sensors

Article

Metadata-Private Resource Allocation in Edge Computing Withstands Semi-Malicious Edge Nodes

Zihou Zhang [1,2], Jiangtao Li [1,2], Yufeng Li [1] and Yuanhang He [3,*]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; zzh@shu.edu.cn (Z.Z.); lijiangtao@shu.edu.cn (J.L.); liyufeng_shu@shu.edu.cn (Y.L.)
[2] Science and Technology on Communication Security Laboratory, Chengdu 610041, China
[3] No. 30 Research Institute of China Electronics Technology Group Corporation, Chengdu 610041, China
* Correspondence: kevin0319@163.com

Abstract: Edge computing provides higher computational power and lower transmission latency by offloading tasks to nearby edge nodes with available computational resources to meet the requirements of time-sensitive tasks and computationally complex tasks. Resource allocation schemes are essential to this process. To allocate resources effectively, it is necessary to attach metadata to a task to indicate what kind of resources are needed and how many computation resources are required. However, these metadata are sensitive and can be exposed to eavesdroppers, which can lead to privacy breaches. In addition, edge nodes are vulnerable to corruption because of their limited cybersecurity defenses. Attackers can easily obtain end-device privacy through unprotected metadata or corrupted edge nodes. To address this problem, we propose a metadata privacy resource allocation scheme that uses searchable encryption to protect metadata privacy and zero-knowledge proofs to resist semi-malicious edge nodes. We have formally proven that our proposed scheme satisfies the required security concepts and experimentally demonstrated the effectiveness of the scheme.

Keywords: Internet of Things; edge computing; privacy-preserving; resource allocation

check for updates

Citation: Zhang, Z.; Li, J.; Li, Y.; He, Y. Metadata-Private Resource Allocation in Edge Computing Withstands Semi-Malicious Edge Nodes. *Sensors* **2024**, *24*, 2989. https://doi.org/10.3390s24102989

Academic Editor: Weizhi Meng

Received: 1 April 2024
Revised: 28 April 2024
Accepted: 6 May 2024
Published: 8 May 2024

1. Introduction

Edge computing offers significantly enhanced computing power by outsourcing end-device tasks to nearby edge nodes or devices with additional computing resources. The reduced transmission latency in edge-based solutions, as compared with traditional cloud-based alternatives, underscores its efficiency. As intelligent requirements for terminal devices continue to evolve, the limitations of traditional cloud computing become evident, especially in handling a multitude of delay-sensitive and computationally intensive tasks. This shift has given rise to the emergence of edge computing [1]. The application scope of edge computing extends widely across industry and manufacturing, agriculture, healthcare, urban management, and environmental protection [2–5], leveraging the transformative potential of the Internet of Things (IoT).

The optimization of task offloading and resource allocation is the core problem of edge computing [6], and the scheme based on a smart gateway [7] is considered a promising solution. Specifically, the end device will send the task to the smart gateway, which will carry out measures such as refining and filtering the data according to the application requirements at the smart gateway end, and finally assign the task to the appropriate edge node for processing, in which the gateway should take into account various factors, such as the type of the task, the time sensitivity, and the computational complexity. In order to realize such a system, metadata defining the factors required for the task need to accompany the task data when they are sent [8], and these metadata can help the gateway to make decisions when allocating resources [9]; e.g., time sensitivity points out that the task needs to be processed in real-time, and computational complexity points out that the task needs more computational resources.

The metadata of edge computing contain sensitive information, such as when a user publishes a task, what kind of resource they request, and how much computation resource they need. Unprotected metadata will expose the end device's private information; e.g., a malicious adversary can launch a side-channel attack by observing how the gateway performs the resource allocation to obtain the user's privacy information [10]. Some research has shown that it is possible to get a detailed and private picture of a person's contacts and interests just by analyzing metadata, and it is technically much more accessible to search massive amounts of metadata than to listen in on millions of phone calls [11]. In recent years, some researchers have focused on the metadata privacy in secure messaging [12,13], however, in the area of edge computing, we still face the challenge of lower computation resources on the end devices and lower bandwidth requirements.

A possible method to achieve metadata privacy in edge computing is for the end device to encrypt the metadata using the public key of the gateway and send it to the gateway, which decrypts it and then completes the resource allocation. Although this scheme can resist malicious eavesdroppers, the gateway can still obtain information about the metadata, and once the gateway is corrupted, the attacker can still obtain the user's privacy. To solve the above problem, a scheme based on searchable encryption has been proposed [14]; by using searchable encryption, the smart gateway can check whether the received encrypted metadata contains a specific keyword without learning any metadata-related content.

It is challenging to run a strong cyber defense system because of the relatively limited computing power of the edge nodes. In addition, the security setup of the edge nodes is also more prone to errors than the cloud [15]. As a result, edge nodes are more easily corrupted and become semi-malicious edge nodes. A semi-malicious edge node is defined as an edge node that will maintain system consistency but will behave maliciously to weaken the privacy of the system. However, the above scheme [14] lacks the consideration of semi-malicious edge nodes, which may allow the gateway to perform equality test searches for specific keywords if one or more edge nodes are corrupted in the group generation phase.

1.1. Our Contribution

Existing resource allocation schemes for metadata privacy protection often lack the consideration of malicious smart gateways and semi-malicious edge nodes. In order to address these challenges, we propose a resource allocation scheme for metadata privacy protection based on searchable encryption. This scheme is valuable in edge computing environments with limited computational resources and bandwidth, such as the Internet of Vehicles. The contributions of this work are as follows:

First, edge nodes can generate a group encryption public key by interacting with each other, and subsequently, for a set of selected keywords, each edge node can compute a partial trapdoor, and the smart gateway can aggregate these partial trapdoors to generate a complete trapdoor. End devices need to encrypt the corresponding metadata using the group public key when sending data and send the encrypted metadata to the gateway together with the task data. The gateway uses a complete trapdoor to determine whether the encrypted metadata contains the specific keyword and finally allocates resources according to the predetermined policy. In the process, the gateway does not learn any keyword-related content. Moreover, our proposed scheme achieves constant message expansion, i.e., the ciphertext length of the encrypted keyword does not increase with the increase in edge nodes.

Second, we use extractable zero-knowledge proofs to resist corrupted edge nodes. Specifically, in the group generation phase, each edge node must provide the corresponding proofs of the selected random parameter, and the parameter without the corresponding proofs will not be added to the corresponding computation of the group public key, which prevents semi-malicious edge nodes from enabling the gateway to perform an equality test search on a specific keyword by generating a malicious parameter in the group generation phase.

Finally, we identify the security concepts of a metadata-privacy resource allocation scheme and perform the corresponding security analysis. Our scheme captures important security properties such as user privacy, full key compromise resistance, and semi-malicious edge node resistance. In addition, we give a formal proof of how our proposed scheme satisfies the defined security concepts. In addition, we experimentally evaluate the performance of the proposed scheme.

1.2. Related Work

The allocation of limited resources is the most critical component of an edge computing system, and its efficiency directly affects the whole system's performance. Recently, many resource allocation strategies and algorithms have been proposed to realize resource allocation and management in edge computing [16–18]. The use of smart gateways [7] is considered a promising solution for edge computing resource allocation, where the smart gateway can utilize system resources better by deciding the time and type of data to be sent based on application feedback. However, it is worth noting that user privacy is not considered in Aazam et al.'s scheme [7].

Lu et al. [19] proposed a lightweight privacy-preserving data aggregation scheme based on homomorphic encryption and the Chinese Remainder Theorem in response to the limitation that existing schemes only support homogeneous devices and experimentally proved the scheme's efficiency. However, the scheme does not consider identity privacy and traceability. In order to reduce privacy leakage in edge computing, Lyu et al. [20] proposed a new privacy-preserving data aggregation scheme based on homomorphic encryption and differential privacy, which guarantees the obliviousness of the aggregator. However, the scheme does not consider the edge node that may steal the user's location privacy. Zhang et al. [21] proposed a resource allocation scheme that can resist malicious edge nodes using blockchain and trust computing. However, since the scheme uses symmetric searchable encryption, it cannot generate complete trapdoors cooperatively. Thus, the scheme is difficult to implement to protect metadata privacy. Kong et al. [22] designed a task offloading strategy based on a multifeedback trust mechanism, which can identify malicious nodes using trust assessment and improve system operation efficiency using the clustering algorithm. Zhou et al. [23] proposed a lightweight, verifiable privacy-preserving outsourcing matching pattern protocol that prevents collusion between the cloud and malicious receivers/senders. Li et al. [24] proposed a verifiable edge outsourcing computing scheme based on blockchain, which protects the privacy of the results. Wang et al. [25] designed a privacy-preserving computational offloading framework based on vehicular fog computing and smart contracts and experimentally demonstrated the effectiveness of the framework. However, we note that all the above schemes lack consideration of metadata privacy issues.

In order to protect the privacy of metadata in edge computing, Zhang et al. [14] proposed a scheme based on searchable encryption that achieves constant message expansion and full key compromise resistance. Angel et al. [26] proposed a new resource allocation scheme that allocates resources to a group of clients and does not disclose to the clients whether there are other clients receiving the resources. Beams et al. [10] proposed a privacy-preserving packet scheduling scheme that prevents a malicious adversary from obtaining the victim's private information by launching a side-channel attack and observing how the switch schedules packets. Ahmad et al. [27] proposed a new voice communication scheme that enables metadata privacy-preserving voice communication over a completely untrusted infrastructure. However, all the above privacy-preserving resource allocation schemes do not consider malicious edge nodes.

Barman et al. [12] proposed a metadata privacy messaging system based on differential privacy and Private Information Retrieval (PIR) [28] that can support asynchronous messaging chats and can run on untrusted servers. Tests of a prototype system show the effectiveness of the scheme. Jiang et al. [13] implemented a scalable metadata-private messaging system based on hardware enclaves that achieves resistance to malicious clients. Experiments show that the scheme has a lower extra cost and higher efficiency. Cai et al. [29] designed a metadata-hiding data analytic system called Vizard using the distributed point function, which allows data owners to share their data in a privacy-preserving manner, and the experimental results proved the practicality of the scheme. Langowski et al. [30] designed a metadata-private anonymous broadcasting scheme called Trellis based on boomerang encryption and anonymous routing tokens that hides all network-level metadata and supports horizontal scalability. However, these solutions are designed for large network systems and are difficult to apply directly to edge systems with limited computational and bandwidth resources.

2. Background

2.1. System Architecture

The edge computing system used in our proposed scheme comprises five types of entities: trusted authority (TA), end device, gateway, edge node, and cloud, as shown in Figure 1.

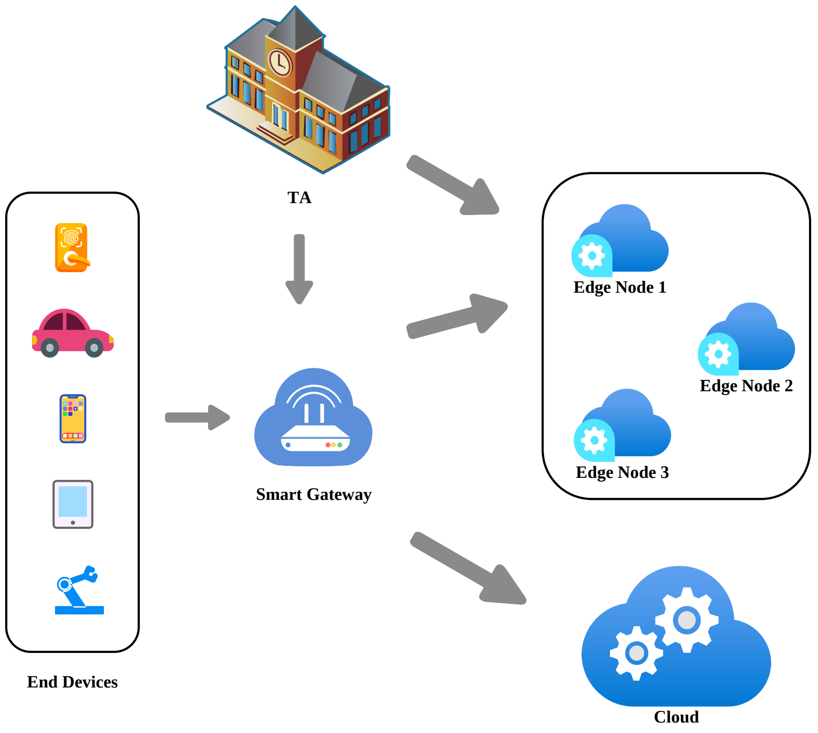

Figure 1. System architecture.

- TA provides registration services for edge nodes and generates system parameters. In our scheme, TA is considered to be entirely trustworthy.
- End devices with limited computing and storage capabilities are users of the edge computing system and need to accomplish various tasks with the help of edge nodes.
- The gateway does the forwarding process for the received task and finds the appropriate edge nodes or cloud. Note that since the resource allocation is done by the gateway and not by the end device itself, only one group exists at every moment. In our scheme, the gateway is considered to be honest but curious [31], meaning that the gateway will honestly follow the protocol to complete the forwarding job but will try to violate the privacy of the end device.
- Edge nodes have a relatively limited computational and storage capacity [32,33], and thus generally need to collaborate to handle tasks. In our scheme, edge nodes may be compromised to become semi-malicious edge nodes. A semi-malicious edge node is defined as an edge node that will maintain system consistency but will behave maliciously (we call it an elimination attack) to weaken the privacy of the system.

The elimination attack is defined as follows: Suppose an edge node with identity ID_i is corrupted by the smart gateway in the group setup phase. The elimination attack may "eliminate the necessity authorization" via edge node ID_j. In other words, with the help of the corrupted edge node ID_i, the gateway is able to aggregate to a valid trapdoor for the message m_l without the permission of edge node ID_j. The attack works as follows: (1) In the group public key generation phase, edge node ID_i broadcasts $r_i - r_j$ instead of $r_i = x_iP$; (2) In the trapdoor generation phase, edge node ID_i generates the trapdoor with x_i as normal; (3) In the trapdoor generation phase, the gateway does not include the partial trapdoor w_j of the edge node ID_j in the aggregation process, i.e., it computes $T_{m_l} = \sum_{i \neq j} w_i$ which turns out to be a valid trapdoor since r_j has been canceled out in the group generation phase.

- The cloud is considered to have a sufficient computing and storage capacity, and some tasks that are not time-sensitive and require significant computing and storage resources will be sent to the cloud for processing.

2.2. Design Goals

Considering the security risks posed by malicious behaviors to the resource allocation process in edge computing, our security goals must satisfy the following points.

1. *Metadata privacy*: The gateway should be able to check whether a legitimate keyword is included in the encrypted metadata without learning the content of the metadata. When the gateway has no complete trapdoor, it will be equivalent to an eavesdropper. For an eavesdropper, indistinguishability is required. That is, the eavesdropper cannot distinguish whether two encrypted keywords are the same. In other words, one of the given two keywords is randomly chosen to be encrypted, and the attacker cannot distinguish which keyword the encrypted keyword corresponds to.
2. *Full key compromise resistance*: Even if the attacker corrupts all the edge nodes in the system, the attacker still cannot decrypt the previously encrypted metadata.
3. *Semi-malicious edge node resistance:* Semi-malicious edge nodes cannot disrupt other nodes' participation in group public key generation via an elimination attack.

2.3. Bilinear Pairing

Suppose that $\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups of prime order q and that the generator of $\mathbb{G}_1$ is P. A pair $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a bilinear pairing if it satisfies

1. Bilinear: $\hat{e}(uP, vP) = \hat{e}(P, P)^{uv}$, where $u, v \in \mathbb{Z}_q^*$.
2. Non-degenerative: For each $x \in \mathbb{G}_1$, there exists exactly one $y \in \mathbb{G}_1$ with $\hat{e}(x, y) = 1$.

2.4. Identity-Based Cryptosystem

The concept of the identity-based cryptosystem (IBC) was first proposed by Shamir [34]. In IBC, the user's public key is their identity, thus reducing the certificate management overhead in q traditional Public Key Infrastructure-based (PKI-based) cryptosystem [35]. Our scheme is designed in IBC [34], where the edge nodes use their identity as the public key, thus reducing complex problems such as certificate updates and revocations.

2.5. Searchable Encryption

Searchable encryption [36] is a cryptographic primitive that allows encrypted data to be searched by a user with a trapdoor without revealing information about the data. Searchable encryption mainly consists of symmetric searchable encryption (SSE) [37] and public key encryption with keyword search (PEKS) [38], which have different focuses on functionality and performance and are used to meet the requirements in different scenarios. The data owner, trapdoor generator, and decryptor in SSE must share a key, while PEKS allows anyone who knows the entity's public key to generate the encrypted keyword. Since PEKS enables smart gateways to determine whether a keyword is present in encrypted metadata without learning its content, PEKS enables metadata privacy protection against the malicious gateway.

In an edge computing system, it is reasonable to assume that a group of edge nodes that agree on the same privacy terms will decide what information can be leaked to the gateway. This means that these edge nodes should generate a trapdoor cooperatively. However, in current searchable encryption schemes for multiple recipients [39], if an attacker obtains one of the recipients' private keys, they can generate any trapdoor to recover the keyword. Although this problem can be solved using a thresholding scheme, it also means that the length of the ciphertext corresponding to the keyword will grow linearly with the number of receivers. Our proposed scheme achieves constant message expansion, i.e., the length of the encryption keyword does not increase with the number of edge nodes, which contributes to the scalability of the scheme.

2.6. Extractable Zero-Knowledge Proof

A zero-knowledge proof [40] allows the prover to prove a statement's truth without revealing additional information to the verifier. The proofs of knowledge used in this work need to be extractable, meaning that there exists an extractor with oracle access to the prover that can output the witness of a statement. We adopt the notation proposed by Camenisch et al. [41] to represent zero-knowledge proofs of discrete logarithms. Specifically, we use $PK\{(x, y, z) : h = aP + yc \wedge h' = xP + zc\}$ to denote the zero-knowledge proofs of x, y, z that can make the equations $h = aP + yc$ and $h' = xP + zc$ hold. If the extractor can extract the witness without rewinding, it is called online-extractable [42], and in this paper, witnesses that need to be online-extractable will be underlined.

3. Proposed Scheme

3.1. Preliminary Definitons

In this section, the relevant definitions of the proposed scheme will be given, and our proposed scheme consists of a PPT algorithm (Init, Reg, GG, KG, ME, TA, RA), which works as follows:

- Initialization. $Init(1^k) \rightarrow (\mathrm{s}, \Delta)$: The randomization algorithm Init takes the security parameter k as input and outputs the system master secret s and public system parameters Δ. The security parameter k is a parameter used to ensure the security of the scheme, usually including the secret key length and the output length of the hash function, etc.; this parameter needs to be chosen based on the tradeoffs between the desired security level and the performance requirements. The system parameters need to be public.
- Registration. $Reg(ID_i) \rightarrow (sk_i)$: The deterministic algorithm Reg is run by TA and takes as input the identity ID_i of the edge node and outputs the private key corresponding to ID_i.
- Group Generation. $GG(ID_1, ID_2, \ldots, ID_k) \rightarrow (GID, E)$: A group of edge nodes $ID_1, \ldots, ID_k$ with total number k that want to build an edge computing system can run the algorithm to generate the corresponding group identity GID and group public key E.
- Keyword Generation. $KG(ID_1, ID_2, \ldots, ID_k) \rightarrow (Keywords)$: A group of edge nodes $ID_1, \ldots, ID_k$ with total number k generates a list of keywords for the end devices in the edge computing system.
- Message Encapsulation. $ME(m, E) \rightarrow (C)$: The algorithm can be run by any end device that knows the group public key. Inputting the group public key E and the metadata keyword m, the algorithm outputs the corresponding encryption keyword C. Finally, C and the task data need to be sent to the gateway.
- Test Authorization. $TA(T_1, T_2, \ldots, T_k) \rightarrow (T)$: In order to authorize the gateway to test a specific keyword m, each edge node ID_i in the group needs to generate the corresponding partial trapdoor T_i. The gateway aggregates these partial trapdoors to generate the final trapdoor T.

- Resource allocation. $RA(C, T) \rightarrow (0, 1)$: The gateway runs the algorithm, and the input is the complete trapdoor T and the encryption keyword C. The algorithm outputs 1 if C contains the corresponding keyword m; otherwise, it outputs 0. Subsequently, the gateway can decide the resource allocation result based on the output and the corresponding tag of the edge node.

3.2. The Proposal

As shown in Figure 2. Our scheme consists of several phases, which are described below: Initialization, Registration, Group Generation, Keyword Generation, Message Encapsulation, Test Authorization, and Resource Allocation. The specific construction is shown below:

- Initialization: Taking as input the security parameter k, the TA generates a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are both cyclic groups with prime order q, and P is the generator of $\mathbb{G}_1$. TA randomly chooses $\mathsf{s} \in \mathbb{Z}_q^*$ as the system master secret and computes $P_{pub} = \mathsf{s}P$; then, three cryptographic hash functions are selected, notated as $h_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1$, $h_2 : \{0,1\}^* \longrightarrow \mathbb{G}_1$ and $h_3 : \mathbb{G}_T \longrightarrow \{0,1\}^k$. Then, TA generates a common reference string, which includes a description of the group $\mathbb{G}_1$, denoted by CRS. Finally, the TA publishes all system parameters:

$$\mathsf{param} = (\mathbb{G}_1, \mathbb{G}_T, \hat{e}, P, P_{pub}, h_1, h_2, h_3, CRS).$$

- Registration: During this phase, TA should provide a registration service for edge nodes. Specifically, each edge node sends its ID_i to TA through a secure channel, and TA first computes $q_i = h_1(ID_i)$ and subsequently computes $s_i = \mathsf{s}q_i$ as the private key of edge node ID_i.
- Group Generation: Suppose an edge system consists of k edge nodes with IDs $ID_1, \ldots, ID_k$, and a group public key needs to be negotiated at that phase. Each edge node needs to maintain a set that contains all valid edge node subscripts, i.e., $S = \{1, 2, \ldots., k\}$. Subsequently, the edge nodes perform the following operations:
  - For $1 \leq i \leq k$, the i-th edge node ID_i chooses a random number $x_i \in \mathbb{Z}_q^*$ and computes $r_i = x_i P$ as well as a proof $\pi_{r_i} = PK\{(\underline{x_i}) : r_i = x_i P\}$. Finally, the edge node ID_i sends $\{ID_i, r_i, \pi_{r_i}\}$ to other edge nodes via a secure channel.
  - Upon receiving a set of $\{(r_i, \pi_{r_i})\}_{i \in S}$ from other edge nodes, an edge node verifies the validity of π_{r_i} with regard to r_i and param for all $i \in S$. If π_{r_i} is not valid, this also implies that the edge node is semi-malicious; therefore, the subscript of the node should be removed from the set S, i.e., update $S := S \setminus \{i\}$. Subsequently, these edge nodes need to select a serial number (which can be instantiated based on a concatenation of the date and the value of a counter of the number of the group generation) to negotiate and publicize a unique group ID:

$$GID = ID_1 || ID_2 || \ldots || ID_{i \in S} || serial\ number. \tag{1}$$

  Finally, the edge node can compute and publish the group public key $E = (r, \Phi, S)$, where

$$r = \sum_{i \in S} r_i, \Phi = \hat{e}(\sum_{i \in S} h_1(ID_i), P_{pub}). \tag{2}$$

- Keyword Generation: In this phase, the edge system nodes need to negotiate to generate a list of keywords for the end device. This list of keywords is sent through the secure channel to the end device that wants to use this edge system. In order to resist the keyword guessing attack [43], the edge system needs to select random strings as keywords and the end devices and edge nodes need to record the correspondence between the keywords and the required factors.

- Message Encapsulation: The end device can send task data with encrypted metadata to the gateway. In order to encrypt the keyword m in the metadata, an end device selects $y \in \mathbb{Z}_q^*$ and computes the encrypted keyword (X, Y), where

$$X = yP, Y = h_3((\hat{e}(h_2(GID, m), r) \cdot \Phi)^y). \tag{3}$$

- Test Authorization: Suppose the set S contains k edge nodes, $GID = ID_1||ID_2||\ldots||ID_k$ $||serial\ number$. To authorize the gateway to test the keyword list $m_1, \ldots, m_n$, the i-th edge node needs to compute the corresponding partial trapdoor $w_i = s_i + x_i h_2(GID, m_l)$ for the keyword m_l, where $l \in \{1, 2 \ldots, n\}$. Subsequently, the edge node sends the corresponding (w_i, tag_i) to the gateway via a secure channel in a specific keyword order, where $tag \in \{0, 1\}$. The $tag = 1$ or $tag = 0$ represents whether the i-th edge node is willing to perform the task represented by m or not, respectively. Receiving $\{w_i, tag_i\}_{1 \le i \le k}$, the gateway computes

$$T_{m_l} = \sum_{i=1}^{k} w_i. \tag{4}$$

  T_{m_l} is the trapdoor used to determine whether the encrypted metadata contain the keyword m_l.
- Resource Allocation: After receiving the encryption keyword (X_j, Y_j) in the encrypted metadata, the gateway needs to determine whether the equation

$$h_3(\hat{e}(T_{m_l}, X_j)) \overset{?}{=} Y_j \tag{5}$$

  holds. If the equation holds, the encrypted metadata contain the keyword m_l; otherwise, the encrypted metadata do not contain m_l. After testing all the keywords, the gateway can complete the resource allocation process according to the tag corresponding to the trapdoor. If no edge node is available to execute the task, it will be sent to the cloud for execution.

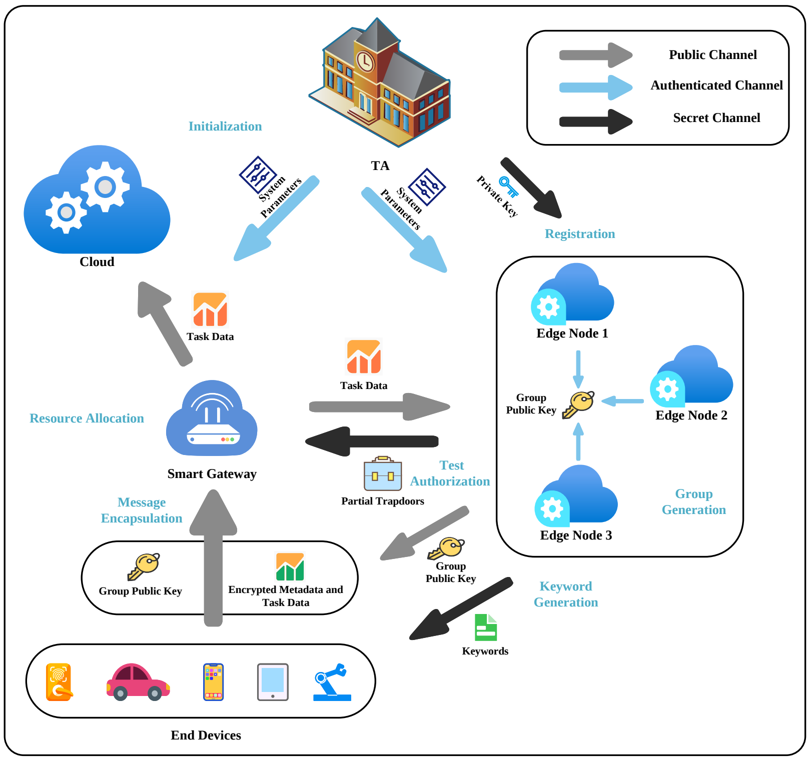

Figure 2. Execution process of our scheme.

4. Security Analysis

This section starts with the definition of the security model for the proposed scheme. The security model captures the properties of user privacy, corrupted edge nodes, and full key compromise resistance. Following this, we provide a formal proof that our proposed scheme satisfies the required security notion.

4.1. Security Model

In our security model, an adversary can be either a gateway or a eavesdropper. We call a gateway adversary a Type I adversary and an eavesdropper adversary a Type II adversary. We define the security of our proposed scheme through the following two games.

Game 1 plays between a challenger $\mathcal{C}$ and a Type I adversary $\mathcal{A}$, which captures the behavior of the Type I adversary. In Game 1, the adversary $\mathcal{A}$ will try to learn the contents of the encrypted metadata. Game 1 consists of three phases.

Initial: $\mathcal{C}$ run Initialization to obtain *master-secret* and the system parameter list. It then sends the system parameters to $\mathcal{A}$ while keeping *master-secret* secret.

Training: Allow $\mathcal{A}$ to adaptively execute a polynomially bounded number of the following types of queries:

- Private key queries: This query is used to model our proposed scheme's full key compromise resistance property. Specifically, $\mathcal{A}$ is allowed to perform this query to obtain the private key of the edge node with the identity ID_i, and the query's output is the corresponding node's private key.
- Group public key queries: $\mathcal{A}$ can use this query to obtain the group public key for a group of edge nodes. $\mathcal{C}$ responds with the corresponding group public key.
- Trapdoor queries: $\mathcal{A}$ can request the trapdoor corresponding to metadata m, and, as a response, $\mathcal{C}$ runs the Test Authorization algorithm to obtain and return the trapdoor.

Output: $\mathcal{A}$ selects a group ID GID^* and sends that group to challenger C. C chooses a keyword m^*, encrypts it, generates a trapdoor T_{m^*}, and sends T_{m^*} to the adversary; if $\mathcal{A}$ is able to correctly decrypt T_{m^*}, we say that $\mathcal{A}$ has won game 1.

Define the advantage of $\mathcal{A}$ winning the above game as

$$Adv(\mathcal{A}) = |\Pr[\xi' = \xi] - \frac{1}{2}|. \tag{6}$$

If for any probabilistic polynomial time adversary $\mathcal{A}$, $Adv(\mathcal{A})$ can be negligible, we say that our scheme is secure against Type I adversaries.

Game 2 plays between a challenger $\mathcal{C}$ and a Type II adversary $\mathcal{A}$, which captures the behavior of the Type II adversary. In Game 2, two encrypted keywords are given to the adversary $\mathcal{A}$, and $\mathcal{A}$ is required to distinguish whether the two contain the same keyword. Game 2 consists of three phases.

Initial: This phase is the same as in Game 1.

Training: Allow $\mathcal{A}$ to adaptively execute a polynomially bounded number of the following types of queries:

- Private key queries: This query is used to model our proposed scheme's full key compromise resistance property. Specifically, $\mathcal{A}$ is allowed to perform this query to obtain the private key of the edge node with identity ID_i, and the query's output is the corresponding node's private key.
- Group public key queries: $\mathcal{A}$ can use this query to obtain the group public key for a group of edge nodes. $\mathcal{C}$ responds with the corresponding group public key.
- Trapdoor queries: $\mathcal{A}$ can request the trapdoor corresponding to metadata m, and, as a response, $\mathcal{C}$ runs the Test Authorization algorithm to obtain and return the trapdoor.
- Semi-malicious corruptions queries: $\mathcal{C}$ needs to maintain an initially empty list L_m with records of the format (ID_i, r_i, π_{r_i}). $\mathcal{A}$ is allowed to add malicious records to L_m.

Output: The adversary $\mathcal{A}$ selects a group ID GID^* and two keywords m_0, m_1. Next, (GID, m_0, m_1) is sent to $\mathcal{C}$. We require that $\mathcal{A}$ cannot query the trapdoor corresponding to m_0 and m_1 under GID^*. However, $\mathcal{A}$ can corrupt all the edge nodes related to GID^*. Then, $\mathcal{C}$ can randomly select the message m_ξ, where $\xi \in \{0,1\}$, and encrypt that message using the group public key corresponding to GID^* to get the encryption keyword C^*, which will be sent to $\mathcal{A}$ as a response. Finally, $\mathcal{A}$ returns its guess: $\xi' \in \{0,1\}$.

If for any probabilistic polynomial time adversary $\mathcal{A}$,

$$Adv(\mathcal{A}) = |\Pr[\xi' = \xi] - \frac{1}{2}| \tag{7}$$

can be negligible, our scheme is secure against Type II adversaries.

4.2. Security Proofs

The security of our proposed scheme is based on the Bilinear Diffie–Hellman (BDH) assumption, which is defined as follows.

BDH Problem: It is difficult to compute $\hat{e}(P,P)^{\alpha\beta\gamma}$ when given $P, \alpha P, \beta P, \gamma P$, where $\alpha, \beta, \gamma \in \mathbb{Z}_q$.

BDH Assumption: Suppose $\mathcal{B}$ is an algorithm that can solve the BDH problem with the advantage

$$Adv(\mathcal{B}) = \Pr\Big[\mathcal{B}(P, \alpha P, \beta P, \gamma P) = \hat{e}(P,P)^{\alpha\beta\gamma}\Big].$$

Then, for any polynomial time algorithm $\mathcal{B}$, $Adv(\mathcal{B})$ can be negligible.

Theorem 1. *In time τ, assuming that $\mathcal{A}$ can win Game 1 with an advantage denoted as $Adv(\mathcal{A})$, this means that there exists an algorithm that breaks the one-way property of the hash function.*

Proof. Next, we will prove that if an adversary $\mathcal{A}$ can break the user privacy, then it means that $\mathcal{C}$ can utilize $\mathcal{A}$ to break the one-way property of the hash function. In the real world, this is considered difficult.

Initial: $\mathcal{C}$ runs Initialization to get *master-secret* and the system parameter list, then sends the system parameters to $\mathcal{A}$ while keeping *master-secret* secret.

Training: Type I adversaries $\mathcal{A}$ are allowed to do the various query requests defined above, and $\mathcal{C}$ responds accordingly as defined in Section 3.2.

Output: $\mathcal{A}$ selects a group ID GID^*, generates the corresponding group encryption public key $E^* = (r^*, \Phi^*)$, and sends (GID^*, E^*) to $\mathcal{C}$. $\mathcal{C}$ randomly chooses a keyword m^* and randomly chooses $o^* \in \mathbb{G}_1$ as the trapdoor corresponding to m^*. If $\mathcal{C}$ outputs its guess as $m^{*\prime} = m^*$, an algorithm exists that breaks the one-way property of the hash function. □

Theorem 2. *Suppose the group size is N; in time τ, the Type II adversary $\mathcal{A}$ requested q_{h_3} queries for h_3 and requested q_t trapdoor queries. Suppose that the adversary $\mathcal{A}$ can win Game 2 with an advantage denoted by $Adv(\mathcal{A})$. This then means that there exists an algorithm to solve the BDH problem, which has an advantage of*

$$\frac{1}{q_{h_3} N e^2}\left(\frac{2}{2q_t + 2}\right)^2 Adv(A).$$

Proof. Next, we will prove that if the adversary $\mathcal{A}$ can break our scheme, then $\mathcal{C}$ can utilize $\mathcal{A}$ to solve the BDH problem. In the real world, this is considered difficult.

Initial: $\mathcal{C}$ sets $P_{pub} = \alpha P$, then generates the system parameters

$$params = (\mathbb{G}_1, \mathbb{G}_T, \hat{e}, P, P_{pub}, h_1, h_2, h_3),$$

and sends the system parameters to $\mathcal{A}$. Assume that the number of edge nodes in the system is N. The i-th edge node ID_i performs the registration when $\mathcal{C}$ flips a coin $coin_i$;

the probability of producing a 1 is δ, and the probability of producing a 0 is $1 - \delta$, where $1 \leq i \leq N$.

Training: We consider h_1, h_2, and h_3 to be random oracles, and $\mathcal{C}$ responds to the request of $\mathcal{A}$ as follows:

- h_1 queries: $\mathcal{C}$ needs to maintain an initially empty list L_{h_1}. When ID_i is input, $\mathcal{C}$ first checks whether the record (ID_i, d_i, q_i, s_i) exists in L_{h_1}, and, if it does, it returns q_i to $\mathcal{A}$; otherwise, $\mathcal{C}$ randomly selects $d_i \in \mathbb{Z}_q^*$, computes $q_i = d_iP, s_i = d_iP_{pub}$, adds the record (ID_i, d_i, q_i, s_i) to L_{h_1}, and returns q_i to $\mathcal{A}$.
- h_2 queries: $\mathcal{C}$ needs to maintain an initially empty list L_{h_2}. When ID_i is input, $\mathcal{C}$ first checks whether the record $(GID_i, m_i, w_i, o_i, coin_i^{h_2})$ exists in L_{h_2}, and, if it does, it returns o_i to $\mathcal{A}$. Otherwise, $\mathcal{C}$ flips a coin $coin_i^{h_2}$, assuming that the probability of the coin yielding 1 is δ and the probability of yielding 0 is $1 - \delta$, and, subsequently, $\mathcal{C}$ randomly selects $w_i \in \mathbb{Z}_q^*$.
  - If $coin_i^{h_2} = 1$, compute $o_i = (w_i + \beta)P$, add record $(GID_i, m_i, w_i, o_i, coin_i^{h_2})$ to L_{h_2}, and return o_i as response.
  - Else, compute $o_i = w_iP$, add record $(GID_i, m_i, w_i, o_i, coin_i^{h_2})$ to L_{h_2}, and return o_i as response.
- h_3 queries: $\mathcal{C}$ needs to maintain an initially empty list L_{h_3}. When Λ_i is input, $\mathcal{C}$ first checks whether record (Λ_i, Υ_i) exists in L_{h_3}, and, if it does, returns Υ_i to $\mathcal{A}$; otherwise, $\mathcal{C}$ randomly selects $\Upsilon_i \in \{0,1\}^*$, adds the record (Λ_i, Υ_i) to L_{h_3}, and returns Υ_i to $\mathcal{A}$.
- Private key queries: The query takes ID_i as input, and, upon receiving this query, $\mathcal{C}$ performs an h_1 query with ID_i as input, and then recovers the corresponding (ID_i, d_i, q_i, s_i) from L_{h_1}, returning s_i as a response.
- Semi-malicious corruptions queries: $\mathcal{A}$ inputs (ID_i, r_i, π_{r_i}), and $\mathcal{C}$ adds the record (ID_i, r_i, π_{r_i}) to L_m.
- Group public key queries: $\mathcal{C}$ maintains a list L_G with records of the form

  $$(GID_l, r_1, r_2, \ldots, r_k, x_1, x_2, \ldots, x_k, E, \Phi).$$

  The query receives $GID_j = (ID_1||ID_2||\ldots||ID_k||serial\ number)$ as input; for $1 \leq i \leq k$, $\mathcal{C}$ first determines whether there is a record associated with ID_i in L_m, and, if not, $\mathcal{C}$ randomly chooses $x_i \in \mathbb{Z}_q^*$, otherwise recovering the corresponding record (ID_i, r_i, π_{r_i}) from L_m, and we note that, in this work, we use extractable knowledge proofs so that we can extract the x_i from π_{r_i}. Then, if $coin_i = 0$, $\mathcal{C}$ computes $r_i = x_iP$, and if $coin_i = 1$, $\mathcal{C}$ calculates $r_i = (x_i - \alpha)P$. Finally, $\mathcal{C}$ computes

  $$E = (r = \sum_{i=1}^{k} r_i, \Phi = \hat{e}(\sum_{i=1}^{k} h_1(ID_i), P_{pub})). \tag{8}$$

  Add the record

  $$(GID_j, r_1, r_2, \ldots, r_k, x_1, x_2, \ldots, x_k, E, \Phi)$$

  to G^{lits}.
- Trapdoor queries: $\mathcal{C}$ maintains a list L_t with records of the form (GID_i, m_i, T_i). The query receives (GID_j, m_j) as input, $\mathcal{C}$ first recovering

  $$(GID_j, r_1, r_2, \ldots, r_k, x_1, x_2, \ldots, x_k, E, \Phi)$$

  from L_G. For $1 \leq i \leq k$, $\mathcal{C}$ recovers the ID_i corresponding to the records (ID_i, d_i, q_i, s_i) from L_{h_1}, and the (GID_j, m_j) corresponding records $(GID_j, m_j, w_j, o_j, coin_j^{h_2})$ from L_{h_2}. Then, $\mathcal{C}$ determines whether a record (GID_j, m_j, T_j) corresponding to (GID_j, m_j) exists in L_t, and, if it does, $\mathcal{C}$ returns T_j as a response. If $1 \leq i \leq k$, then $\mathcal{C}$ executes the following steps:

- If $coin_i = 0$, since $\mathcal{C}$ has knowledge of x_i and the private key of ID_i, $\mathcal{C}$ can use the *Trapdoor* algorithm to generate w_i.
- Else, if $coin_i^{h_2} = 0$, computer $w_i = s_i + w_j r_i$.
- Else, abort. We denote the event by Event 1.

If Event 1 does not occur, $\mathcal{C}$ computes $T_j = \sum_{i=1}^{k} w_i$ and adds the record (GID_j, m_j, T_j) to L_t.

- *Challenge*: $\mathcal{A}$ chooses a group ID GID^* corresponding to $\mathbb{L}_{ID}^* = \{ID_1^*, ID_2^* \dots, ID_x^*\}$, two keywords m_0^*, m_1^*, and the group public key $E^* = (r^*, \Phi^*)$. Then, $\mathcal{A}$ sends $(GID^*, E^*, m_0^*, m_1^*)$ to $\mathcal{C}$. $\mathcal{C}$ randomly selects $\xi \in \{0,1\}$, $\Upsilon^* \in \{0,1\}^l$, sends $(\gamma P, \Upsilon^*)$ as a response to $\mathcal{A}$, and finally, $\mathcal{A}$ outputs its guess ξ'.
- *Output*: If $\xi' = \xi$, $\mathcal{C}$ recovers the records $(GID^*, r_1^*, \dots, r_k^*, x_1^*, \dots, x_k^*, E^*, \Phi^*)$ from L_G, for $1 \leq l \leq k$, $\mathcal{C}$ recovers the record (ID_l, μ_l, f_l, s_l) from L_{h_1}. For (GID^*, m_ξ^*), $\mathcal{C}$ recovers the corresponding records $(GID^*, m_\xi^*, w_\xi^*, o_\xi^*, H_2 coin_\xi^*)$ from L_{h_2}. For $1 \leq l \leq k$, we denote the value of the coin flip corresponding to ID_l as $coin_l^*$. It requires that only one coin corresponds to a value of 1. Then, $\mathcal{C}$ randomly selects the pair (Λ_i, Υ_i) from L_{h_3}. Finally, $\mathcal{C}$ outputs

$$\Lambda_i / (e((\gamma \sum_{i=1}^{k} d_i^*)P, P_{pub}) e(w_\xi^* r, \gamma P) e((\beta \sum_{i=1}^{k} x_i^*)P, \gamma P)) \tag{9}$$

as the answer to the BDH problem.
When Event 1 does not happen, $\mathcal{A}$ will not notice the difference between the simulation and the real world, so we have

$$Pr[\xi' = \xi] \geq Adv(\mathcal{A}), Pr[\neg Event1] \geq ((1-\delta)(1-N\delta))^{q_t}. \tag{10}$$

We note that for $\mathcal{C}$ to output a solution to the BDH problem, it is required that, for an index $l \in [1,k]$, $coin_l^* = 1$ and $coin_\xi^{h_2 *} = 1$. And these occur with a probability of at least $N\delta^2$. Thus, we have $\mathcal{C}$ outputting a solution to the BDH problem with the probability

$$\frac{1}{q_{h_3}} N\delta^2 ((1-\delta)(1-N\delta))^{q_t} Adv(\mathcal{A}) \geq \frac{1}{q_{h_3} Ne^2} (\frac{2}{2q_t + 2})^2 Adv(A). \tag{11}$$

□

5. Security Comparison and Performance Evaluation

Although the work of Barman et al. [12] and Jiang et al. [13] is robust, as mentioned before, it is unrealistic to introduce these complex schemes in edge computing systems with limited computational and bandwidth resources. Therefore, we compare the security of our proposed scheme with Zhang et al.'s proposed scheme [14], which also uses searchable encryption. We have compared the security of the two schemes via formal security proofs, and the comparison results are shown in Table 1. The results show that our scheme can resist semi-malicious edge nodes, which Zhang et al.'s scheme cannot do. Our proposed scheme and Zhang et al.'s proposed scheme have similar computational overheads in the other phases, with the main gap being the extractable zero-knowledge proofs introduced in the group generation phase. To evaluate these extra computational overheads, we performed experiments on the zero-knowledge proof scheme proposed by Cramer et al. [44] The experiment was performed using the Miracl library on a PC equipped with an i5-9400F CPU and 16 GB RAM. The experimental results show that the proof generation time is 2.98 s, while the verification time is 26.32 ms. Since the group generation phase is executed only once, such extra computational overhead is acceptable. In recent years, some more efficient zero-knowledge proof schemes have been proposed [45]. In practice, the appropriate zero-knowledge proof schemes can be selected as the building blocks of our proposed scheme as needed.

Table 1. Comparison of security.

	Metadata Privacy	Full Key Compromise Resistance	Semi-Malicious Edge Nodes Resistance
Zhang et al.'s scheme [14]	✓	✓	✗
Our proposed scheme	✓	✓	✓

Next, we evaluate our proposed scheme in terms of performance. To our knowledge, no metadata privacy-preserving resource allocation scheme exists for resistance to semi-malicious edge nodes. Therefore, we only evaluate the performance of our proposed scheme. In addition, since the other phases need to be executed only once, we only evaluate the efficiency of the Message Encapsulation phase, the Test Authorization phase, and the Resource Allocation phase. Specifically, in this section, we use the petrelic library to evaluate the computational overhead of our proposed scheme. We performed the measurements on a PC with an Intel i5-9400F CPU, 16GB of RAM, and Ubuntu 18.04.

We increase the number of encrypted keywords from 1 to 100 and calculate the required execution time in order to evaluate the performance of the Message Encapsulation phase. The results are shown in Figure 3. Our proposed scheme requires an execution time of about 1.607 ms for a single keyword. Similarly, we evaluate the Resource Allocation phase, where the execution time is about 0.858 ms for a single keyword. Since the metadata attached to the edge computation will not contain too many keywords, the corresponding two phase computation overhead will not be particularly large. In addition, to evaluate the scheme's performance on edge devices with limited computational resources, we performed experiments on a Raspberry Pi 3 Model B+ using the PBC library. The results are shown in Figure 4. The execution times of our scheme in the Message Encapsulation phase and Resource Allocation for a single keyword are 37.071 ms and 15.102 ms, respectively. Since the metadata attached to the edge computation will not contain too many keywords, the corresponding two-phase computation overhead is acceptable for edge devices.

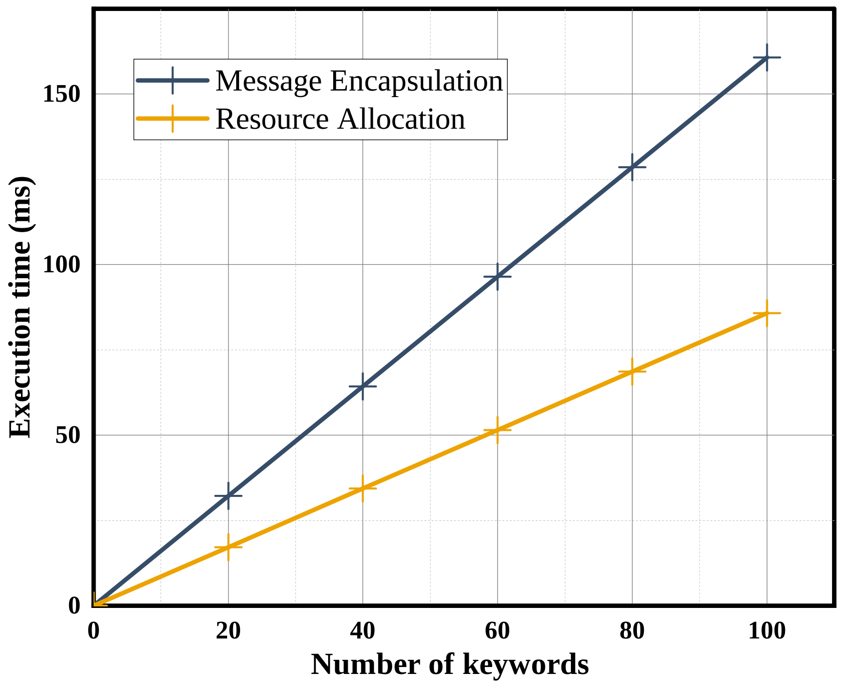

Figure 3. Execution time of Message Encapsulation phase and Resource Allocation phase on the PC.

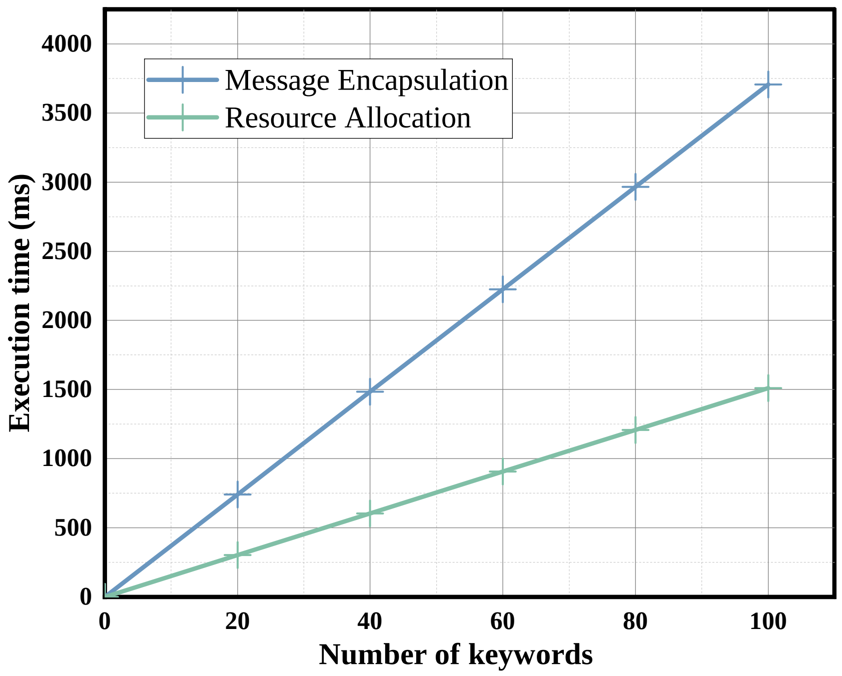

Figure 4. Execution time of Message Encapsulation phase and Resource Allocation phase on the Raspberry Pi.

Since the number of keywords contained in the metadata and the number of edge nodes in the edge computing system will not be too many, we set the number of encrypted keywords to grow from 1 to 10, and the number of partial trapdoors to increase from 1 to 100, to evaluate the performance efficiency of the Test Authorization phase. Similarly, the experiment was performed on a PC and a Raspberry Pi, and the results are shown in Figures 5 and 6, respectively. We can see that the performance efficiency of the Test Authorization phase is usually sufficient to meet the requirements.

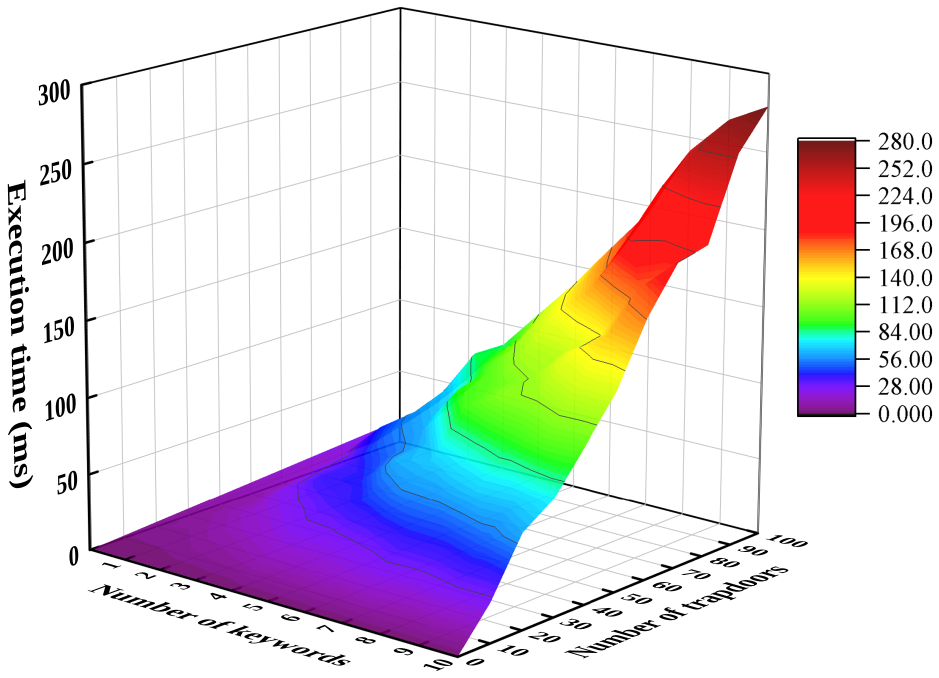

Figure 5. Execution time of Test Authorization phase on the PC.

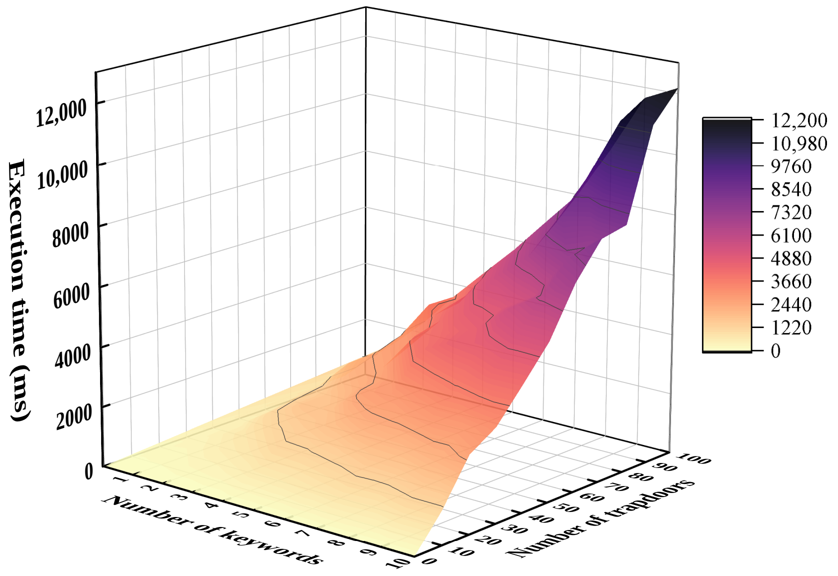

Figure 6. Execution time of Test Authorization phase on the Raspberry Pi.

6. Conclusions

Resource allocation is the core problem of edge computing, but existing resource allocation schemes in edge computing often lack the protection of metadata privacy and consideration of semi-malicious edge nodes. In this paper, we propose a metadata-privacy resource allocation scheme based on searchable encryption and use a zero-knowledge proof to resist semi-malicious edge nodes. Our proposed scheme achieves constant message expansion, which contributes to scalability in practice. Through a formal security analysis, we demonstrate that the scheme satisfies the necessary security and privacy requirements. We show that the scheme's efficiency can meet the requirements through experiments on PC and Raspberry Pi. Overall, our proposed scheme provides a practical and robust solution for resource allocation in edge computing.

In future work, we will consider introducing trust mechanisms and integrating them with existing edge computing frameworks, which will contribute to the dynamism and scalability of the scheme. In addition, introducing more powerful malicious edge nodes and de-trusting TA are two interesting issues that will provide more robust security for edge computing systems.

Author Contributions: Conceptualization, Z.Z. and Y.H.; methodology, Z.Z. and J.L.; validation, Z.Z. and J.L.; formal analysis, Z.Z. and Y.L.; investigation, Z.Z., J.L. and Y.L.; writing—original draft preparation, Z.Z.; writing—review and editing, Z.Z., J.L. and Y.H.; visualization, Z.Z.; supervision, Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program (2023YFB2504800), Science and Technology on Communication Security Laboratory Foundation (6142103022208), SongShan Labtory Pre-Research Project (No. YYJC042022016), Shanghai Sailing Program (No. 21YF1413800).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Yuanhang He was employed by the company China Electronics Technology Group Corporation. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Satyanarayanan, M. The emergence of edge computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
2. Yang, C.; Shen, W.; Wang, X. The internet of things in manufacturing: Key issues and potential applications. *IEEE Syst. Man Cybern. Mag.* **2018**, *4*, 6–15. [CrossRef]
3. Tzounis, A.; Katsoulas, N.; Bartzanas, T.; Kittas, C. Internet of Things in agriculture, recent advances and future challenges. *Biosyst. Eng.* **2017**, *164*, 31–48. [CrossRef]
4. Jin, J.; Gubbi, J.; Marusic, S.; Palaniswami, M. An information framework for creating a smart city through internet of things. *IEEE Internet Things J.* **2014**, *1*, 112–121. [CrossRef]
5. Yuehong, Y.; Zeng, Y.; Chen, X.; Fan, Y. The internet of things in healthcare: An overview. *J. Ind. Inf. Integr.* **2016**, *1*, 3–13.
6. Mach, P.; Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
7. Aazam, M.; Huh, E.N. Fog computing and smart gateway based communication for cloud of things. In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud, Barcelona, Spain, 27–29 August 2014; pp. 464–470.
8. Zhang, T.; Li, Y.; Chen, C.P. Edge computing and its role in Industrial Internet: Methodologies, applications, and future directions. *Inf. Sci.* **2021**, *557*, 34–65. [CrossRef]
9. Hunkeler, U.; Truong, H.L.; Stanford-Clark, A. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In Proceedings of the 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08), Bangalore, India, 5–10 January 2008; pp. 791–798.
10. Beams, A.; Kannan, S.; Angel, S. Packet scheduling with optional client privacy. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, 15–19 November 2021; pp. 3415–3430.
11. Sasy, S.; Goldberg, I. SoK: Metadata-Protecting Communication Systems. In Proceedings of the 24th Privacy Enhancing Technologies Symposium (PETS 2024), Bristol, UK, 15–20 July 2024.
12. Barman, L.; Kol, M.; Lazar, D.; Gilad, Y.; Zeldovich, N. Groove: Flexible {Metadata-Private} Messaging. In Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), Carlsbad, CA, USA, 11–13 July 2022; pp. 735–750.
13. Jiang, P.; Wang, Q.; Cheng, J.; Wang, C.; Xu, L.; Wang, X.; Wu, Y.; Li, X.; Ren, K. Boomerang:{Metadata-Private} Messaging under Hardware Trust. In Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23), Boston, MA, USA, 17–19 April 2023; pp. 877–899.
14. Zhang, L.; Li, J. Enabling robust and privacy-preserving resource allocation in fog computing. *IEEE Access* **2018**, *6*, 50384–50393. [CrossRef]
15. Xiao, Y.; Jia, Y.; Liu, C.; Cheng, X.; Yu, J.; Lv, W. Edge computing security: State of the art and challenges. *Proc. IEEE* **2019**, *107*, 1608–1631. [CrossRef]
16. Dinh, T.Q.; Tang, J.; La, Q.D.; Quek, T.Q. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Trans. Commun.* **2017**, *65*, 3571–3584.
17. Tran, T.X.; Pompili, D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Trans. Veh. Technol.* **2018**, *68*, 856–868. [CrossRef]
18. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [CrossRef]
19. Lu, R.; Heung, K.; Lashkari, A.H.; Ghorbani, A.A. A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced IoT. *IEEE Access* **2017**, *5*, 3302–3312. [CrossRef]
20. Lyu, L.; Nandakumar, K.; Rubinstein, B.; Jin, J.; Bedo, J.; Palaniswami, M. PPFA: Privacy preserving fog-enabled aggregation in smart grid. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3733–3744. [CrossRef]
21. Zhang, L.; Zou, Y.; Wang, W.; Jin, Z.; Su, Y.; Chen, H. Resource allocation and trust computing for blockchain-enabled edge computing system. *Comput. Secur.* **2021**, *105*, 102249. [CrossRef]
22. Kong, W.; Li, X.; Hou, L.; Yuan, J.; Gao, Y.; Yu, S. A Reliable and Efficient Task Offloading Strategy Based on Multifeedback Trust Mechanism for IoT Edge Computing. *IEEE Internet Things J.* **2022**, *9*, 13927–13941. [CrossRef]
23. Zhou, J.; Choo, K.K.R.; Cao, Z.; Dong, X. PVOPM: verifiable privacy-preserving pattern matching with efficient outsourcing in the malicious setting. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 2253–2270. [CrossRef]
24. Li, T.; Tian, Y.; Xiong, J.; Bhuiyan, M.Z.A. FVP-EOC: Fair, Verifiable, and Privacy-Preserving Edge Outsourcing Computing in 5G-Enabled IIoT. *IEEE Trans. Ind. Inform.* **2023**, *19*, 940–950. [CrossRef]
25. Wang, Y.; Su, Z.; Luan, T.H.; Li, J.; Xu, Q.; Li, R. SEAL: A Strategy-Proof and Privacy-Preserving UAV Computation Offloading Framework. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 5213–5228. [CrossRef]
26. Angel, S.; Kannan, S.; Ratliff, Z. Private resource allocators and their applications. In Proceedings of the 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 18–20 May 2020; pp. 372–391.
27. Ahmad, I.; Yang, Y.; Agrawal, D.; El Abbadi, A.; Gupta, T. Addra: Metadata-private voice communication over fully untrusted infrastructure. In Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21), Virtual, 14–16 July 2021.
28. Chor, B.; Kushilevitz, E.; Goldreich, O.; Sudan, M. Private information retrieval. *J. ACM* **1998**, *45*, 965–981. [CrossRef]

29. Cai, C.; Zang, Y.; Wang, C.; Jia, X.; Wang, Q. Vizard: A metadata-hiding data analytic system with end-to-end policy controls. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; pp. 441–454.
30. Langowski, S.; Servan-Schreiber, S.; Devadas, S. Trellis: Robust and scalable metadata-private anonymous broadcast. Cryptology ePrint Archive. 2022. Available online: https://eprint.iacr.org/2022/1548 (accessed on 31 March 2024).
31. Liu, D. Efficient processing of encrypted data in honest-but-curious clouds. In Proceedings of the 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 27 June–2 July 2016; pp. 970–974.
32. Samie, F.; Tsoutsouras, V.; Bauer, L.; Xydis, S.; Soudris, D.; Henkel, J. Computation offloading and resource allocation for low-power IoT edge devices. In Proceedings of the 2016 IEEE 3rd world forum on internet of things (WF-IoT), Reston, VA, USA, 12–14 December 2016; pp. 7–12.
33. Qiu, T.; Chi, J.; Zhou, X.; Ning, Z.; Atiquzzaman, M.; Wu, D.O. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2462–2488. [CrossRef]
34. Boneh, D.; Franklin, M. Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **2003**, *32*, 586–615. [CrossRef]
35. Gupta, D.S.; Ray, S.; Singh, T.; Kumari, M. Post-quantum lightweight identity-based two-party authenticated key exchange protocol for internet of vehicles with probable security. *Comput. Commun.* **2022**, *181*, 69–79. [CrossRef]
36. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, S&P 2000, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
37. Poh, G.S.; Chin, J.J.; Yau, W.C.; Choo, K.K.R.; Mohamad, M.S. Searchable symmetric encryption: designs and challenges. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–37. [CrossRef]
38. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (Advances in Cryptology—EUROCRYPT 2004), Interlaken, Switzerland, 2–6 May 2004; pp. 506–522.
39. Hwang, Y.H.; Lee, P.J. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In Proceedings of the International Conference on Pairing-Based Cryptography, Tokyo, Japan, 2–4 July 2007; pp. 2–22.
40. Goldwasser, S.; Micali, S.; Rackoff, C. The knowledge complexity of interactive proof-systems. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*; ACM Books: New York, NY, USA, 2019; pp. 203–225.
41. Camenisch, J.; Kiayias, A.; Yung, M. On the portability of generalized schnorr proofs. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, 26–30 April 2009; pp. 425–442.
42. Fischlin, M. Communication-efficient non-interactive proofs of knowledge with online extractors. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2005; pp. 152–168.
43. Byun, J.W.; Rhee, H.S.; Park, H.A.; Lee, D.H. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In Proceedings of the Workshop on Secure Data Management, Seoul, Republic of Korea, 10–11 September 2006; pp. 75–83.
44. Cramer, R.; Damgård, I.; MacKenzie, P. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In Proceedings of the International Workshop on Public Key Cryptography, Victoria, Australia, 18–20 January 2000; pp. 354–372.
45. Block, A.R.; Holmgren, J.; Rosen, A.; Rothblum, R.D.; Soni, P. Public-coin zero-knowledge arguments with (almost) minimal time and space overheads. In Proceedings of the Theory of Cryptography Conference, Durham, NC, USA, 16–19 November 2020; pp. 168–197.