electronics

MDPI

Article

Enabling Privacy-Preserving Data Sharing with Bilateral Access Control for Cloud

Tong Wu [1], Xiaochen Ma [2] and Hailun Yan [3,4,*]

[1] School of Computer Science and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China; tongw@ustb.edu.cn
[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100811, China; xiaochenm@bit.edu.cn
[3] School of Crypotology, University of Chinese Academy of Sciences, Beijing 101408, China
[4] China Industrial Control Systems Cyber Emergency Response Team, Beijing 100040, China
* Correspondence: hailun.yan@ucas.ac.cn

Abstract: Cloud computing plays an essential role in various fields. However, the existing cloud services face a severe challenge, which is how to share the data among a large scale of devices securely. In this paper, we introduce a cloud-based privacy-preserving data sharing scheme, derived from identity-based matchmaking encryption. In our scheme, the access policies are designed by both the sender and receiver simultaneously, to support bilateral access control. To improve efficiency, we delegate the match algorithm to the cloud server, reducing the computation cost and communication overhead on end devices without revealing the users' privacy. Through formal security analysis, we show that our scheme holds security, authenticity, and privacy. Finally, we evaluate our scheme by conducting extensive experiments, indicating that our scheme is more efficient than the other data-sharing schemes in ME-based services in a real-world dataset.

Keywords: bilateral access control; cloud computing; identity-based matchmaking encryption; privacy preservation

check for updates

Citation: Wu, T.; Ma, X.; Yan, H. Enabling Privacy-Preserving Data Sharing with Bilateral Access Control for Cloud. *Electronics* **2023**, *12*, 4798. https://doi.org/10.3390/electronics12234798

Academic Editor: Andrei Kelarev

Received: 15 October 2023
Revised: 20 November 2023
Accepted: 21 November 2023
Published: 27 November 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Recently, cloud services [1] have been rapidly promoted by the development of the internet technique. As a widely used paradigm of outsourcing service, the cloud has been accepted by the market (e.g., iCloud, Dropbox, and Microsoft Cloud), providing a convenient and low-cost method for data storage and data sharing [2]. As shown in Figure 1, cloud services play an important role in our daily life, such as smart healthcare, smart agriculture, smart cities, and smart transport [3–6]. According to the report released by Gartner in 2021, more than 45% of IT spending will be on building infrastructure, applications, and business process outsourcing, shifting from traditional solutions to the cloud by 2024. Despite the proliferation of the cloud, data security and privacy preservation arise as long-term concerns from the user side, since they lose physical control of their data. Therefore, cloud service providers (CSPs) are commonly treated as honest-but-curious (HBC) entities. On the other hand, different cloud services should prevent data breaches on the cloud to enhance CSPs' reliability [7]. Therefore, it is crucial to design a secure and privacy-preserving data sharing scheme for cloud services.

The General Data Protection Regulation (GDPR) sets strict privacy requirements for CSPs. Specifically, three principles must be satisfied: (1) *Receiver access control.* From the data collection limitation principle, the data should only be sent to receivers that meet the data sender's access policies. (2) *Sender access control.* From the data quality principle, the data sender should be identified to ensure data accuracy. (3) *Data privacy*. From the data privacy principle, sensitive data (e.g., access policies and shared data) should not be disclosed. Thereby, the access control in the data sharing scheme should be designed

by both the sender and receiver. Moreover, the cloud-based data sharing scheme should guarantee data privacy when devices share data and store it on the cloud.

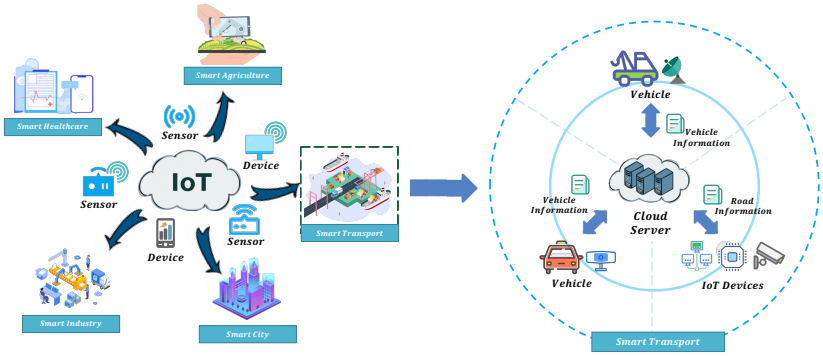

Figure 1. Data-sharing system for cloud-based services.

Inherently, several significant challenges arise when applying existing data sharing schemes to cloud services [8–14], which are not only caused by data breaches but also by the users' strict privacy requirements. Taking the smart transport system as an example, end devices (e.g., distance sensors, speed sensors, and temperature sensors) collect information from vehicles [15]. By analyzing the relevant information, the smart transport system is able to perform more precise and effective traffic management. Due to the restricted computational resources and storage capability of end devices, vehicles primarily outsource the data collection to the cloud server. As is shown in Figure 1, vehicles are willing to share their data for the purpose of avoiding traffic jams and planning optimal travel paths. The data collections are transmitted from end devices to the cloud server. Then, the required data will be sent to target vehicles by the cloud server. Commonly, the collected data might contain sensitive information such as individuals' daily action trajectories and real-time locations. If this information stored on the cloud server is accessible to anyone, it will directly threaten users' data security. Therefore, it is necessary to ensure that sensitive information cannot be snooped on by the cloud server and prevent unauthorized entities from illegal access.

However, most current access control schemes only support one-side access control (i.e., sender/receiver access control). The one-side access control schemes cannot satisfy the practical privacy requirements, but result in vast communication overhead for transmitting information in the system. By applying bilateral access control, the access policy can be designed both by senders and receivers. Specifically, vehicles expect to grant access privileges to their information to the designated end devices. The end devices can also decide to get the information from the specified vehicles or other devices simultaneously. Intuitively, attribute-based encryption (ABE) seems to be a possible solution to address access control among multiple users [16]. The standard ABE approaches cannot support bilateral access control. To tailor the ABE technique for bilateral access control, ABE with a keyword search (ABKS) enables receivers to seek suitable senders using keywords [10]. Their scheme, however, requires additional interactions between the users and the cloud server, introducing extra communication overheads to users. Later, Ateniese et al. [17] presented an encryption primitive on CRYPTO'19, named Matchmaking Encryption (ME), to achieve bilateral access control without revealing any privacy for both senders and receivers. When the matching fails, nothing (i.e., the access policies and data) will be disclosed. However, the matching process brings heavy computation and communication overhead to end devices. To further improve efficiency, it is desirable to delegate the matching process to the cloud without revealing any users' private information. To summarize, the practical secure data sharing for cloud services should be with privacy preservation and bilateral access control in order to facilitate the blossom of cloud services.

In this paper, we introduce a cloud-based privacy-preserving data sharing scheme with bilateral access control. By analyzing the practical security requirements, we formalize

the crucial challenges in the state-of-the-art. Specifically, to provide bilateral access control, we construct our scheme based on identity-based matchmaking encryption (IB-ME) for realizing both sides designing the match policies simultaneously. To achieve high efficiency, we delegate the matching process to the cloud server while protecting the user's private information and data by designing a signature-based match tag. The contributions of our work are summarized as follows:

- We suggest a data-sharing scheme for cloud services, derived from identity-based matchmaking encryption, named IBME-DS. The access policies in IBME-DS are specified by both the sender and receiver to achieve bilateral access control.
- To further improve the system efficiency, we design a privacy-preserving matching mechanism to delegate the matching process to the cloud server, which ensures user privacy and data confidentiality during the matching procedure.
- We formally define the system model, threat model, and security model of IBME-DS. Then, a comprehensive security analysis is to demonstrate that our proposed scheme meets the practical security requirements.
- Finally, we evaluate the performance of IBME-DS by conducting extensive experiments on a real-world dataset to show that IBME-DS is more efficient than relevant works.

Organization. The remainder of this paper is structured as follows. Section 2 discusses the preliminary adopted in this paper. In Section 3, we define the system model, threat model and security model of our scheme. Section 4 provides the concrete construction based on bilinear groups. Then, in Section 5, we give rigorous security proof to prove the security of our scheme. Then, Section 6 presents the theoretical analysis and experimental performance. In Section 7, we introduce relevant works on access control and matchmaking encryption. In Section 8, we discuss the advantages of our research and the limitations of it. Finally, we conclude our work in Section 9.

2. Preliminary

Definition 1 (Bilinear Pairing). *Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups of order p, with p being a large prime. We call them bilinear groups with such a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, if they hold the properties as follows.*

1. Bilinearity: The map e is bilinear, if $e(g^a, h^b) = e(g^b, h^a) = e(g, h)^{ab}$, for $\forall g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$.
2. Non-degeneracy: There exists $e(g, h) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity in $\mathbb{G}_T$.
3. Computability: $e(g, h)$ can be computed efficiently, for $\forall g \in \mathbb{G}_1, h \in \mathbb{G}_2$.

Our scheme is constructed on symmetric pairing that $\mathbb{G}_1 = \mathbb{G}_2$. In the following part of this paper, we denote the bilinear pairing e as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

2.1. Matchmaking Encryption

In this part, we briefly review the matchmaking encryption [17]. In ME, the access control is specified by both sender and receiver. Specifically, ME contains six algorithms:

- **Setup**$(1^\lambda) \to (\mathtt{mpk}, \mathtt{kpol}, \mathtt{msk})$: Given the security parameter 1^λ, the algorithm is to initialize the system and output a set of master keys, as master public/policy/secret key $\{\mathtt{mpk}, \mathtt{kpol}, \mathtt{msk}\}$.
- **SKGen**$(\mathtt{msk}, \sigma) \to ek_\sigma$: Given the sender's attributes $\sigma \in \{0,1\}^*$, and the master secret key $\mathtt{msk}$, the algorithm is to generate the encryption key ek_σ to the sender with attributes σ.
- **RKGen**$(\mathtt{msk}, \rho) \to dk_\rho$: Given the receiver's attributes $\rho \in \{0,1\}^*$, and the master secret key $\mathtt{msk}$, the algorithm is to generate the decryption key dk_ρ to the sender with attributes ρ.
- **PolGen**$(\mathtt{kpol}, \mathbb{S}) \to dk_\mathbb{S}$: Given the access policy $\mathbb{S}$, and the master policy key $\mathtt{kpol}$, the algorithm is to generate the decryption key $dk_\mathbb{S}$ for the access policy $\mathbb{S}$.

- **Enc**$(ek_\sigma, \mathbb{R}, m) \rightarrow c$: Given the encryption key ek_σ, the access policy $\mathbb{R}$, and the message m, the algorithm is to generate the ciphertext c.
- **Dec**$(dk_\rho, dk_\mathbb{S}, c) \rightarrow m$ or $\perp$: Given the decryption key dk_ρ, the decryption key $dk_\mathbb{S}$, and the ciphertext c, the algorithm is to output either the message m or $\perp$.

If $\rho = \mathbb{R}$ and $\sigma = \mathbb{S}$, the given ciphertext will be correctly decrypted by the receiver. Worthy, it will reveal nothing except the matching does not occur.

2.2. Identity-Based Matchmaking Encryption

Additionally, we recap identity-based matchmaking encryption, including five algorithms:

- **Setup**$(1^\lambda) \rightarrow (\mathtt{mpk}, \mathtt{msk})$: Given the security parameter 1^λ, the algorithm will output the master public/secret key $\{\mathtt{mpk}, \mathtt{msk}\}$.
- **SKGen**$(\mathtt{msk}, \sigma) \rightarrow ek_\sigma$: Given the sender's identity σ, and the master secret key $\mathtt{msk}$, the algorithm will output the encryption key ek_σ.
- **RKGen**$(\mathtt{msk}, \rho) \rightarrow dk_\rho$: Given the receiver's identity ρ, and the master secret key $\mathtt{msk}$, the algorithm will output the decryption key dk_ρ.
- **Enc**$(\mathtt{mpk}, ek_\sigma, \mathtt{rcv}, m) \rightarrow c$: Given the target receiver's identity $\mathtt{rcv}$, the master public key $\mathtt{mpk}$, the encryption key ek_σ, and the message $m \in \{0,1\}^n$, and the algorithm will output the ciphertext c, associated to both σ and $\mathtt{rcv}$.
- **Dec**$(\mathtt{mpk}, dk_\rho, \mathtt{snd}, c) \rightarrow m$ or $\perp$: Given the target sender's identity $\mathtt{snd}$, the master public key $\mathtt{mpk}$, the decryption key dk_ρ, and the ciphertext c, the algorithm will compute and output either the message m or $\perp$.

The message will be correctly recovered from the decryption algorithm if $\rho = \mathtt{rcv}$ and $\sigma = \mathtt{snd}$.

3. Definition and System Model

In this section, we formally define the system model of IBME-DS. Furthermore, we describe the potential threats to IBME-DS and formally define the security model and the design goal of our proposed scheme.

3.1. System Model

To clarify the system architecture of IBME-DS, we describe the system model in this section. There are four entities involved in our scheme, including key generation center (KGC), cloud server (CS), sender and receiver, as shown in Figure 2. The CS is responsible for storing and managing the data, sent from users (i.e., sender and receiver), who are registered in KGC. Once the senders and receivers have been registered in KGC, both of them will be distributed a pair of keys for the further communications.

- **Key Generation Center**: KGC is defined as the fully trusted party in the system, to initialize system parameters and generate master public/secret keys for users. By taking the identity from users as input, it secretly outputs the encryption key and decryption key to senders and receivers via the secure channel.
- **Cloud Server**: CS receives the information from users and performs matching. Then, it returns the successful matching results to the receiver.
- **Sender**: The sender has his/her own unique identity σ. Through the identity σ, the sender can be uniquely designated. In particular, the sender can specify the target receiver in the ciphertext. The sender's identity will not be revealed even if the match fails.
- **Receiver**: The receiver also has his/her own unique identity ρ. Similarly, the receiver can also specify the target sender.

We formally define IBME-DS, which consists of seven algorithms, as follows:

- **Setup**$(1^\lambda) \rightarrow (\mathtt{mpk}, \mathtt{msk})$: Given the security parameter 1^λ, the probabilistic algorithm will output the master public/secret key $\{\mathtt{mpk}, \mathtt{msk}\}$.
- **SKGen**$(\mathtt{msk}, \sigma) \rightarrow ek_\sigma$: Given the sender's identity σ and the master secret key $\mathtt{msk}$, the probabilistic algorithm will output the encryption key ek_σ.

- **RKGen**$(\mathtt{msk}, \rho) \rightarrow dk_\rho$: Given the receiver's identity ρ and the master secret key $\mathtt{msk}$, the probabilistic algorithm will output the decryption key dk_ρ.
- **Enc**$(\mathtt{mpk}, ek_\sigma, \rho_r, m) \rightarrow C$: Given the target receiver's identity ρ_r, the master public key $\mathtt{mpk}$, the encryption key ek_σ, and the message $m \in \{0,1\}^n$, the probabilistic algorithm will output the ciphertext C.
- **MatchTag**$(\mathtt{mpk}, \sigma_s) \rightarrow Tag$: Given the target sender's identity σ_s, the master public key $\mathtt{mpk}$, the probabilistic algorithm will output the match tag Tag.
- **Match**$(\mathtt{mpk}, C, Tag) \rightarrow$ "$\mathtt{accepted}$" or "$\mathtt{failed}$": Given the master public key $\mathtt{mpk}$, the ciphertext C, and the match tag Tag, the deterministic algorithm will output "$\mathtt{accepted}$", if the match occurs. Otherwise, "$\mathtt{failed}$".
- **Dec**$(\mathtt{mpk}, dk_\rho, \sigma_s, C) \rightarrow m$: Given the target sender's identity σ_s, the master public key $\mathtt{mpk}$, the decryption key dk_ρ, and the ciphertext C, the deterministic algorithm will recover the message m, if $\rho = \rho_r$ and $\sigma = \sigma_s$.

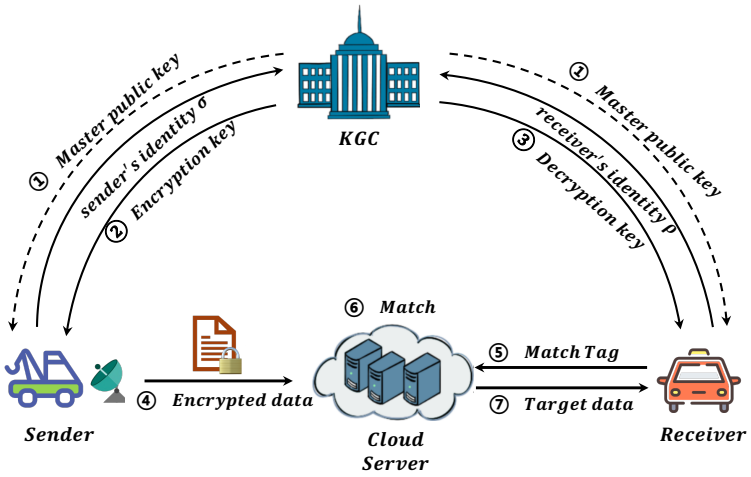

Figure 2. System model of data sharing scheme with bilateral access control.

3.2. Threat Model

KGC is fully trusted and is responsible for generating master public/secret keys and secretly distributing encryption/decryption keys via the secure channel. In our system, CS is considered to be HBC as a semi-trusted party. CS will return correct results to users, but keep curious about users' data. Moreover, both senders and receivers are considered to be untrusted. The sender may pretend to be another sender to generate the ciphertext. The receiver may try to access the unauthorized data. Specifically, we summarize the threat model of IBME-DS as follows:

Type I Adversary: The Type I adversary is the HBC cloud server, which receives the ciphertext message from the sender and the match tag from the receiver. Then, it completes the matching operation based on the received information. In this process, the Type I adversary will launch the ciphertext-only attack to spy on the user's data, designing an access policy.

Type II Adversary: The Type II adversary is the malicious user (sender and receiver), who possesses the encryption key. The malicious users initiate the chosen ciphertext attack to get others' data and the access policy.

3.3. Security Model

Definition 2. *Our proposed IBME-DS is said to be IND-CPA secure if it can resist the probabilistic polynomial-time (PPT) adversary by a game, as follows:*

- **Setup**: The system is established with the input security parameter. Then, the challenger sets the master public/secret key $\{\mathtt{mpk}, \mathtt{msk}\}$.

- **Hash Query**: The adversary requests the hash oracle to get the corresponding hash values for polynomial times.
- **KeyGen Query**: The adversary requests the key generation oracle $\mathcal{O}_{\mathbf{SKGen}}(\mathtt{msk}, \cdot)$ and $\mathcal{O}_{\mathbf{RKGen}}(\mathtt{msk}, \cdot)$ for polynomial times to obtain the corresponding ek_i, dk_j, respectively.
- **Challenge**: The adversary claims m_0, m_1 to be challenged, and provides two instances $I_0 = (m_0, \rho_{r,0}, \sigma_0)$ and $I_1 = (m_1, \rho_{r,1}, \sigma_1)$. Then, the challenger chooses $b \in \{0, 1\}$ randomly. By running **Enc** algorithm, the challenger can compute $C^* = \mathbf{Enc}(ek_{\sigma_b}, \rho_{r,b}, m_b)$. Finally, C^* will be sent to the adversary.
- **Guess**: After receiving C^* from the challenger, the adversary outputs a guess b' on b.

If the adversary can give a correct guess on b, that $b' = b$, we say the adversary can break our proposed IBME-DS. We define the advantage of the adversary breaking the security of IBME-DS as Equation (1):

$$\mathrm{Adv}^{\mathrm{IND-CPA}}_{\mathcal{A},\mathrm{IBME-DS}} = \left| \Pr[\mathrm{b}' = \mathrm{b}] - \frac{1}{2} \right| \leq \epsilon. \tag{1}$$

Definition 3. *Our proposed IBME-DS holds authenticity if it can resist the PPT adversary by a game, as follows.*

- **Setup**: The system is established with the input security parameter. Then, the challenger sets the master public/secret key $\{\mathtt{mpk}, \mathtt{msk}\}$.
- **Hash Query**: The adversary requests the hash oracle to get the corresponding hash values for polynomial times.
- **SKGen Query**: The adversary requests the **SKGen** oracle by inputting σ_i, the challenger returns the sender's encryption key.
- **RKGen Query**: The adversary requests the **RKGen** oracle by inputting ϕ_i, the challenger returns the receiver's decryption key.
- **Forgery**: The adversary sends the tuple (C, ρ, σ') to the challenger, in which σ' has never been input to the **SKGen** oracle. The challenger generates dk_ρ by executing the **RKGen** algorithm. Then, the challenger computes the message m from the ciphertext C by executing the **Dec** algorithm.

If the above game holds, i.e., the message m belongs to the message space M, we say that the adversary can break the authenticity of our proposed IBME-DS. Specifically, the adversary can forge the valid ciphertext even if it is not authorized. We define the advantage of the adversary breaking the authenticity of IBME-DS as Equation (2):

$$\mathrm{Adv}^{\mathrm{Authenticity}}_{\mathcal{A},\mathrm{IBME-DS}} = \Pr[\mathrm{m} = \mathrm{Dec(C)} | \mathrm{m} \in \mathrm{M}] \leq \epsilon. \tag{2}$$

3.4. Design Goal

According to the system model, threat model and security definition defined for our proposed IB-ME-based data sharing scheme, we summarize the design goals to clarify the vital features of our proposed scheme.

- **Security.** The security is to ensure the system is with semantic security under the attack launched by any PPT adversary. The security is the basic demand of the data-sharing scheme in cloud services, which ensures the message m is unknown to others.
- **Privacy.** The privacy is aimed at preventing the stored data and access policy from being revealed to the cloud, even in the matching phase.
- **Authenticity.** The authenticity means that a valid ciphertext under identity σ can only be generated by a valid encryption key ek_σ from KGC. In other words, it guarantees that if a sender with the proper identity can produce a ciphertext, the ciphertext can be decrypted correctly.
- **Bilateral Access Control.** The bilateral access control ensures that the access policy is designed by both the sender and receiver. Compared to the existing access control

scheme, of which the policy is only designed by one side, bilateral access control is a practical requirement for data sharing to cloud services.

4. Concrete Construction

We will describe our proposed IBME-DS, derived from IB-ME. The notations used in IBME-DS are listed in Table 1. Then, we will give a workflow of our proposed IBME-DS, and a concrete construction based on bilinear groups.

Table 1. Notations.

Notation	Description
e	bilinear map
$\mathbb{G}, \mathbb{G}_T$	bilinear groups
p	large prime
g	random number $g \in \mathbb{G}$
H_1, H_2	hash functions
Φ	padding function
n	the length of message
$\mathtt{msk}$	master secret key
$\mathtt{mpk}$	master public key
σ	sender's identity
ρ	receiver's identity
ρ_r	target receiver's identity
ek_σ	sender's encryption key
dk_ρ	receiver's decryption key
m	message
C	ciphertext

4.1. Workflow of IBME-DS

IBME-DS can provide a privacy-preserving data-sharing solution for end devices. The workflow of our scheme includes four phases: system initialization, data updating, user matching and data downloading, as shown in Figure 3.

System Initialization: By running **Setup**, KGC is to generate system parameters. Also, KGC generates encryption/decryption keys for users through **SKGen** and **RKGen**.

Data Upload: The sender runs **Enc** to generate the ciphertext. After that, the sender uploads the ciphertext to CS.

User Matching: Firstly, the receiver will generate the match tag by running **MatchTag** to specify the target sender. The match tag will be sent to CS. After that, CS runs **Match** to find the matched ciphertext, then sends the matched ciphertext to the receiver. During the matching process, CS will get nothing except whether the match occurs.

Data Download: If the matching occurs, the receiver runs **Dec** to decrypt the ciphertext.

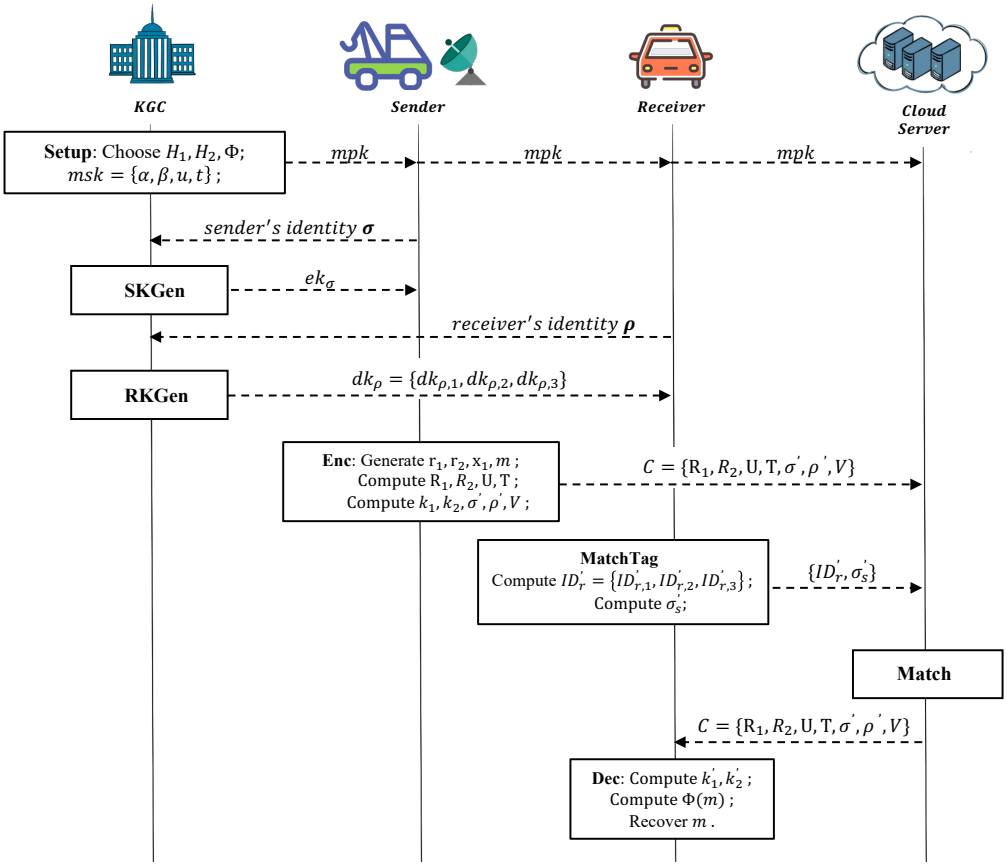

Figure 3. The workflow of IBME-DS.

4.2. IB-ME-Based Data-Sharing Scheme

The detailed construction of our proposed IBME-DS is described as the following Algorithms 1–6.

- **Setup**(1^λ): The system runs this probabilistic algorithm. Taking in the security parameter 1^λ, the system sets $(p, \mathbb{G}, \mathbb{G}_T, g, \hat{g}, e)$, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Then, it selects two hash functions as $H_1 : \{0,1\}^* \to \mathbb{G}$, and $H_2 : \mathbb{G}_T \to \{0,1\}^l$, and a padding function $\Phi : \{0,1\}^n \to \{0,1\}^l$, where Φ is efficiently invertible. Additionally, the system chooses four random numbers $\alpha, \beta, u, t \leftarrow \mathbb{Z}_p$ and sets $g_\alpha = g^\alpha$, $g_\beta = g^\beta, g_u = g^u, \hat{g_\alpha} = g^{\alpha \cdot u}$, and $g_t = g^t$. The system publishes the master public key $\mathtt{mpk} = (e, \mathbb{G}, \mathbb{G}_T, g, g_\alpha, g_\beta, \hat{g_\alpha}, g_u, g_t)$. The master secret key $\mathtt{msk} = (\alpha, \beta, u, t)$ keeps secretly.
- **SKGen**$(\mathtt{msk}, \sigma)$: KGC runs this probabilistic algorithm. Taking in the sender's identity σ, and the master secret key $\mathtt{msk}$, KGC computes the encryption key $ek_\sigma = H_1(\sigma)^\beta$. Then, KGC sends ek_σ to the sender with the identity σ.

Algorithm 1 SKGen$(\mathtt{msk}, \sigma)$

Input: the sender's identity σ, the master secret key $\mathtt{msk}$.
Output: the sender's encryption key ek_σ.
Compute $ek_\sigma = H_1(\sigma)^\beta$.
return the sender's encryption key ek_σ

- **RKGen**$(\mathtt{msk}, \rho)$: KGC runs this probabilistic algorithm. Taking in the receiver's identity ρ, and the master secret key $\mathtt{msk}$, KGC computes the decryption key $dk_\rho = (dk_{\rho,1}, dk_{\rho,2}, dk_{\rho,3})$, as Equation (3):

$$\begin{aligned} dk_{\rho,1} &= H_1(\rho)^\alpha, \\ dk_{\rho,2} &= H_1(\rho)^\beta, \\ dk_{\rho,3} &= H_1(\rho). \end{aligned} \tag{3}$$

  Then, KGC sends dk_ρ to the receiver with the identity ρ.

Algorithm 2 RKGen$(\mathtt{msk}, \rho)$

Input: the receiver's identity ρ, the master secret key $\mathtt{msk}$.
Output: the receiver's decryption key dk_ρ.
Compute $dk_{\rho,1} = H_1(\rho)^\alpha, dk_{\rho,2} = H_1(\rho)^\beta, dk_{\rho,3} = H_1(\rho)$.
return the receiver's decryption key $dk_\rho \leftarrow (dk_{\rho,1}, dk_{\rho,2}, dk_{\rho,3})$.

- **Enc**$(\mathtt{mpk}, ek_\sigma, \rho_r, m)$: The sender runs this probabilistic algorithm. Taking in the target receiver's identity ρ_r, the master public key $\mathtt{mpk}$, the sender's encryption key ek_σ, and the message $m \in \{0,1\}^n$, the sender conducts as the following steps:
  1. Select $r_1, r_2, x_1 \in \mathbb{Z}_p$.
  2. Compute $R_1 = g^{r_1}, R_2 = g^{r_2}$.
  3. Compute $T = g_t^{x_1}$.
  4. Compute Equation (4)

$$\begin{aligned} k_1 &= e(H_1(\rho_r), \hat{g_\alpha}), \\ k_2 &= e(H_1(\rho_r), g_t \cdot ek_\sigma), \\ \sigma' &= H_1(\sigma)^{x_1}, \\ \rho' &= H_1(\rho_r)^{x_1}. \end{aligned} \tag{4}$$

  5. Compute $V = \Phi(m) \oplus H_2(k_1 \cdot e(H_1(\sigma)^{r_1}, g^{x_1})) \oplus H_2(k_2 \cdot e(H_1(\rho_r)^{r_2}, g^{x_1}))$.
  6. Output ciphertext $C = (R_1, R_2, T, \sigma', \rho', V)$.

Algorithm 3 Enc$(\mathtt{mpk}, ek_\sigma, \rho_r, m)$

Input: the target receiver's identity ρ_r, the master public key $\mathtt{mpk}$, the sender's encryption key ek_σ, and the message m.
Output: ciphertext C.
Select random numbers $r_1, r_2, x_1 \in \mathbb{Z}_p$.
Compute $R_1 = g^{r_1}, R_2 = g^{r_2}$.
Compute $T = g_t^{x_1}$.
Compute

$$\begin{aligned} k_1 &= e(H_1(\rho_r), \hat{g_\alpha}), \\ k_2 &= e(H_1(\rho_r), g_t \cdot ek_\sigma), \\ \sigma' &= H_1(\sigma)^{x_1}, \\ \rho' &= H_1(\rho_r)^{x_1}. \end{aligned}$$

Compute $V = \Phi(m) \oplus H_2(k_1 \cdot e(H_1(\sigma)^{r_1}, g^{x_1})) \oplus H_2(k_2 \cdot e(H_1(\rho_r)^{r_2}, g^{x_1}))$.
return ciphertext $C \leftarrow (R_1, R_2, T, \sigma', \rho', V)$.

- **MatchTag**$(\mathtt{mpk}, \sigma_s)$: The receiver runs this probabilistic algorithm. Taking in the target sender's identity σ_s, the master public key $\mathtt{mpk}$, the receiver computes the auxiliary

information as the match tag when the matching is delegated to the CS. The receiver chooses randomly $x_2 \leftarrow \mathbb{Z}_p$. It computes Equation (5) as follows:

$$\begin{array}{rl} ID'_r & = (ID'_{r,1}, ID'_{r,2}, ID'_{r,3}), \\ \sigma_s{}' & = H_1(\sigma_s)^{x_2} \cdot g_t^{x_2}. \end{array} \tag{5}$$

The detailed computations on ID'_r are as Equation (6):

$$\begin{array}{rl} ID'_{r,1} & = dk_{\rho_{r,1}}^{x_2}, \\ ID'_{r,2} & = dk_{\rho_{r,2}}^{x_2}, \\ ID'_{r,3} & = dk_{\rho_{r,3}}^{x_2}. \end{array} \tag{6}$$

The receiver sends $Tag = (ID'_r, \sigma_s{}')$ to the CS.

Algorithm 4 MatchTag$(\mathtt{mpk}, \sigma_s)$

Input: the target sender's identity σ_s, the master public key `mpk`.
Output: the match tag Tag.
Choose randomly $x_2 \leftarrow \mathbb{Z}_p$.
Let $ID'_r = (ID'_{r,1}, ID'_{r,2}, ID'_{r,3})$.
Compute

$$ID'_{r,1} = dk_{\rho_{r,1}}^{x_2},$$

$$ID'_{r,2} = dk_{\rho_{r,2}}^{x_2},$$

$$ID'_{r,3} = dk_{\rho_{r,3}}^{x_2};$$

Compute $\sigma_s{}' = H_1(\sigma_s)^{x_2} \cdot g_t^{x_2}$.
return the match tag $Tag \leftarrow (ID'_r, \sigma_s{}')$.

- **Match**$(\mathtt{mpk}, C, Tag)$: CS runs this deterministic algorithm. Taking in the master public key `mpk`, the ciphertext C, and the match tag Tag, CS executes check Equation (7)

$$e(\rho', \sigma_s{}') \stackrel{?}{=} e(ID'_{r,3}, \sigma' \cdot T). \tag{7}$$

If the equation holds, CS outputs "`accepted`", to indicate the match occurs. Otherwise, "`failed`".

Algorithm 5 Match$(\mathtt{mpk}, C, Tag)$

Input: the master public key `mpk`, the ciphertext C, and the match tag Tag.
Output: the judgment result.
if $e(\rho', \sigma_s{}') \stackrel{?}{=} e(ID'_{r,3}, \sigma' \cdot T)$ **then**
  return "`accepted`".
else
  return "`failed`".
end
return "`accepted`" or "`failed`".

- **Dec**$(\mathtt{mpk}, dk_\rho, \sigma_s, C)$: The receiver runs this deterministic algorithm. Taking in the target sender's identity σ_s, master public key `mpk`, the decryption key dk_ρ, and the ciphertext C, the receiver conducts the following operations:
  1. Parse C as $(R_1, R_2, T, \sigma', \rho', V)$.
  2. Compute
$$\begin{array}{rl} k'_1 & = e(dk_{\rho,1}, g_u), \\ k'_2 & = e(dk_{\rho,2}, H_1(\sigma_s)) \cdot e(dk_{\rho,3}, g_t). \end{array}$$
  3. Compute $\Phi(m) = V \oplus H_2(k'_1 \cdot e(R_1, \sigma')) \oplus H_2(k'_2 \cdot e(R_2, \rho'))$.

4. Recover the message m by the reversibility of $\phi(m)$.

Algorithm 6 Dec$(\mathtt{mpk}, dk_\rho, \sigma_s, C)$

Input: the target sender's identity σ_s, the master public key $\mathtt{mpk}$, the decryption key dk_ρ, and the ciphertext C.
Output: the message m.
Parse C as $(R_1, R_2, T, \sigma', \rho', V)$;
Compute

$$\begin{aligned} k_1' &= e(dk_{\rho,1}, g_u), \\ k_2' &= e(dk_{\rho,2}, H_1(\sigma_s)) \cdot e(dk_{\rho,3}, g_t). \end{aligned}$$

Compute $\Phi(m) = V \oplus H_2(k_1' \cdot e(R_1, \sigma')) \oplus H_2(k_2' \cdot e(R_2, \rho'))$.
return the message $m \leftarrow \phi(m)$.

Correctness. We demonstrate the correctness of IBME-DS from Equations (8)–(10):

$$\begin{aligned} k_1 &= e(H_1(\rho_r), \hat{g_\alpha}) = e(H_1(\rho_r), g^{u\alpha}), \\ k_1' &= e(dk_{\rho,1}, g_u) = e(H_1(\rho)^\alpha, g^u). \end{aligned} \tag{8}$$

The above equations clearly show that $k_1 = k_1'$ if $\rho_r = \rho$. Then, similarly, we can check $k_2 = k_2'$, if $\rho_r = \rho$ and $\sigma_s = \sigma$.

$$\begin{aligned} k_2 &= e(H_1(\rho_r), g_t \cdot ek_\sigma) \\ &= e(H_1(\rho_r), g_t) \cdot e(H_1(\rho_r), ek_\sigma) \\ &= e(H_1(\rho_r), g)^t \cdot e(H_1(\rho_r), H_1(\sigma))^\beta, \\ k_2' &= e(dk_{\rho,2}, H_1(\sigma_s)) \cdot e(dk_{\rho,3}, g_t) \\ &= e(H_1(\rho)^\beta, H_1(\sigma_s)) \cdot e(H_1(\rho), g)^t. \end{aligned}$$

To successfully recover the message, Equations (9) and (10) should hold.

$$H_2(k_1 \cdot e(H_1(\sigma)^{r_1}, g^{x_1})) = H_2(k_1' \cdot e(R_1, \sigma')), \tag{9}$$

$$H_2(k_2 \cdot e(H_1(\rho_r)^{r_2}, g^{x_1})) = H_2(k_2' \cdot e(R_2, \rho')) \tag{10}$$

As we proved before, $k_1 = k_1'$ and $k_2 = k_2'$. We have to ensure $e(H_1(\sigma)^{r_1}, g^{x_1}) = e(R_1, \sigma')$ and $e(H_1(\rho_r)^{r_2}, g^{x_1}) = e(R_2, \rho')$. Therefore, if $\sigma' = H_1(\sigma)^{x_1}$ and $\rho' = H_1(\rho)^{x_1}$, the decryption is successful and the message will be recovered correctly.

5. Security Analysis

Theorem 1. *If the underlying IB-ME is IND-CPA secure, our proposed IBME-DS is secure.*

Proof. Assume that a PPT adversary $\mathcal{A}$ can break IBME-DS, a simulator $\mathcal{B}$ can use $\mathcal{A}$ to break the underlying IB-ME.
Setup: Choose random values $\alpha, \beta, u, t \in \mathbb{Z}_p$ and set $g_\alpha = g^\alpha, g_\beta = g^\beta, g_u = g^u, \hat{g_\alpha} = g^{\alpha \cdot u}$, and $g_t = g^t$. The master secret key is $\mathtt{msk} = (\alpha, \beta, u, t)$ and the master public key is the tuple $\mathtt{mpk} = (p, \mathbb{G}, \mathbb{G}_T, e, g, g_\alpha, g_\beta, \hat{g_\alpha}, g_u, g_t, H_1, H_2)$. Then, the $\mathtt{mpk}$ is sent to $\mathcal{B}$. $\mathcal{B}$ selects a random value $\beta \in \mathbb{Z}_p$. Then, $\mathcal{B}$ sends system parameters $(p, \mathbb{G}, \mathbb{G}_T, e, g, g_\alpha, \hat{g_\alpha}, g_u, g_t, H_1, H_2)$ to $\mathcal{A}$. And the padding function Φ is under control.
H_1 Queries: $\mathcal{B}$ performs the hash queries on H_1 to construct hash table list $\mathcal{L}_1$:

1. If query ρ_i has been requested before, that the query can be found in $(\rho_i, Q_i, \gamma_i, d_i) \in \mathcal{L}_1$, $\mathcal{B}$ returns Q_i. Otherwise, $\mathcal{B}$ generates a coin d_i, $\Pr[d_i = 0] = \delta$.
2. If $d_i = 0$, $\mathcal{B}$ chooses $\gamma_i \in \mathbb{Z}_p$ and computes $Q_i = g^{\gamma_i}$. Then, add $(\rho_i, Q_i, \gamma_i, 0)$ to $\mathcal{L}_1$. Otherwise, $\mathcal{B}$ sets $Q_i = g^{x\gamma_i}$, and adds $(\rho_i, Q_i, \perp, 1)$ to $\mathcal{L}_1$, where x is unknown to $\mathcal{B}$.
3. Return Q_i.

H_2 Queries: $\mathcal{B}$ performs the hash queries on H_2. The hash list $\mathcal{L}_2 = \{res_i, Z_i\}$ is constructed by $\mathcal{B}$. If res_i was already queried, $\mathcal{B}$ returns the value Z_i. Otherwise, $\mathcal{B}$ chooses $Z_i \in \{0,1\}^l$. Then, it adds $\{res_i, Z_i\}$ to $\mathcal{L}_2$. Finally, $\mathcal{B}$ returns Z_i to $\mathcal{A}$.
KeyGen Queries: Upon σ_i being the input, $\mathcal{B}$ obtains $H_1(\sigma_i) = Q_i$ by (σ_i, Q_i) from $\mathcal{L}_1$, and returns $(Q_i)^{\beta}$. Upon ρ_i being the input, $\mathcal{B}$ obtains $H(\rho_i) = Q_i$ from $\mathcal{L}_1$. If $d_i = 0$, $\mathcal{B}$ returns $dk_{\rho_i} = ({Q_i}^{\alpha}, {Q_i}^{\beta}, Q_i)$. Otherwise, $\mathcal{B}$ terminates the game.
Challenge: $\mathcal{A}$ sends $(m_0, m_1, \rho_{r,0}, \rho_{r,1}, \sigma_0, \sigma_1)$ to $\mathcal{B}$, where $\rho_{r,0} = \rho_0$, $\rho_{r,1} = \rho_1$. Then $\mathcal{B}$ performs as follows:

1. $\mathcal{A}$ queries $H_1(\rho_0) = Q_0$ and $H_1(\rho_1) = Q_1$. Let $d_0 = 1$ and $d_1 = 1$, it means that $Q_i = g^{x\gamma_i}$.
2. $\mathcal{B}$ selects a random $r_1, r_2, x_1 \in \mathbb{Z}_p$.
3. $\mathcal{B}$ sends $C^* = (R_1, R_2, T, \sigma', \rho', V)$ to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs a b' as a guess on b.

If $b' = b$ holds, we say that $\mathcal{A}$ breaks the security of our proposed scheme. Then, $\mathcal{B}$ can make use of the result from $\mathcal{A}$ to break the underlying IB-ME. $res_i / e(H_1(\sigma)^{r_1}, g^{x_1})$ and $res_i / e(H_1(\rho_r)^{r_2}, g^{x_1})$ are used in IB-ME. If $\mathcal{A}$ can tell C^* is computed from m_0, m_1, $\mathcal{A}$ also can tell that in IB-ME with the same advantage.

Moreover, in **MatchTag** and **Match**, $\{ID', {\sigma_s}'\}$ are computed with a randomness x_2. From the view of $\mathcal{A}$, the distribution of these elements is indistinguishable from the random elements. If $\mathcal{A}$ can tell the difference, then $\mathcal{A}$ can solve the discrete logarithm problem. □

Theorem 2. *Our proposed IBME-DS holds authenticity, if the bilinear Diffie–Hellman (BDH) problem is hard.*

Proof. Suppose that a PPT adversary $\mathcal{A}$ breaks the authenticity of IBME-DS, a simulator $\mathcal{B}$ is able to use $\mathcal{A}$ to break BDH problem with non-negligible advantage. Receiving the challenge (g, g^a, g^b, g^c), $\mathcal{B}$ computes $D = e(g,g)^{abc}$.
Setup: $\mathcal{B}$ sends the public system parameters $(p, \mathbb{G}, \mathbb{G}_T, e, g, g_\alpha, \hat{g_\alpha}, g_u, g_t, H_1, H_2)$ to $\mathcal{A}$, where two hash functions H_1 and H_2 are under control.
H_1 Queries: $\mathcal{B}$ performs the hash queries on H_1 to construct hash table list $\mathcal{L}_1$:

1. If query ρ_i has been requested before, that the query can be found in $(\rho_i, Q_i, \gamma_i, d_i) \in \mathcal{L}_1$, $\mathcal{B}$ returns Q_i. Otherwise, $\mathcal{B}$ generates a coin d_i randomly, $\Pr[d_i = 0] = \delta$.
2. If $d_i = 0$, $\mathcal{B}$ randomly selects $\gamma_i \in \mathbb{Z}_p$ and computes $Q_i = g^{\gamma_i}$. Otherwise, $\mathcal{B}$ computes $Q_i = g^{c\gamma_i}$. Then, add $(\rho_i, Q_i, \gamma_i, d_i)$ to $\mathcal{L}_1$.
3. Finally, send Q_i to $\mathcal{A}$.

H_1 Queries: $\mathcal{B}$ performs the hash queries on H_1 to construct the hash table list $\mathcal{L}_2$:

1. If query σ_i has been requested before, then the query can be found in $(\sigma_i, Q_i, \gamma_i, d_i) \in \mathcal{L}_2$, $\mathcal{B}$ returns Q_i. Otherwise, $\mathcal{B}$ generates a coin d_i randomly, $\Pr[d_i = 0] = \delta$.
2. If $d_i = 0$, $\mathcal{B}$ randomly selects $\gamma_i \in \mathbb{Z}_p$ and computes $Q_i = g^{\gamma_i}$. Otherwise, $\mathcal{B}$ computes $Q_i = g^{a\gamma_i}$. Then, add $(\rho_i, Q_i, \gamma_i, d_i)$ to $\mathcal{L}_2$.
3. Finally, send Q_i to $\mathcal{A}$.

H_2 Queries: $\mathcal{B}$ performs the hash queries on H_2. The hash list $\mathcal{L}_3 = \{X_i, h_i\}$ is constructed by $\mathcal{B}$. If X_i was already queried, $\mathcal{B}$ returns the value h_i. Otherwise, $\mathcal{B}$ randomly chooses $h_i \in \{0,1\}^l$. Then, it adds $\{X_i, h_i\}$ to $\mathcal{L}_2$. Finally, $\mathcal{B}$ returns h_i to $\mathcal{A}$.
SKGen Queries: Input σ_i, $\mathcal{B}$ obtains $H_1(\sigma_i) = Q_i$ by (σ_i, Q_i) from $\mathcal{L}_2$. If $d_i = 0$, $\mathcal{B}$ returns $(ek_{\sigma_i}) = g^{b\gamma_i}$; otherwise, $\mathcal{B}$ aborts.
RKGen Queries: Input ρ_i, $\mathcal{B}$ obtains $H_1(\rho_i) = Q_i$ by (ρ_i, Q_i) from $\mathcal{L}_1$. If $d_i = 0$, $\mathcal{B}$ returns $(dk_{\rho_i}) = (g^{a\gamma_i}, g^{b\gamma_i}, Q_i = g^{\gamma_i})$; otherwise, $\mathcal{B}$ aborts.
Forgery: $\mathcal{A}$ sends the tuple (C, ρ, σ') to $\mathcal{B}$. Let $\sigma' = \sigma$, $\mathcal{B}$ performs as follows:

1. $\mathcal{B}$ queries $H_1(\rho) = Q_0$ and $H_1(\sigma) = Q_1$.
2. If both the tuples $(\rho, Q_0, \gamma_0, d_0) \in \mathcal{L}_1$ and $(\sigma, Q_1, \gamma_1, d_1) \in \mathcal{L}_2$ without $d_0 = 1$ and $d_1 = 1$, $\mathcal{B}$ aborts. If not, $dk_{\rho,2} = g^{cb\gamma_0}$, $H(\sigma) = g^{a\gamma_1}$, $H_2(k_2' \cdot e(R_2, \rho)) =$

$H_2(e(dk_{\rho,2}, H_1(\sigma))e(dk_{\rho_3}, g_t), e(R_2, \rho))$, where $e(dk_{\rho,2}, H_1(\sigma)) = e(g^{cb\gamma_0}, g^{a\gamma_1}) = D^{\gamma_0\gamma_1}$, and $Q_0 = dk_{\rho,3}$.

3. $\mathcal{B}$ parses C as (R_1, R_2, T, V), computes $z = 1/(\gamma_0\gamma_1)$ and selects a random tuple (X_i, h_i).
4. Return $D' = (X_i \cdot e(Q_0, g_t)^{-1} \cdot e(R_2, \rho)^{-1})^z$.

If the above simulation holds, we say that $\mathcal{A}$ breaks the authenticity of our proposed scheme. $\mathcal{A}$ can forge the valid ciphertext even if it is not authorized, as it can solve the BDH problem.

Next, we assume that the adversary makes q_R and q_S queries to oracle **SKGen** and **RKGen**, and analyze the advantage that $\mathcal{B}$ outputs the solution to the BDH assumption. By the above proof, the probability that $\mathcal{B}$ does not abort for any of these calls is $\delta^{q_R+q_S}$ and the probability that $\mathcal{B}$ does not abort in the forgery phase is $(1-\delta)^2$. Thus, the total probability that $\mathcal{B}$ does not abort is $\delta^{q_R+q_S}(1-\delta)^2$ and will get the maximum value when $\delta = (q_R + q_S)/(q_R + q_S + 2)$ which is $4/e^2(q_R + q_S + 2)^2$. If $\mathcal{B}$ does not abort, it outputs the correct solution D' with a probability at least $2\epsilon/q_{H_2}$. Hence, $\mathcal{B}$ can solve the BDH problem with advantage $8\epsilon/e^2(q_R + q_S + 2)^2 q_{H_2}$. □

Theorem 3. *Our proposed IBME-DS holds privacy preservation, if the DL problem is hard.*

Proof. The privacy preservation of IBME-DS lies in the fact that no one can obtain any private information (i.e., the outsourced data and access policy) from the communication by intercepting or making any unauthorized modifications.

From the data privacy aspect, IBME-DS encrypts the data using the user's secret encryption key. According to the security of IBME-DS, the encrypted data is a string of random characters from the view of the adversary. Therefore, any PPT adversary cannot get the data contained in the system.

From the aspect of the sender's identity and access policy, even though receivers can specify the sender σ to generate the sender's corresponding hash $H_1(\sigma)$, it is impossible to further calculate the sender's secret key $ek_\sigma = H_1(\sigma)^\beta$ due to the hardness of the DL problem. The designed access policy is embedded in the ciphertext as $H_1(\rho_r)^{x_1}$.

From the aspect of the receiver's identity and access policy, the identity and access policy are hidden in dk_ρ, which should be kept secretly by receivers. During the matching process, receivers will hide their secret decryption key by randomly choosing parameter x_2.

Therefore, any PPT adversary cannot find out the certain identity and access policy of the sender and receiver. We complete the proof that IBME-DS holds privacy preservation, if the DL problem is hard. □

Theorem 4. *There does not exist any PPT adversary who can forge the match tag for the match algorithm in IBME-DS, if the CDH assumption holds.*

Proof. With the master public key `mpk`, the ciphertext C and the match tag Tag, the cloud server can get $C = (R_1, R_2, T, \sigma', \rho', V) = (g^{r_1}, g^{r_2}, H_1(\sigma)^{x_1}, H_2(\rho_r)^{x_1}, V)$ and $Tag = (ID'_r, {\sigma_s}') = ((dk_{\rho_{r,1}}^{x_2}, dk_{\rho_{r,2}}^{x_2}, dk_{\rho_{r,3}}^{x_2}), H_1(\sigma_s)^{x_2} \cdot g_t^{x_2})$. However, since the random numbers r_1, r_2, x_1, x_2 are unknown to the cloud server, the cloud server cannot retrieve any sensitive information about σ, ρ_r, σ_s from C and Tag, according to the hardness of the CDH problem. Furthermore, for the final result of match $e(\rho', {\sigma_s}') \stackrel{?}{=} e(ID'_{r,3}, \sigma' \cdot T)$, its construction is based on the BDH problem. Hence, the cloud server only knows what we expect to disclose, which is whether the equation holds. We complete the proof that there does not exist any PPT adversary who can forge the match in IBME-DS, if the CDH assumption holds. □

6. Theoretical Analysis and Performance Evaluation

In this section, we present the comprehensive theoretical analysis of computational overhead. Then, we conduct extensive experiments to evaluate our proposed IBME-DS, comparing it with the two most related works, AFNV19 [17], called ME, and CFDS20 [18], called MABE. Specifically, AFNV19 [17] is the first paper to propose the matchmaking

encryption algorithm, and in this paper, the IB-ME scheme is implemented. CFDS20 is an improvement on AFNV19 that implements the matchmaking encryption and ME-based data-sharing scheme using fog computing, which allows bilateral access control. These two schemes are similar to our scheme in terms of construction and implemented functionality. Therefore, we have selected these schemes to compare computational cost and communication overhead.

6.1. Theoretical Analysis

We theoretically compare the computational overhead of our proposed IBME-DS with the existing similar schemes, in terms of key generation, encryption and decryption.

Specifically, we compare the computational overhead of our proposed IBME-DS with [17–19] intuitively. We choose the main cryptographic operations from algorithms, such as bilinear pairing and multiplication operation, as the important references to measure the computational overhead of similar schemes. Our scheme takes E_G, $2E_G$, $4P + 8E_G + M_G + 2M_{G_T}$, $3P + 3M_{G_T}$ for the **SKGen**, **RKGen**, **Enc** and **Dec**, respectively. Moreover, only the computational overhead of the encryption and decryption algorithm is slightly higher than that of IB-ME [17]. Based on the above analysis, the comparison of computational overhead is as shown in Table 2.

Table 2. Theoretical analysis of computational overhead.

Scheme	Approach	SKGen	RKGen	Enc	Dec
Damgaard [19]	ACE	$n_s E_G$	$n_r E_G$	$n_s(5E_G + M_G)$	$n_r(E_G + M_G)$
Ateniese [17]	IB-ME	E_G	$2E_G$	$2P + 3E_G + M_G$	$3P + M_{G_T}$
Xu [18]	MABE	$n_S(E_G + M_G) + 2E_G$	$n_R(3E_G + M_G)$	$(n_R + n_{S'} + 3)E_G + n_{S'}M_{G_T})$	$n_{R'}(2P + E_G) + M_{G_T}$
Our scheme	IB-ME	E_G	$2E_G$	$4P + 8E_G + M_G + 2M_{G_T}$	$3P + 3M_{G_T}$

SKGen: the sender's encryption key generation algorithm; **RKGen**: the receiver's decryption key generation algorithm; **Enc**: the encryption algorithm; **Dec**: the decryption algorithm; $\boldsymbol{E_G}$: the computational cost of an exponential function in $\mathbb{G}$; $\boldsymbol{M_G}$: the computational cost of a multiplication function in $\mathbb{G}$; $\boldsymbol{M_{G_T}}$: the computational cost of a multiplication function in $\mathbb{G}_{\mathbb{T}}$; $\boldsymbol{P}$: the computational cost of pairing; $\boldsymbol{n_s}$: the number of senders; $\boldsymbol{n_r}$: the number of receivers; $\boldsymbol{n_S}$: the number of the sender's attributes; $\boldsymbol{n_R}$: the number of the receiver's attributes.

6.2. Experimental Setting

As is shown in Table 3, the configuration is with a laptop running 64-bit Windows 10 with Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz and 8 GB RAM. We implement the proposed scheme using JAVA 14.0.2 on the laptop using the JPBC library. Type A elliptic curves are used to implement the experiment, by initializing the system parameters from "a.properties". To make the experimental data more accurate, we average the resulting times after running each algorithm 100 times. To verify the feasibility of the scheme, we simulate a multi-user scenario by setting different numbers of senders. We choose two related schemes based on ME: IB-ME [17] and MABE [18].

Dataset: In our experiments, we combine the real-world dataset and the simulated dataset to evaluate the performance of IBME-DS. Specifically, the real data are obtained from a real-world dataset of OpenITS https://www.openits.cn/openData2/746.jhtml, consisting of the traffic data (including traffic signal control data, section travel time data and intersection lane traffic volume data) collected on 15 December 2016, in Anhui, China. The simulated dataset mainly comprises the vehicle information (consisting of the license plate number, geolocation and speed). These traffic data are used to represent the transmitted messages, and the identity of the user is set by license plate number or road ID. We implement our proposed IBME-DS to simulate data sharing among users based on the above dataset. The total number of users in the experiment will be capped at 30, and the number of attributes will be set to 5.

Table 3. Experimental configuration.

Hardware Configuration	
CPU	Intel(R)Core(TM)i5-6200U@2.30 GHz
RAM	8.00 GB
Software Configuration	
OS	64-bit Windows 10
Language	Java 14.0.2
Library	JPBC-2.0.0
Elliptic Curve	
Type	Type A
Security Level	AES-80
Based Field Size	512 bits
Group Order Size	160 bits

6.3. Computational Cost

The first experiment is to obtain the runtime of system initialization. The second experiment is to obtain the runtime of key generation. The third and fourth experiments are to obtain the runtime for encryption and decryption. The fifth experiment is to obtain the runtime for the matching process.

In Figure 4, the experiment shows the running time required for **Setup** versus the number of senders. The experimental results indicate that the initialization computational cost of our scheme is between the MABE and IB-ME. Compared with IB-ME, our scheme has a slightly higher initialization time cost, for the reason that two additional parameters are set to ensure that the matching process is safely outsourced to the cloud server. Moreover, MABE needs two additional pairing calculations during the system initialization, so the time cost is higher than our scheme.

In Figure 5, the experiment shows the running time required for **SKGen** and **RKGen**. The experimental results indicate that our scheme takes about the same amount of time as IB-ME. As the characteristics of the identity-based scheme and the appropriate access control policy settings, our scheme achieves less computation cost than MABE. And as the number of senders increases, the gap becomes more pronounced.

In Figure 6, the experiment shows the running time required for **Enc**. The experimental results indicate that the MABE takes more time than IB-ME and ours, as directly scaling from an attribute-based setting to an identity would incur high computational cost. Our scheme adds two pairing computations to delegate the matching process to the cloud server. Thus, the computational cost of data encryption in our proposed IBME-DS is slightly higher than in IB-ME.

In Figure 7, the experiment shows the running time required for **Dec**. The experimental results indicate that we spend less time on decryption than MABE. Compared with IB-ME, the computational cost of data decryption in our scheme is slightly larger than that in IB-ME for a single decryption process. In IB-ME, the receivers require to determine whether the message is sent from the designed senders when they receive each ciphertext. In practical applications, such an operation will cause the waste of huge computation and communication resources on the receiver side.

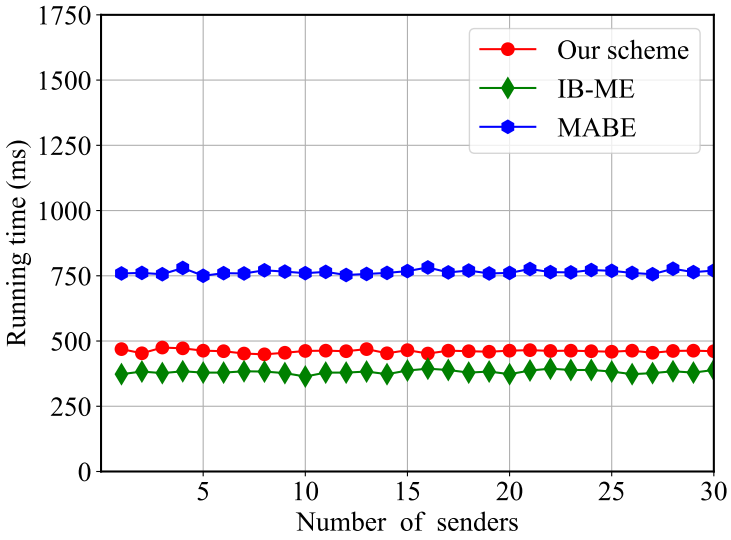

Figure 4. Running time for setup.

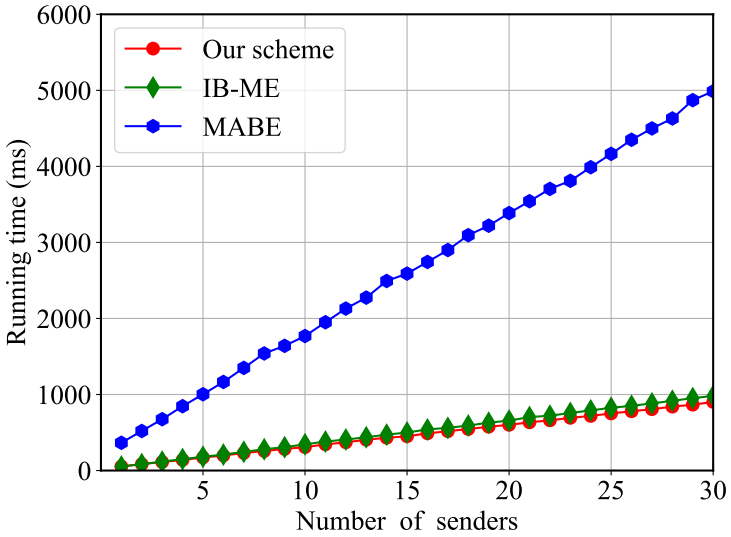

Figure 5. Running time for key generation.

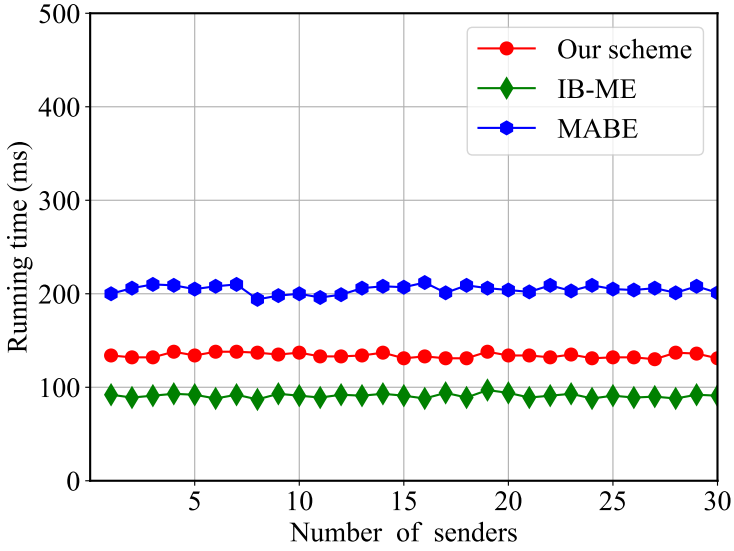

Figure 6. Running time for encryption.

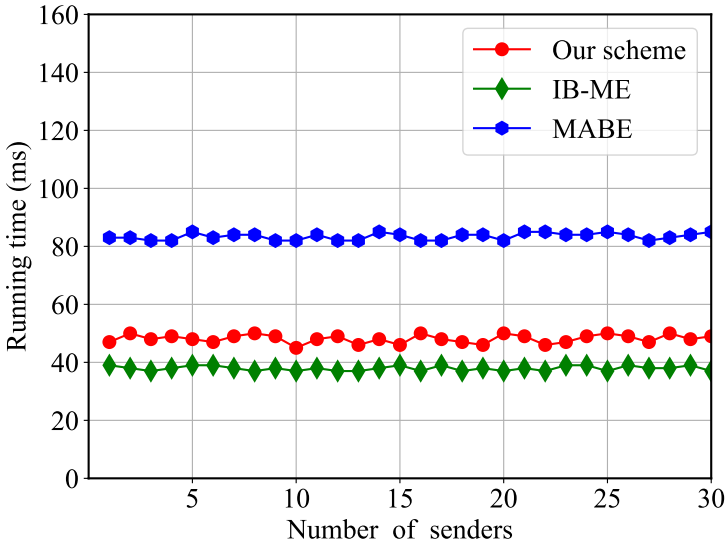

Figure 7. Running time for decryption.

In Table 4, the experiment shows the running time required for **Match** versus the number of senders. The experimental results indicate that the computational cost of the matching process grows linearly with the number of senders. By delegating the matching process to the cloud server, the computation cost on the local side will be significantly reduced.

Table 4. Running time for match.

Experimental Performance						
Number of senders	5	10	15	20	25	30
Running time (ms)	515	1010	1543	2047	2679	3058

Note: to verify the practicality of the scheme, we conduct a series of experiments, including **Setup**, **SKGen**, **RKGen**, **Enc**, **Dec** and **Match**. We develop a matching mechanism to outsource the matching process to the cloud server to reduce the computational and communication overhead on the user side. As in the theoretical analysis of the scheme, the experiments show that our algorithm has slightly higher computational overhead in **Setup**, **SKGen**, **RKGen**, **Enc** and **Dec** than IB-ME, since we build an outsourced matching mechanism and add E_G operations to ensure data privacy. Although we have more time overhead compared to IB-ME, our scheme is capable of meeting practical requirements for cloud services.

6.4. Communication Overhead

In Table 5 and Figure 8, the experiment shows the communication overhead on the receiver and the sender side in terms of the encryption key, decryption key and ciphertext. The experimental results indicate that by delegating the matching process to the cloud server, the communication overhead on the receiver side will be significantly reduced than in others, although our scheme has slightly more communication overhead on the ciphertext than IB-ME.

Table 5. Space cost of different schemes.

Scheme	**Encryption Key**	**Decryption Key**	**Ciphertext**
Ateniese19 [17]	512	1536	1280
Xu22 [18]	3072	5120	7168
Our scheme	512	1536	2816

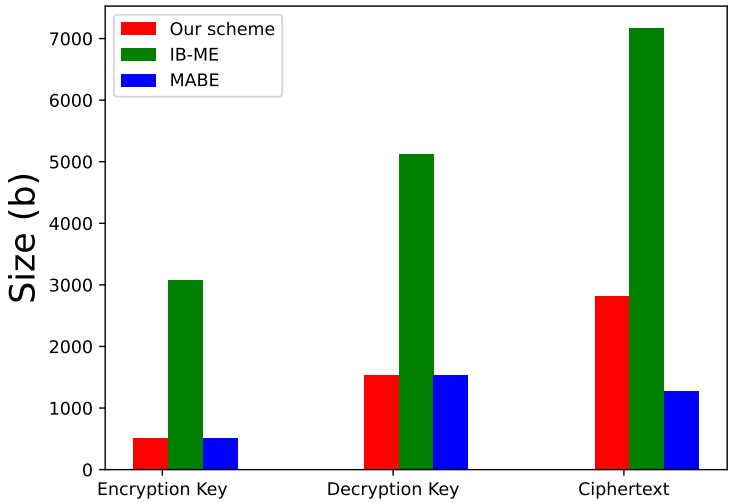

Figure 8. Communication overhead of different schemes.

7. Related Work

We will describe the state-of-the-art works on access control and matchmaking encryption in this section. As is shown in Table 6, we compare the existing schemes with our scheme from different perspectives.

Table 6. Comprehensive comparison among the state-of-the-art schemes.

	Type of Scheme	Data Privacy	Sender Privacy	Outsource Match	Identity-Based Encryption	Access Control
[20]	**ABAC**	×	×	×	×	one-way
[21–23]	**ABE**	✓	✓	×	×	one-way
[19]	**ACE**	✓	×	✓	×	bilateral
[17]	**ME**	✓	✓	×	×	bilateral
[24,25]	**ME**	✓	✓	✓	×	bilateral
[26]	**CL-ME**	✓	✓	×	✓	bilateral
[18,27]	**AB-ME**	✓	✓	✓	×	bilateral
[28,29]	**IB-ME**	✓	✓	×	✓	bilateral
Ours	**IB-ME**	✓	✓	✓	✓	bilateral

7.1. Access Control

In 1969, Lampson et al. [30] first proposed access control technology and introduced the concept of subject and object and autonomous access control. The owner of autonomous access control can independently decide to grant object permissions to other subjects. To realize an access control in the cross-domain network environment, Eric et al. [20] described a dynamic Attribute-Based Access Control (ABAC). ABAC takes attributes as the core and authorization basis, providing fine-grained authorized access and large-scale user dynamic expansion in complex network environments. Specifically, Attribute-Based Encryption (ABE) was first introduced by Sahai and Waters [21] in the fuzzy identity encryption scheme in 2006, which aimed to improve the security of attribute-based access control. Subsequently, Goyal et al. [22] divided ABE into Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE). But none of them satisfy the protection of sender privacy. To solve the above problem, Zheng et al. [23] first introduced the Attribute-based Keyword Search (ABKS), allowing the user to search for encrypted data outsourced by the data owner. However, their scheme cannot support multiple rounds to search the messages securely. Damgard et al. [19] proposed Access Control Encryption (ACE) based on security policies for information flow to allow fine-grained access control. In ACE, different authorizations

will be given to users for reading and writing. However, to ensure secure data flow, ACE needs to introduce the sanitizer that controls the communication, which is a fully trusted party. In terms of our literature review on conventional access control, the aforementioned schemes have two crucial issues when applied to cloud services. Firstly, most access control schemes only support that one side specifies the access policy (e.g., CP-ABE, KP-ABE). Secondly, even if some access control scheme (e.g., ACE) is equipped with bilateral access control, it has to involve a fully trusted party. The strong trusted requirement will throw threats to the system security.

7.2. Matchmaking Encryption

Ateniese et al. [17] proposed the matchmaking encryption (ME) on CRYPTO'19. In their work, an efficient identity-based matchmaking encryption scheme (IB-ME) is given as an instantiation. Both senders and receivers are allowed to specify their access policies simultaneously, which realizes bilateral access control. But none of the schemes proposed in this paper support outsourced matching, which may become a performance bottleneck for devices with limited resources. To support the outsourced matching, Li et al. [27] proposed a bilateral friend-matching scheme by combining ME with ABE. However, the outsourcing center in their scheme can identify which party does not meet the access policy of the others. Chen et al. [26] proposed an efficient certificateless matchmaking encryption (CL-ME) scheme for IoT. In the cloud-fog computing environment, Xu et al. [24] proposed a data-sharing system based on lightweight ME. Subsequently, Xu et al. [18] proposed a data-sharing system specified for cloud-fog devices (CFDS), based on matchmaking attribute-based encryption (MABE). In the field of IIoT, Sun et al. [25] proposed a privacy-preserving bilateral access control scheme, which is based on fine-grained access control and ME. In 2022, Wu et al. [29] extended ME to cross-domain scenarios and proposed a cross-domain IB-ME, by which users from different domains can get their secret keys from different authority centers. However, the aforementioned ME-based works are constructed under the attribute-based cryptosystem, with huge computation overhead to the resource-restricted devices. Moreover, Danilo et al. [28] proposed an IB-ME scheme based on standard assumption. The matching is conducted by users while decrypting the ciphertext. Also, the resource-restricted devices are not competent to resolve the complex computation and huge communication overhead. Hence, there is a certain gap in implementing a privacy-preserving data-sharing scheme with bilateral access control, supporting the matching delegation for devices in cloud services.

8. Discussion

In this section, we focus on discussing the advantages of our research and the limitations of it. The advantages of our proposed scheme are obvious by reviewing the related works as shown in Table 6. Our proposed IBME-DS possesses the semantic security, authenticity and data privacy, where the delegation will not reveal sensitive information (e.g., access policies and shared data) for cloud services. As a new way to protect both receiver and sender access control simultaneously, ME enables bilateral access control [17]. We construct our scheme based on IB-ME to fulfill the practical requirements of identity authentication in cloud services. To enhance the security, we design a privacy-preserving matching mechanism to realize the secure outsource match. There is a limitation to be improved, which concerns the communication overhead that needs to be further reduced during large-scale deployment of cloud services.

9. Conclusions

In this work, we propose a data-sharing scheme with bilateral access control, based on IB-ME. Our scheme can provide effective privacy preservation for both the sender and receiver in cloud services. Due to delegating the matching process to the cloud server, our scheme can provide more efficient data-sharing capabilities, and reduce the computation overhead on end devices without revealing any private information. Additionally, we point

out some interesting issues to be resolved in the future. Firstly, we consider protecting forward secrecy by applying the key exposure resistant cryptographic primitives. Secondly, our scheme is designed for devices in cloud services. We will attempt to further enhance the efficiency of our scheme by applying the edge computing technique [31].

Author Contributions: Conceptualization, T.W.; Methodology, X.M.; Validation, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (grant Nos. 62102027 and 62202444).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ganapathy, D.N.; Joshi, K.P. A Semantically Rich Framework to Automate Cloud Service Level Agreements. *IEEE Trans. Serv. Comput.* **2022**, *16*, 53–64. [CrossRef]
2. Zhang, C.; Zhu, L.; Xu, C.; Ni, J.; Huang, C.; Shen, X. Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing. *IEEE Trans. Mob. Comput.* **2021**, *21*, 4410–4425. [CrossRef]
3. Mahmoud, M.M.; Rodrigues, J.J.; Ahmed, S.H.; Shah, S.C.; Al-Muhtadi, J.F.; Korotaev, V.V.; De Albuquerque, V.H.C. Enabling technologies on cloud of things for smart healthcare. *IEEE Access* **2018**, *6*, 31950–31967. [CrossRef]
4. Rajeswari, S.; Suthendran, K.; Rajakumar, K. A smart agricultural model by integrating IoT, mobile and cloud-based big data analytics. In Proceedings of the 2017 International Conference on Intelligent Computing and Control (I2C2), Coimbatore, India, 23–24 June 2017; pp. 1–5.
5. Jiang, D. The construction of smart city information system based on the Internet of Things and cloud computing. *Comput. Commun.* **2020**, *150*, 158–166. [CrossRef]
6. Wang, X.; Li, Z. Traffic and Transportation Smart with Cloud Computing on Big Data. *Int. J. Comput. Sci. Appl.* **2016**, *13*, 1–16.
7. Meixner, F.; Buettner, R. Trust as an integral part for success of cloud computing. In Proceedings of the International Conference on Internet and Web Applications and Services, Stuttgart, Germany, 27 May–1 June 2012; pp. 207–214.
8. Esposito, C.; Castiglione, A.; Martini, B.; Choo, K.K.R. Cloud manufacturing: Security, privacy, and forensic concerns. *IEEE Cloud Comput.* **2016**, *3*, 16–22. [CrossRef]
9. Zhang, Y.; Xu, C.; Li, H.; Yang, K.; Zhou, J.; Lin, X. HealthDep: An efficient and secure deduplication scheme for cloud-assisted eHealth systems. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4101–4112. [CrossRef]
10. Zhou, Y.; Zheng, S.; Wang, L. Privacy-preserving and efficient public key encryption with keyword search based on CP-ABE in cloud. *Cryptography* **2020**, *4*, 28. [CrossRef]
11. Miao, Y.; Ma, J.; Liu, X.; Li, X.; Jiang, Q.; Zhang, J. Attribute-based keyword search over hierarchical data in cloud computing. *IEEE Trans. Serv. Comput.* **2017**, *13*, 985–998. [CrossRef]
12. Zhang, C.; Zhao, M.; Zhu, L.; Zhang, W.; Wu, T.; Ni, J. FRUIT: A blockchain-based efficient and privacy-preserving quality-aware incentive scheme. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3343–3357. [CrossRef]
13. Chatterjee, A.; Sengupta, I. Sorting of fully homomorphic encrypted cloud data: Can partitioning be effective? *IEEE Trans. Serv. Comput.* **2017**, *13*, 545–558. [CrossRef]
14. Xu, C.; Wang, N.; Zhu, L.; Sharif, K.; Zhang, C. Achieving searchable and privacy-preserving data sharing for cloud-assisted E-healthcare system. *IEEE Internet Things J.* **2019**, *6*, 8345–8356. [CrossRef]
15. Wang, N.; Yang, W.; Wang, X.; Wu, L.; Guan, Z.; Du, X.; Guizani, M. A blockchain based privacy-preserving federated learning scheme for Internet of Vehicles. *Digit. Commun. Netw.* 2022, *in press*. [CrossRef]
16. Nasiraee, H.; Ashouri-Talouki, M. Anonymous decentralized attribute-based access control for cloud-assisted IoT. *Future Gener. Comput. Syst.* **2020**, *110*, 45–56. [CrossRef]
17. Ateniese, G.; Francati, D.; Nuñez, D.; Venturi, D. Match me if you can: Matchmaking encryption and its applications. *J. Cryptol.* **2021**, *34*, 1–50. [CrossRef]
18. Xu, S.; Ning, J.; Li, Y.; Zhang, Y.; Xu, G.; Huang, X.; Deng, R.H. Match in my way: Fine-grained bilateral access control for secure cloud-fog computing. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 1064–1077. [CrossRef]
19. Damgård, I.; Haagh, H.; Orlandi, C. Access control encryption: Enforcing information flow with cryptography. In Proceedings of the Theory of Cryptography: 14th International Conference, Beijing, China, 31October–3 November 2016; pp. 547–576.
20. Yuan, E.; Tong, J. Attributed based access control (ABAC) for web services. In Proceedings of the IEEE International Conference on Web Services (ICWS'05), Orlando, FL, USA, 11–15 July 2005.
21. Sahai, A.; Waters, B. Fuzzy identity-based encryption. In Proceedings of the Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005; pp. 457–473.

22. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 30 October–3 November 2006; pp. 89–98.
23. Zheng, Q.; Xu, S.; Ateniese, G. VABKS: Verifiable attribute-based keyword search over outsourced encrypted data. In Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 522–530.
24. Xu, S.; Ning, J.; Ma, J.; Huang, X.; Pang, H.H.; Deng, R.H. Expressive bilateral access control for internet-of-things in cloud-fog computing. In Proceedings of the 26th ACM Symposium on Access Control Models and Technologies, Virtual, 16–18 June 2021; pp. 143–154.
25. Sun, J.; Yuan, Y.; Tang, M.; Cheng, X.; Nie, X.; Aftab, M.U. Privacy-preserving bilateral fine-grained access control for cloud-enabled industrial IOT healthcare. *IEEE Trans. Ind. Inform.* **2021**, *18*, 6483–6493. [CrossRef]
26. Chen, B.; Xiang, T.; Ma, M.; He, D.; Liao, X. CL-ME: Efficient certificateless matchmaking encryption for internet of things. *IEEE Internet Things J.* **2021**, *8*, 15010–15023. [CrossRef]
27. Li, T.; Li, X.; Liu, X. An efficient privacy-preserving bidirectional friends matching scheme in mobile social networks. In Proceedings of the 2019 International Conference on Networking and Network Applications (NaNA), Daegu, Republic of Korea, 10–13 October 2019; pp. 51–57.
28. Francati, D.; Guidi, A.; Russo, L.; Venturi, D. Identity-based matchmaking encryption without random oracles. In Proceedings of the Progress in Cryptology–INDOCRYPT 2021: 22nd International Conference on Cryptology in India, Jaipur, India, 12–15 December 2021; Proceedings 22; Springer: Berlin/Heidelberg, Germany, 2021; pp. 415–435.
29. Wu, A.; Weng, J.; Luo, W.; Yang, A.; Liu, J.N.; Jiang, Z. Cross-domain identity-based matchmaking encryption. *Cryptol. ePrint Arch.* **2022**. Available online: https://eprint.iacr.org/2022/085 (accessed on 20 November 2023).
30. Lampson, B.W. Dynamic protection structures. In Proceedings of the November 18–20, Fall Joint Computer Conference, Las Vegas, NV, USA, 18–20 November 1969; pp. 27–38.
31. Lu, Y.; Maharjan, S.; Zhang, Y. Adaptive edge association for wireless digital twin networks in 6G. *IEEE Internet Things J.* **2021**, *8*, 16219–16230. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.