_0 + x_i^2\right)\right] \bmod n^2$. As a result, the LPDA stores a total of 2048×*N* bits.

TPCS [22]: The approach employs a pseudonym and a Paillier password technique to guarantee the confidentiality of the processors. It further establishes three authentication techniques to guarantee that only the genuine processor can successfully carry out the authentication procedure.

Table III presents the communication overhead of the three kinds, which varies depending on the quantity of equipment. The security paradigm outlined in this section offers distinct benefits compared to current encryption methods in terms of communication overhead.

B. Analysis of System Performance Execution Cost

This section focuses on data encryption overhead's effect on model application performance. Develop TPCS, LPDA, and AggBPE prototype programs to start the experiment. To assure experiment fairness, we standardized all model parameters to the same number of bits and encrypted the plaintext value. The average value of the 50-run experiment is taken. The experimental findings are in Figures 7 and 8. The suggested method and three alternative models perform worse than plaintext operations, according to experiments.

TABLE III: ANALYSIS OF NETWORK COMMUNICATION BURDEN IN ENCRYPTED SCHEMES

IoT device number	AggBPE	LPDA	TPCS	UdesMec
0	0	0	0	0
100	0.45	0.28	0.37	0.19
200	0.75	0.46	0.69	0.28
300	1.21	0.58	1.12	0.39
400	1.58	0.76	1.38	0.41
500	2.08	1.1	1.76	0.5
600	2.43	1.22	2.18	0.56
700	2.78	1.43	2.34	0.69
800	3.18	1.54	2.83	0.73
900	3.58	1.69	3.07	0.93
1000	4.05	2.01	3.29	1.1

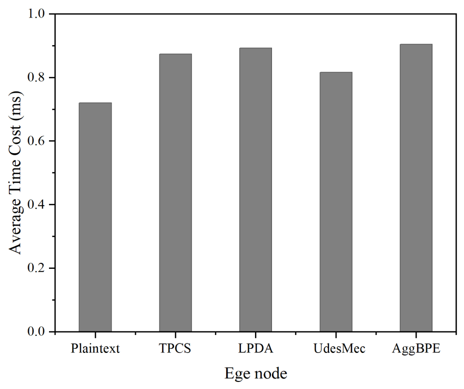

Fig. 7: Average encryption cost per frame.

This study presents a ciphertext model with reduced computational cost and better performance than the other three solutions at the edge and in the cloud.

The model processes structured data in a unified edge computing environment. To determine how much the ciphertext model costs for various data processing tasks, we examine its average time cost under common operators. The results are in Table IV. It illustrates how the three operations generally operate slower on ciphertext than plaintext. In the encryption model, adding and subtracting are multiplications. The encryption model includes a key updating mechanism that slows the procedure. Addition and subtraction don't tax modern computer hardware. System overhead makes the ciphertext model unacceptably sluggish, even if it adds and subtracts 50 times faster than plaintext in 0.0087 ms and 0.0084 ms per frame.

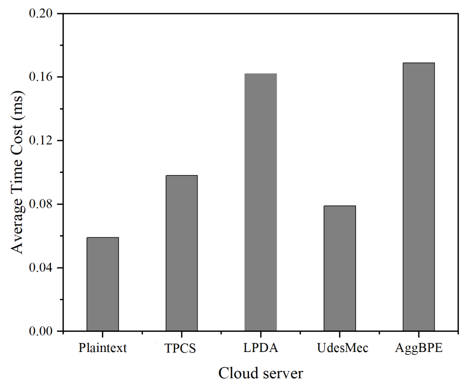

Fig. 8: Average encryption cost per frame.

TABLE IV: THE MULTIPLE OF OPERATION COST OF DATA CIPHERTEXT OPERATOR RELATIVE TO PLAINTEXT COST

Operation	Multiply	Add	subtract
Average Time Cost	9.86×	58.19×	51.86×

C. *System Feasibility Assessment*

This section is dedicated to evaluating the system's capability when integrated with a security framework, focusing on its ability to ensure a processing time within acceptable limits. The objective here is to illustrate the feasibility of the model presented in this study within a practical edge computing scenario. In the experimental setup, a network of nine cameras is utilized to simulate a camera network, where the cameras continuously collect raw video footage, capturing images at a frequency of three frames per second.

The procured images are then processed to identify and catalog the motion and color profile of individuals present within the footage by leveraging the YOLO algorithm [23]. The interaction between the cloud service and edge layers is facilitated through socket communication, employing TCP/IP protocol for data exchange. As soon as the frame data is procured, it is promptly dispatched to the cloud server, which performs instantaneous encrypted data processing on the received information to ascertain encrypted details regarding the person's relative location. The calculation formula for the relative position *rp* of the person along the x-axis of the camera in the experiment is based on the assumption that the camera resolution has a width of *Camw* and a length of *Camh*.

$$rp = \frac{(x_1 + x_2)/2 - Camw/2}{Camw/2} \tag{20}$$

The formula shown above indicates that in order to obtain the relative position *rp*, it is essential to perform a multiplication operation that is not an integer on the encrypted value. It is important to note that the data included inside the data model of the system can only be calculated within the boundaries of integer values. Therefore, in the event that the user demands the final unencrypted result when performing operations with decimal numbers, a compromise technique must be used in order to maintain the encrypted value of x_1+x_2 and decrypt it at the application layer. In scenarios that occur in the real world, code designs may be modified to accommodate certain computing requirements. For the purpose of validating the latency of the system, the experiment uses two different scenarios.

TABLE V: TWO CONFIGURATIONS CONSTRUCTED FOR EXPERIMENTAL EVALUATION

Scene	Camera	Cloud Server	Edge Node
No. 1	2	2	2
No. 2	9	2	4

Figures 9 and 10 show the output of the experiment that was conducted. Within the first scenario, the typical amount of time required to process each frame is around 0.35 seconds, while the greatest amount of delay is 0.37 seconds. This occurs due to the fact that the amount of data gathered rises proportionally with the number of cameras,

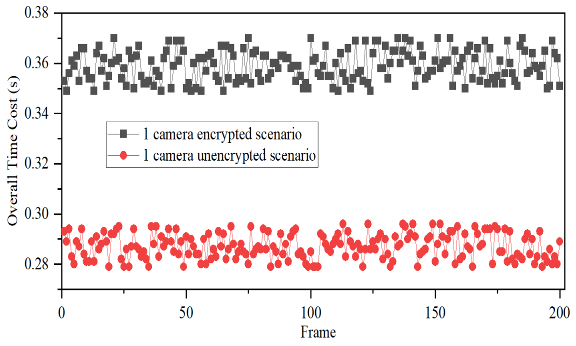

Fig. 9: Delay analysis in real time processing (1 camera).

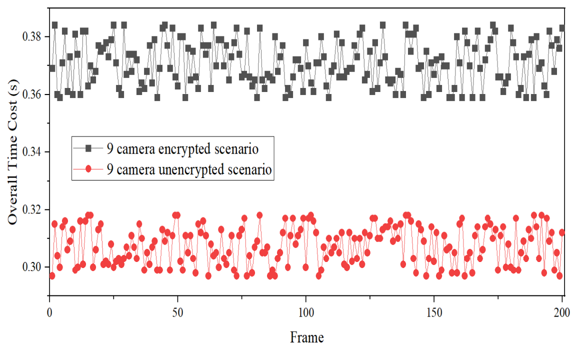

Fig. 10: Delay analysis in real time processing (9 camera).

and the cloud server and edge nodes need more time in order to analyze this data.

Nevertheless, the edge nodes bear the computing burden resulting from the additional cameras, while the cloud server is simply required to handle uncomplicated formatted data. In the second situation, the average processing time per frame is only increased by 0.02 seconds compared to the scenario with just one camera. For video with a frame rate of 3 frames per second (FPS), it takes about 0.334 seconds to create frame data. In both instances, the security integrated system may show the character track data to the user in near real time, before the current time point. Employing the edge computing approach within an extensive network of dispersed cameras allows for the deployment of additional edge nodes, which augments the performance of video data processing. Furthermore, there is potential to augment the cloud server capabilities or refine its settings to ensure the proficiency of aggregated data processing, thereby ensuring the system's capacity for expansion.

D. *Query Performance*

Through the use of TPC-H (TPC Benchmark-H) [14], the primary objective is to assess the capacity of certain queries to provide decision assistance. The benchmark includes the simulation of the database operation in the decision support system, the testing of the response time of complicated queries in the database system, and the use of the number of queries run per hour as the measurement index. In the experiment, TPC-H is used to test the system in order to conduct a study of the performance of queries. All query operators that are routinely used as well as sophisticated queries are included in the TPC-H test. Generally speaking, TPC-H indicates that the database is capable of supporting normal usage as well as dealing with certain difficult business situations. MONOMI [24] and CryptDB [14] are two others that have been extensively researched algorithms for encrypting database models. This comparison is made with the aforementioned algorithms.

All the claims of TPC-H can be implemented accurately in the experiment. Tables VI to VIII show the ratio between the execution time of the 22 sentences of UdesMec running TPC-H and the execution time of the plaintext query. The clauses Q4, Q11, Q12, Q13, Q16, and Q21 are absent from UdesMec since they do not pertain to procedures using ciphertext. The CryptDB and MONOMI models are incapable of handling queries Q13, Q15, and Q16. UdesMec demonstrates superior efficiency compared to CryptDB in terms of execution time for most statements, approaching that of MONOMI.

V. Conclusion

This paper presents a comprehensive model for encrypting, storing, and processing data in edge computing. The model is designed to handle the large number of edge nodes and the different characteristics of the network. The use of secret sharing and homomorphic encryption techniques is employed to handle IoT terminal data in ciphertext, as a result of the constrained computational power and restricted storage capacity of the edge nodes. This transfers a substantial part of the computational burden to the cloud service layer. The approach minimizes the computational and storage burden on edge computing nodes by transferring the majority of the processing to the cloud service layer. The approach minimizes the computational and storage burden on edge computing nodes, while guaranteeing efficient gathering and processing of device data via encryption techniques. The security integration model incorporates an authentication technique at the application layer to safeguard the privacy of data owners. Providers possess the capability to amalgamate their server capacities to render services to diverse user communities within an integrated technological framework, utilizing an access control protocol to confine user access to personal data. The experimental analysis quantified the expenses related to the models communicative and computational aspects, and appraised its viability for implementation in practical scenarios.

Our work not only contributes to the broader discourse on the transformative potential of IoT and 5G in healthcare but also directly addresses the pressing challenges associated with processing and utilizing massive amounts of health-related data at the network edge, ultimately enhancing the ability of healthcare providers to predict, prevent, and manage complex medical crises more effectively.

References

[1] S. B. Basapur, B. Shylaja *et al.*, "Constraints-relaxed functional dependency based data privacy preservation model." *Engineering Letters*, vol. 31, no. 1, pp. 19–34, 2023.

[2] Linzhi, Jiang, Liqun, Chen, Thanassis, Giannetsos, Bo, Luo, Kaitai, and Liang, "Toward practical privacy-preserving processing over encrypted data in iot: An assistive healthcare use case," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 177–10 190, 2019.

[3] A. Murugesan, B. Saminathan, F. Al-Turjman, and R. L. Kumar, "Analysis on homomorphic technique for data security in fog computing," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 9, p. e3990, 2021.

[4] G. Srivastava, C. W. Lin, D. Pamucar, and S. Kotsiantis, "Editorial: Applications of fuzzy systems in data science and big data," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 1, pp. 1–3, 2021.

[5] H. Miyajima, N. Shigei, H. Miyajima, and N. Shiratori, "Secure learning systems using vertically partitioned data with iot," *IAENG International Journal of Computer Science*, vol. 49, no. 1, pp. 61–68, 2022.

[6] J. Liu, X. Wang, S. Shen, G. Yue, S. Yu, and M. Li, "A bayesian q-learning game for dependable task offloading against ddos attacks in sensor edge cloud," *IEEE Internet of Things Journal*, vol. 8, no. 9, pp. 7546–7561, 2020.

[7] V. Stephanie, M. Chamikara, I. Khalil, and M. Atiquzzaman, "Privacy-preserving location data stream clustering on mobile edge computing and cloud," *Information Systems*, vol. 107, p. 101728, 2022.

[8] S. Shen, H. Ma, E. Fan, K. Hu, S. Yu, J. Liu, and Q. Cao, "A non-cooperative non-zero-sum game-based dependability assessment of heterogeneous wsns with malware diffusion," *Journal of Network and Computer Applications*, vol. 91, pp. 26–35, 2017.

[9] H. Zhou, S. Shen, and J. Liu, "Malware propagation model in wireless sensor networks under attack–defense confrontation," *Computer Communications*, vol. 162, pp. 51–58, 2020.

[10] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the internet of things: Research issues and challenges," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2188–2204, 2018.

[11] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *The journal of supercomputing*, vol. 76, no. 12, pp. 9493–9532, 2020.

[12] H. Liu, P. Zhang, G. Pu, T. Yang, S. Maharjan, and Y. Zhang, "Blockchain empowered cooperative authentication with data traceability in vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4221–4232, 2020.

[13] D. Zheng, G. Shen, X. Cao, and B. Mukherjee, "Towards optimal parallelism-aware service chaining and embedding," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2063–2077, 2022.

[14] J. Feng, F. R. Yu, Q. Pei, J. Du, and L. Zhu, "Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 4321–4334, 2020.

[15] M. Hartmann, U. S. Hashmi, and A. Imran, "Edge computing in smart health care systems: Review, challenges, and research directions," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 3, p. e3710, 2022.

[16] J. Srinivas, A. K. Das, N. Kumar, and J. J. Rodrigues, "Tcalas: Temporal credential-based anonymous lightweight authentication scheme for internet of drones environment," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6903–6916, 2019.

[17] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," *IEEE systems journal*, vol. 12, no. 2, pp. 2039–2042, 2016.

[18] N.-W. Lo and J.-L. Tsai, "An efficient conditional privacy-preserving authentication scheme for vehicular sensor networks without pairings," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1319–1328, 2015.

[19] J. Kim, S. Camtepe, W. Susilo, S. Nepal, and J. Baek, "Identity-based broadcast encryption with outsourced partial decryption for hybrid security models in edge computing," in *Proceedings of the 2019 ACM Asia conference on computer and communications security*, 2019, pp. 55–66.

[20] L. Jing, T. Qiu, C. Wen, K. Xie, and F. Q. Wen, "Robust face recognition using the deep c2d-cnn model based on decision-level fusion," *Sensors*, vol. 18, no. 7, p. 2080, 2018.

[21] X. Li, F. Li, and M. Gao, "Flare: A fast, secure, and memory-efficient distributed analytics framework," *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1439–1452, 2023.

[22] M. Gheisari, H. E. Najafabadi, J. A. Alzubi, J. Gao, G. Wang, A. A. Abbasi, and A. Castiglione, "Obpp: An ontology-based framework for privacy-preserving in iot-based smart city," *Future Generation Computer Systems*, vol. 123, pp. 1–13, 2021.

[23] Y. Zhang, R. Wang, M. S. Hossain, M. F. Alhamid, and M. Guizani, "Heterogeneous information network-based content caching in the internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 216–10 226, 2019.

[24] L. Wiese, T. Waage, and M. Brenner, "Clouddbguard: A framework for encrypted data storage in nosql wide column stores," *Data & Knowledge Engineering*, vol. 126, p. 101732, 2020.

TABLE VI: THE PROPORTION OF EXECUTION TIME WITH PLAINTEXT (Q1∼Q7).

Methods	Q1	Q2	Q3	Q4	Q5	Q6	Q7
CrptDB	38.17X	2.4X	4.6X	3.5X	3.45X	6.7X	2.8X
MONOMI	2.6X	2.5X	2.4X	2.1X	1.9X	2.2X	1.7X
UdesMec	11.29X	1.9X	1.9X	NULL	1.89X	15.79X	1.68X

TABLE VII: THE PROPORTION OF EXECUTION TIME WITH PLAINTEXT (Q8∼Q14).

Methods	Q8	Q9	Q10	Q11	Q12	Q13	Q14
CrptDB	5.6X	4.8X	4.94X	5X	4.95X	NULL	6.1X
MONOMI	2.42X	2.45X	2.5X	2.54X	2.5X	NULL	2.51X
UdesMec	2.48X	2.4X	2.45X	NULL	NULL	NULL	2.43X

TABLE VIII: THE PROPORTION OF EXECUTION TIME WITH PLAINTEXT (Q15∼Q22).

Methods	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
CrptDB	NULL	NULL	5.65X	59.94X	5.64X	6.4X	NULL	4.8X
MONOMI	NULL	NULL	2.51X	2.56X	2.5X	2.61X	NULL	1.9X
UdesMec	5X	NULL	2.45X	22.14X	2.45X	2.48X	NULL	1.74X

Qi Li was born in Jiangsu, China, in 1987. He began his academic journey at the prestigious Northwestern Polytechnical University, where he pursued his passion for engineering and technology. In 2015, he completed his Master of Science degree in Communication Engineering, demonstrating his expertise in the field. He continued his commitment to higher education and research. He remained at Northwestern Polytechnical University to continue his studies. In 2019, he received his Doctor of Philosophy degree in Computer Science, marking a significant milestone in his academic career. He is currently employed at Shaoxing University, where he contributes his knowledge and skills to the academic community. His research interests include graph neural networks and network security.

Jian Huang was born in Shaoxing, China, in 1967. He received his Bachelor of Agriculture degree in Economic Forestry from Zhejiang A&F University, located in Hangzhou, Zhejiang Province, in 1987. He received the Master of Economics degree in National Economics by Zhejiang University,a prestigious university in Hangzhou, Zhejiang Province, in 2006. His research interests include economics,image processing, machine learning,and data mining.

Sihan Li was born in Wenzhou, China in 2002. He completed his undergraduate studies at the University of Waterloo, Canada, in 2020, where he received a Bachelor of Arts degree. He continued his education at the University of Toronto, Canada, where he received a Master of Science degree in Computer Science in 2023. Throughout his academic journey, he has developed a strong interest in the fields of machine learning and knowledge graphs.

Chenze Huang was born in 1970 in Wenzhou, China. he received his M.Sc. degree in communication engineering from Shanghai Maritime University in 1995, and he is currently pursuing his Ph.D. degree at Northwestern Polytechnical University. His research interests include complex networks, network security, and so on.

Date of modification: July 5, 2024

Brief description of the changes: Revise the biographies of Qi Li and Huang Jian.