electronics

MDPI

Article

Conditional Proxy Re-Encryption-Based Key Sharing Mechanism for Clustered Federated Learning

Yongjing Zhang [1], Zhouyang Zhang [2,*], Shan Ji [1], Shenqing Wang [1] and Shitao Huang [3]

[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[2] Research Center for Basic Theories of Intelligent Computing, Research Institute of Basic Theories, Zhejiang Laboratory, Hangzhou 311100, China
[3] School of Computer Science, Nanjing University of Science Information and Technology, Nanjing 210044, China
* Correspondence: zhangzhouyang@zhejianglab.com

Abstract: The need of data owners for privacy protection has given rise to collaborative learning, and data-related issues heterogeneity faced by federated learning has further given rise to clustered federated learning; whereas the traditional privacy-preserving scheme of federated learning using homomorphic encryption alone fails to fulfill the privacy protection demands of clustered federated learning. To address these issues, this research provides an effective and safeguarded answer for sharing homomorphic encryption keys among clusters in clustered federated learning grounded in conditional representative broadcast re-encryption. This method constructs a key sharing mechanism. By combining the functions of the bilinear pairwise accumulator and specific conditional proxy broadcast re-ciphering, the mechanism can verify the integrity of homomorphic encryption keys stored on cloud servers. In addition, the solution enables key management centers to grant secure and controlled access to re-encrypted homomorphic encryption keys to third parties without disclosing the sensitive information contained therein. The scheme achieves this by implementing a sophisticated access tree-based mechanism that enables the cloud server to convert forwarded ciphertexts into completely new ciphertexts customized specifically for a given group of users. By effectively utilizing conditional restrictions, the scheme achieves fine-grained access control to protect the privacy of shared content. Finally, this paper showcases the scheme's security against selective ciphertext attacks without relying on random prediction.

Keywords: key sharing mechanism; clustered federated learning; proxy re-encryption

Citation: Zhang, Y.; Zhang, Z.; Ji, S.; Wang, S.; Huang, S. Conditional Proxy Re-Encryption-Based Key Sharing Mechanism for Clustered Federated Learning. *Electronics* **2024**, *13*, 848. https://doi.org/10.3390/electronics13050848

Academic Editor: Aryya Gangopadhyay

Received: 7 January 2024
Revised: 4 February 2024
Accepted: 16 February 2024
Published: 22 February 2024

1. Introduction

Developments in computing power have been instrumental in the rapid evolution of AI across fields such as finance, healthcare, computer vision, and autonomous driving [1]. At the heart of AI's progress are increasingly sophisticated machine learning algorithms, drawing keen interest from the research community [2]. The efficacy of these algorithms is deeply dependent on access to large volumes of high-quality data. However, the richness of such data often entails privacy-sensitive information, making data owners cautious about sharing it. This caution leads to the formation of data silos, which not only diminish the quality of accessible data, but more critically, limit the availability of comprehensive datasets necessary for advancing machine learning research. Such restrictions pose significant barriers to innovation in machine learning, thereby directly affecting the pace and scope of advancements in AI [3,4].

To address the issue of data isolation, McMahan et al. suggested federated learning [5]. In federated learning, users perform model computation locally using their own datasets, and a server aggregates the individual local models, thereby updating the global model.

Federated learning allows for model training without the exchange of data between users, thus significantly preserving data privacy [5].

However, federated learning requires the exchange of intermediate parameters between the client and the server for collaborative training, and in the process of data exchange, the raw data carried by the intermediate parameters may be exposed to all the training participants, which leads to privacy leakage [6–8]. To solve this problem, homomorphic encryption is widely used to encrypt the data exchange process between the client and the server [9,10]. Homomorphic encryption provides a cryptographic solution that enables the client to encrypt the intermediate parameter and send it to the server, which is able to complete the aggregation operation on the encrypted data without obtaining the plaintext information of the data [11,12]. The client receives the result of the encryption operation and decrypts it, which is the same as the result of the aggregation operation on the plaintext [13–15].

In addition to data privacy issues, data heterogeneity is also an important challenge facing federated learning today. In federated learning, data heterogeneity refers to the differences in distribution, characteristics, and size of data among different participants [16]. These differences may impact the effectiveness of federated learning and the model's effectiveness. To tackle this issue, Ghosh et al. introduced the idea of clustered federated learning [17]. Clustering federated learning divides the participants' data into different clusters by introducing clustering techniques using clustering algorithms (e.g., K-mean clustering or hierarchical clustering) [18]. Within each cluster, the participants share their data for model training, similar to traditional federated learning. After each cluster has trained its own local model, the model parameters can be selectively aggregated together to form a global model [19].

However, cluster federated learning brings new challenges to privacy protection based on homomorphic encryption. It requires that knowledge not be shared between clusters, and in traditional privacy-preserving schemes for federated learning systems using homomorphic encryption, all participants share the same encrypting public key and decrypting key, which results in the data of a participant in one cluster being potentially exposed to participants in other clusters. Therefore, different encryption and decryption keys are required between each cluster.

To this end, we propose a conditional proxy broadcast re-encryption method for distributing and managing homomorphic encryption keys for each cluster in clustered federated learning. By using proxy broadcast re-encryption, keys for homomorphic encryption produced by the Key Management Center (KMC) can be efficiently forwarded to participants in a cluster, while proxies and other cluster members cannot access useful data. This scheme also introduces the concept of access trees to achieve precise management of re-encryption authorization. In addition, we analyze the security and privacy protection of the solution in detail.

Moreover, homomorphic encryption keys may be compromised either intentionally or unintentionally due to non-fully trusted proxy servers [20–23]. Periodic integrity checks can verify key ownership and detect any unauthorized modifications [24]. This scheme uses a bilinear pair accumulator following a deterministic verification method to experiment with the integrity verification functionality. Unlike sampling detection, the verifier checks all the data blocks in the dataset, thus preventing any unauthorized operations from occurring.Through detailed security analysis, the program was proven to exhibit a high level of security against attacks.

In summary, this paper provides an efficient and secure solution for sharing homomorphic encryption keys among clusters in clustered federated learning. The particular contributions can be outlined as follows:

1. In this paper, a key distribution and management scheme is proposed based on proxy re-encryption to ensure the privacy of homomorphic encryption key storage and sharing in cluster federated learning. The scheme employs a fine-grained strategy and provides a framework model and a security model.

2. This paper uses a bilinear pair accumulator to implement integrity verification of homomorphic encryption keys and evaluates the effectiveness of the proposed scheme.

2. Related Work

2.1. Proxy Re-Encryption

In Eu-rocrypt98 [25], Blaze and his team proposed Proxy Re-Encryption (PRE). This scheme enables the agent to switch ciphertexts between Alice and Bob. The PRE scheme enables proxies to perform double encryption on users' ciphertext, allowing Bob to directly decrypt the re-encrypted ciphertext while preventing the proxy from accessing any valuable information. However, the original PRE scheme lacked proper control over conversion permissions, as proxies could convert encrypted files without Alice's consent.

Weng et al. put forward conditional proxy re-encryption (C-PRE) [26], where the proxy possesses the capability to solely re-encrypt the ciphertext after satisfying the criteria specified by Alice. However, this scheme consumes amounts of time and storage resources [27]. Chu et al. proposed a solution called Conditional Proxy Broadcast Re-encryption (CPBRE) to address this problem [28], which only once requires the re-encryption of the ciphertext.

Currently, proxy re-encryption techniques have undergone significant advancements [29–31]. These advancements have enabled the widespread application of proxy re-encryption in various systems, including cloud data sharing, distributed file systems, and network backups [32,33]. Despite these developments, previous approaches lacked free control over the proxy's conversion conditions. In this system, the proposed proxy broadcast re-encryption scheme supports a variable number of conditions, allows for arbitrary combinations of conditions, and facilitates partial condition matching.

In this situation, the ciphertext undergoes encryption using a collection of keywords, W, while the access tree $\mathcal{T}$ generates a re-encryption key. The agent converts transforming Alice's cryptographic text for a squad of users solely if the set of keywords W fulfills the requirements specified by the access tree $\mathcal{T}$.

2.2. Cryptographic Accumulator

The notion of an accumulator was initially introduced by Benaloh et al. [34]. If for all $x \in X$ and all $y_1,\ y_2 \in Y$, the one-way hash function $l{:}X \times Y \to X$ satisfies the quasi-exchange property:

$$l(l(x, y_1), y_2) = l(l(x, y_2), y_1). \tag{1}$$

The accumulator can accumulate all the elements in the finite set $X = \{x_1, \ldots, x_n\}$ into a compact value acc_X, while the compact value is independent of the order of x_i. Randomly select $d \in D$ as the foundation, and the accumulator is described as:

$$acc_X = l(l(\ldots l(l(l(d, x_1), x_2), x_3), \ldots), x_{k-1}), x_k). \tag{2}$$

By computing the witness wit_{x_i} for every element $x_i \in X$, one can verify $l(wit_{x_i},\ x_i) = acc_X$ and demonstrate the membership of x_i in acc_X. The witness in a dynamic accumulator can be updated by the user. The process involves calculating the witness wit_{x_i} for each element and verifying $l(wit_{x_i},\ x_i) = acc_X$ to prove $x_i's$ membership in acc_X. Apart from offering member witnesses, the universal accumulator is also capable of providing non-member witnesses for $y \notin X$.

In the current research, many scholars have proposed many accumulator schemes with different characteristics based on different number theory hypotheses. For instance, hash-based accumulators are favored for their simplicity and efficient data processing capabilities, although they may not support complex dynamic data update operations. In contrast, RSA-based accumulators leverage the difficulty of large integer factorization problems to provide robust security for data, making them particularly suitable for security-sensitive applications, but this strong security often comes at the cost of computational efficiency. On the other hand, elliptic curve accumulators optimize data representation and computation

processes, maintaining high security standards while improving processing speed, thus making them suitable for situations that demand high performance and security.

Initially, an accumulator is used to build a timestamp for recording a specific point in time for an event. With the continuous development of the accumulator scheme, its use has become more and more extensive, such as reliable certificate management, distributed signature, anonymous credentials, and digital cryptocurrency [35–40]. Subsequently, Barić enhanced the initial accumulator scheme and incorporated relevant security concepts [41]. Based on this solution, Camenisch added a dynamic add/delete value operation to build the first dynamic accumulator scheme [42]. Nguyen constructed the first dynamic accumulator scheme based on bilinear pairing, which uses the t-SDH assumption for security proof and allows for multiple values to be accumulated from a domain Z_p [43]. Based on this scheme, Damgård et al. added general-purpose features to the bilinear pair accumulator [44].

3. Preliminaries

3.1. Bilinear Mapping

Consider two cyclic groups with multiplication operations, D and D_T, having identical prime orders g. The group D is generated by the element d. We have a bilinear map $v : D \times D \to D_T$ that fulfills the prerequisites below:

1. $v\left(d_1^a, d_2^b\right) = v(d_1, d_2)^{ab}$ for all $a, b \xleftarrow{R} Z_p^*$ and $d_1, d_2 \in D$.
2. $v(d, d) \neq 1$.

3.2. The N-BDHE Presupposition

Let us denote the set $\{0, 1, \ldots, g-1\}$ as Z_p and the set $\{1, 2, \ldots, g-1\}$ as Z_p^*. We consider a prime number g. Now, we have a bilinear map $v : D \times D \to D_T$. We are given $2k+2$ elements:

$$\left(l, d, d^{\alpha}, d^{\alpha^2}, \ldots, d^{\alpha^k}, d^{\alpha^{k+2}}, \ldots, d^{\alpha^{2k}}, T\right) \in D^{2k+1} \times D_T. \tag{3}$$

Use d_i to indicate d^{α^i}. The advantage of an adversary H is as follows:

$$Adv_{D,H}^{n-BDHE} \left| \begin{array}{c} \mathsf{Pr}[H(l, d, d_1, \ldots, d_k, \ldots, d_{k+2}, d_{2k}, v(d_{k+1}, l))] = 1 \\ -\mathsf{Pr}[H(l, d, d_1, \ldots, d_k, d_{k+2}, \ldots, d_{2k}, T)] = 1 \end{array} \right|, \tag{4}$$

where $d, l \in D$, $\alpha \in Z_p^*$, and $T \in D_T$ are chosen stochastically.

3.3. The Q-SDH Presupposition

Let g be a prime number with a bit length of κ, and d be a generator of D, $\alpha \in_R Z_p^*$. For all PPT opponents H:

$$\mathsf{Pr}\left[\left(c, d^{\frac{1}{\alpha+c}}\right) \leftarrow H\left(d, d^{\alpha}, \ldots\ldots, d^{\alpha^q}\right)\right] \leq \epsilon(\kappa), \tag{5}$$

for $\exists c \in Z_p \backslash \{-\alpha\}$.

3.4. Tag Index Table

The simplified tag retrieval table is derived from the mapping version table [45]. Data owners create distinct tags for individual data blocks and store them in the verifier to facilitate dynamic data operations. The tag index table comprises two components: data block indices and corresponding tag values. The data block index is used to locate the location of the data block quickly. Tag values are used to prevent conflicts between data blocks. The verifier verifies the legitimacy of the cloud storage provider's certificate by locating the challenge data block. The data owner quickly performs data update, insert, and delete operations through the tag index block.

4. Scheme

4.1. Overview

The framework of a homomorphic encryption key sharing mechanism for clustered federated learning is shown in Figure 1. The roles in the mechanism are key management center (KMC), cloud agent, witness, and target cluster S'. Among them, the KMC generates homomorphic encryption keys for each cluster and acts as an authorizer in the proxy re-encryption process for homomorphic encryption keys, while the target cluster S' is an authorized party; the cloud proxy also acts as an aggregation server. The solid arrows in the figure indicate the data movement within the key sharing process; the dashed arrows indicate the data movement within the key integrity verification process.

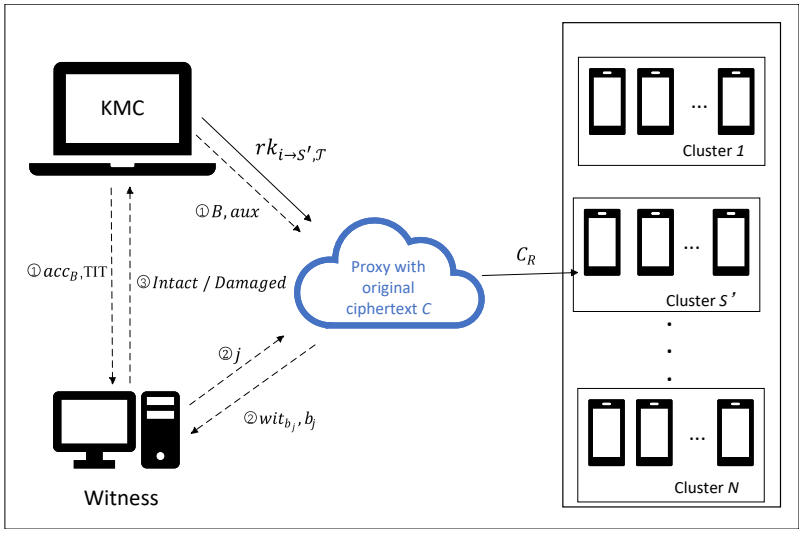

Figure 1. Overall diagram of key sharing mechanism.

KMC saves data label $B = \{b_1, b_2, b_3, \ldots\}$. When the data stored in the cloud (i.e., the ciphertext C of the homomorphic encryption key pair $< ek, sk >$ of the target cluster S') needs to be confirmed whether it has been corrupted or not, the KMC first hands over the data label B and the auxiliary threshold aux to the agent. It then uses a bilinear pair accumulator to compute the accumulated value of the data tag acc_B, generates the tag index table TIT, and passes it to the verifier Witness Next which generates a random index j and hands it to the provider. After obtaining the index j, the provider removes b_j from the tags, applies the accumulator to calculate the witness wit_{bj} for the remaining tags, and then hands over wit_{bj} and b_j to the witness. Finally, the witness performs integrity verification using the received data and transmits the verification outcome to the data owner KMC.

When the KMC intends to grant access to its cloud storage data to a third party, it must generate the transformation key $rk_{i\to S',\mathcal{T}}$ corresponding to the conditional access tree $\mathcal{T}$ and provide it to the agent. Once the agent receives the request from the KMC, it proceeds to verify that all participants individually satisfy the forwarding requirements specified in the access tree $\mathcal{T}$ and finds the cluster S' constituted by all participants that satisfy the conditions. Upon successful validation, the agent re-encrypts the data that resides on the server using the re-encryption key. Subsequently, the agent forwards the generated re-encrypted ciphertext exclusively to the participants in the target cluster S' that satisfy the conditions specified by the KMC, and then the participants in S' can decrypt the ciphertext with their individual private keys and acquire $< ek, sk >$. Meanwhile, the agent cannot obtain any KMC content in this process.

4.2. Re-Encryption Construction

This section will elaborate on the architecture of this system.

Let us define the Lagrange coefficient $\Delta_{\beta,F(x)}$ which would be used in Equation (19) to generate the components of the re-encrypted ciphertext, denoted as $\beta \in Z_p$, for a given set P consisting of elements in Z_p. The Lagrange coefficient is shown below in Equation (6):

$$\Delta_{\beta,F}(x) = \prod_{i \in P, i \neq \beta} \frac{x-i}{\beta-i}. \tag{6}$$

The scheme includes the following algorithms:

- $Setup(\lambda, k)$: Let us generate a set of instructions for constructing a bilinear map parameter (g, d, D, D_T, v) and message $\Psi = \{0,1\}^n$. Start by stochastically selecting α and ϖ from Z_p and Z from D. Assign $d_i = d^{\alpha^i}$ for $i = 1, 2, \ldots, k, k+2, \ldots, 2k$. Introduce $L_\alpha : Z_p^* \to D$ and $L_\omega : \{0,1\}^n \to Z_p^*$ as collision-resistant hash functions. Calculate $e = d^\gamma$. The output will be the public key PK and the main secret key φsk, defined as $PK = (d, d_1, \ldots, d_k, d_{k+2}, d_{2k}, e, Z, L_\alpha, L_\omega)$, $\varphi sk = \varpi$.
- $KeyGen(PK, \varphi sk, i)$: The private key of user i is $sk_i = d_i^\varpi$.
- $Encrypt(PK, S, \varphi, W)$: To securely encrypt information $\varphi \in \Psi$ for a user-set $S \subseteq \{1, 2, \ldots, k\}$ based on the prerequisite set W, the encrypt function is employed. First, a random selection of $\mu \in D_T$ and $t \in Z_p^*$ is made. The initial ciphertext consists of six components from Equation (7) to Equation (10), including $C_1, C_2, C_3, C_4, C_5, S$. These components include not only the encrypted information but also the part used to construct the re-cipher key Equation (15). The resulting ciphertext is denoted as $C = (svk, C_1, C_2, C_3, C_4, C_5, S)$.

$$C_1 = \mu \cdot v(d_1, d_k)^t, C_2 = d^t, C_3 = \left(e \cdot \prod_{j \in S} d_{k+1-j} \right)^t, \tag{7}$$

$$C_4 = \left(L_{\alpha(\beta)} \right)^t, \tag{8}$$

$$C_5 = [PRF(\mu, C_2)^{K-n} || ([PRF(\mu, C_2)]_n \oplus \varphi), \tag{9}$$

$$\mathcal{G}(\lambda) \to (svk, ssk), S = \mathcal{S}(ssk, (C_2, C_4, C_5)). \tag{10}$$

- $RKGen(PK, sk_i, S', \mathcal{T})$: The following definitions are made: q represents the polynomial, x is a non-leaf node, $\mathcal{T}$ is the tree, r is the root node, and $q_r(0)$ expresses the degree of the root of the tree. Given the inputs $sk_i = d_i^\varpi$, $S' \in \{1, 2, \ldots, k\}$, and $\mathcal{T}$, we proceed with the following steps. Firstly, we stochastically select $\mu \in \{0,1\}^n$ and a q_x for each x in the $\mathcal{T}$. The process begins at the r and $RKG(\mathcal{T}, L_\omega(\mu))$ is used to opt for the polynomials in a top-down manner. $RKG(\mathcal{T}, L_\omega(\mu))$ is described as follows: for each node x, the q_x is set with a degree of $f_x = n_x - 1$. The $q_r(0)$ is set to $L_\omega(\mu)$. For other x, we set $q_x(0)$ to $q_g(x)$, and then stochastically opt for the remaining coefficients to completely define the polynomial q_x. We set $\beta = keyword(x)$ for each x. Now, let us calculate the re-encryption key $rk_{i \to S', \mathcal{T}'} = (\mathcal{T}, A_x, B_x, rk_1, rk_2, rk_3, rk_4, S')$ in Equation (15) for the agent:

  Choose a random value $r_x \xleftarrow{R} Z_p^*$ and calculate it: $A_x = sk_i \cdot Z^{q_x(0)} \cdot L_\alpha(\beta)^{r_x}; B_x = d^{r_x}$. Selects stochastic value $t' \in Z_p^*, r' \in D_T,\ R \in \{0,1\}^n$ and sets:

$$rk_1 = r' \cdot v(d_1, d_n)^{t'}, rk_2 = d^{t'}, \tag{11}$$

$$rk_3 = \left(e \cdot \prod_{j \in S'} d_{k+1-j} \right)^{t'}, \tag{12}$$

$$rk_4 = [PRF(\mu', rk_2)^{K-n} || ([PRF(\mu', rk_2)]_n \oplus R), \tag{13}$$

$$\mathcal{G}(\lambda) \to (svk', ssk'), S' = \mathcal{S}(ssk', (rk_2, rk_4))^{t'}. \tag{14}$$

Output the re-cipher key which is used to encrypt the ciphtext C into the cryptographic text C_R Equation (21) that can be decrypted by others' private keys in the group:

$$rk_{i\to S',\mathcal{T}'} = \left(\mathcal{T}, A_x, d^{r_x}, r' \cdot v(d_1, d_n)^{t'}, d^{t'}, \left(e \cdot \prod_{j\in S'} d_{k+1-j} \right)^{t'}, rk_4, S' \right). \tag{15}$$

- $ReEnc(PK, rk_{i\to S',\mathcal{T}'}, i, S, S', C)$: Enter a $rk_{i\to,S',W'}$ and a C. Verify if the equations below hold:

$$v\left(C_2, e \cdot \prod_{j\in S} d_{k+1-j} \right) \stackrel{?}{=} v(d, C_3), \tag{16}$$

$$\mathcal{V}(svk, S, (C_2, C_4, C_5)) \stackrel{?}{=} 1, \tag{17}$$

$$v(C_2, L_\alpha(\mathcal{T})) \stackrel{?}{=} v(d, C_4). \tag{18}$$

Equations (16)–(18) are used to verify the integrity of ciphertext C.
In the event that any of the aforementioned equations fail to hold, the output will be $\perp$. Conversely, a recursive algorithm named $NodeReEnc(C, rk_{i\to S',\mathcal{T}'}, x)$ is introduced to process the initial C, the $rk_{i\to S',\mathcal{T}'}$, and the note x within the tree.

1. When it comes to leaf x, if $\beta \in W$, let $\beta = keyword(X)$, then
$NodeReEnc\big(C, rk_{i\to S',\mathcal{T}'}, x\big) = \frac{v(C_2, A_x)}{v(B_x, C_4)} = \frac{v\left(d^t, sk_i \cdot Z^{q_x(0)} \cdot L_\alpha(\beta)^{r_x}\right)}{v\left(d^{r_\beta}, L_\alpha(\beta)^t\right)}$
$= v\big(sk_i, d^t\big) \cdot v(Z, d)^{t \cdot q_x(0)}$. Otherwise, output $\perp$.
2. In the case where x represents a non-leaf node, the recursive procedure $NodeReEnc(C, rk_{i\to j,\mathcal{T}'}, z)$ is called by all descendent nodes z of ancestor nodes x, and the resulting outcome is stored as $\mathcal{T}_z$. Let F_x denote a random selection of children nodes z with a size of k_x, ensuring that $\mathcal{T}_z \neq \perp$. If the condition is not satisfied, $NodeReEnc$ returns the value $\perp$. However, if a satisfactory set $F'_x = \{index(z) : z \in S\}$ can be formed, then the following computation is carried out using the Lagrange coefficient generated in Equation (6) and the result of $NodeReEnc(C, rk_{i\to j,\mathcal{T}'}, z)$:

$$\begin{aligned} T_x &= \prod_{z\in F_x, i=index(z)} (T_z)^{\Delta_{i,F'_x(0)}} \\ &= \prod_{z\in F_x, i=index(z)} \left(v\big(sk_i, d^t\big) \cdot v(Z, d)^{t\cdot q_x(0)} \right)^{\Delta_{i,F'_x}(0)} \\ &= v\big(sk_i, d^t\big) \cdot \prod_{z\in F_x, i=index(z)} \left(v(Z, d)^{t\cdot q_{g(z)} index(z)} \right)^{\Delta_{i,F'_x}(0)} \\ &= v\big(sk_i, d^t\big) \cdot \prod_{z\in F_x, i=index(z)} \left(v(Z, d)^{t\cdot q_x(i)} \right)^{\Delta_{i,F'_x}(0)} \\ &= v\big(sk_i, d^t\big) \cdot v(Z, d)^{sq_x(0)}. \end{aligned} \tag{19}$$

In the end, using Equation (19) to calculate $\widetilde{C_1}$:

$$\widetilde{C_1} = C_1 \cdot \frac{v\left(T_r \cdot \prod_{j\in S, j\neq i} d_{k+1-j+i}, C_2 \right)}{v(d_i, C_3)}. \tag{20}$$

The ciphertext C is re-encrypted into re-cipher C_R:

$$C_R = \left(svk, \widetilde{C_1}, C_2, C_4, C_5, S, r' \cdot v(d_1, d_n)^{t'}, d^{t'}, \left(e \cdot \prod_{j \in S'} d_{k+1-j} \right)^{t'}, rk_4, S' \right). \quad (21)$$

The original ciphertext C can be decrypted by user i's private sk_i and the re-cipher C_R can be decrypted by user j's private key sk_j. If the result of $DecryptO$ and $DecryptR$ is equal, then the re-encryption succeed.

- $DecryptO(PK, sk_i, i, S, C)$: Enter a sk_i and a $C = (C_1, C_2, C_3, C_4, C_5, S)$ with the following program:
  1. Verifies the validity of Equation (16) to Equation (18). If any of these equations fail to hold, then the output will be $\perp$, indicating the termination of the process.
  2. Calculates $\mu = C_1 \cdot v\left(sk_i, \prod_{j \in S, j \neq i} d_{k+1-j+i}, C_2\right) / v(d_i, C_3)$. If $PRF[\mu, C_2]^{K-n} = [C_5]^{K-n}$ hold, returns $\varphi = PRF[\mu, C_2]_n \oplus [C_5]_n$.
- $DecryptR(PK, sk_j, i, j, S, S', C_R)$: Enter a sk_j and a C_R with the following program:
  1. Checks the equations:

$$v\left(rk_2, g \cdot \prod_{j \in s} d_{k+1-j}\right) \stackrel{?}{=} v(d, rk_3) \quad (22)$$

$$\mathcal{V}\left(svk', S', (rk_2, rk_4)\right) \stackrel{?}{=} 1 \quad (23)$$

  Equations (22) and (23) are used to verify the integrity of re-cipher C_R. Success goes to the next step while failure returns $\perp$.
  2. Calculate $\mu' = rk_1 \cdot v\left(sk_j \cdot \prod_{a \in S', a \neq j} d_{k+1-a+j}, rk_2\right) / v\left(d_j, rk_3\right)$, if $PRF[\mu', rk_2]^{K-n} = [rk_4]^{K-n}$, output $R = PRF[\mu', rk_2]_n \oplus [rk_4]_n$. Success goes to the next step while failure returns $\perp$.
  3. Calculate $\mu = \widetilde{C_1} / v\left(C_2, Z^{L_\omega(R)}\right)$. If $PRF[\mu, C_2]^{K-n} = [V]^{K-n}$, output $\varphi = PRF[\mu, C_2]_n \oplus [C_5]_n$. Otherwise, returns $\perp$ and call off.

Consistency: For any set of common parameters pair, any message M in plaintext space, any user public-private key pair (pk_i, sk_i), (pk_j, sk_j), the following equation holds:

$$\mathrm{DecryptO}(\mathrm{par}, \mathrm{sk_i}, \mathrm{Enc}(\mathrm{par}, \mathrm{M}, \mathrm{pk_i})) = \mathrm{M} \quad (24)$$

$$\mathrm{DecryptR}\Big(\mathrm{par}, \mathrm{sk_j}, \mathrm{ReEnc}\Big(\mathrm{par}, \mathrm{RKGen}\Big(\mathrm{par}, \mathrm{sk_i}, \mathrm{pk_j}\Big), C_R\Big)\Big) = \mathrm{M} \quad (25)$$

The integrity of the ciphertext has been verified in the DecryptO and DecryptR, so here only consistency checks need to be performed on the C_1 and $\widetilde{C_1}$ of the ciphertext C and the re-encrypted ciphertext C_R Equation (21).

1. If $C = (svk, C_1, C_2, C_3, C_4, C_5, S)$ represents the initial ciphertext, then the following conditions apply:

$$\begin{aligned} & C_1 \cdot \frac{v\left(sk_i \cdot \prod_{j \in S, j \neq i} g_{k+1-j+i}, C_2\right)}{v(d_i, C_3)} \\ &= \mu \cdot v(d_1, d_n)^t \cdot \frac{v\left(d_i^{\omega} \cdot \prod_{j \in S, j \neq i} d_{k+1-j+i}, d^t\right)}{e\left(d_i, d^{\omega} \cdot \prod_{j \in S} d_{k+1-j}\right)^t} \\ &= \mu \cdot v(d_1, d_n)^t \cdot \frac{e\left(d^t, \prod_{j \in S, j \neq i} g_{k+1-j+i}\right)}{e\left(d^t, \prod_{j \in S} g_{k+1-j+i}\right)} \\ &= \mu \cdot \frac{v(d_1, d_n)^t}{v(d^t, d_{n+1})} \\ &= \mu \end{aligned} \tag{26}$$

2. If $C_R = (svk, \widetilde{C_1}, C_2, C_4, S, svk', rk_1, rk_2, rk_3, rk_4, rk_5, S')$ represents the re-encrypted ciphertext, then the following conditions apply:

$$\begin{aligned} \widetilde{C_1} &= C_1 \cdot \mathcal{T}_r \cdot \frac{v\left(\prod_{j \in S, j \neq i} g_{k+1-j+i}, C_2\right)}{v(d_i, C_3)} \\ &= \mu \cdot v(d_1, d_k)^t \cdot v\left(sk_i, d^t\right) \cdot v(Z, d)^{sq_x(0)} \cdot \frac{v\left(\prod_{j \in S, j \neq i} d_{k+1-j+i}, d^t\right)}{v\left(d_i, e \cdot \prod_{j \in S} g_{k+1-j}\right)^t} \\ &= \mu \cdot v\left(d^t, Z\right)^{L_{\omega}(R)} \end{aligned} \tag{27}$$

It is eventually feasible of calculating:

$$\frac{\widetilde{C_1}}{v\left(C_2, Z^{L_{\omega}(R)}\right)} = \mu \tag{28}$$

From the results of Equations (26)–(28), it can be seen that the decryption results of the corresponding parts of the ciphertext and re-encrypted ciphertext during the decryption process are the same, thus confirming consistency.

4.3. Integrity Verification Construction

Setup: Input a security parameter λ and original ciphertext C, whereby the raw data owner proceeds as follows:

1. Construct bilinear map tuple $t = (g, D_1, D_2, D_T, v, d_1, d_2)$ and $t' = \left(d_2, d_2^s, \ldots\ldots, d_2^{s^k}\right)$. Randomly select $s \xleftarrow{R} Z_p^*$.
2. Divide the data C into n copies, which is $C = \{c_1, c_2, c_3, \ldots\}$. Then, each data block corresponds to a label τ_i, where i is the segment index. Store each label in TIT afterward.
3. Add tags to the corresponding ciphertext segment c_i, generate a data block $B = \{b_1, b_2, b_3, \ldots\}$, and calculate the accumulated value of the data block $acc_B = d_1^{\prod_{i=1}^{k}(b_i+s)}$.
4. Generate $aux = \left(d_2, d_2^s, \ldots\ldots, d_2^{s^k}\right)$ and outsource them to the provider.
5. The provider calculates $f(s) = \prod_{b \in B \setminus b_i}(b + s)$, where s represents the unknown number and a_i represents the coefficient of s. Finally, obtain $aux = \left\{d_2, d_2^s, \ldots\ldots, d_2^{s^k}\right\}$.

6. The witness calculates the challenge block: $wit_{b_j} = d_2^{a_0} * (d_2^s)^{a_1} * \ldots * \left(d_2^{s^{k-1}}\right)^{a_{k-1}} = \prod_{i=0}^{k-1} \left(d_2^{s^i}\right)^{a^i}$.
7. The cloud storage provider sends $\left(wit_{b_j}, b_j\right)$ to the trusted witness.

Verify: The algorithms included in the verification process are as follows:

1. The witness first checks $v(acc_B, d_2) \stackrel{?}{=} v\left(d_1^{b_j} d_1^s, wit_{b_j}\right)$. In case the preceding equation fails to satisfy, an error symbol $\perp$ should be outputted. If the equation holds, then proceed with the subsequent instructions.
2. The witness retrieves the data segment c_i and its associated indicator τ_i from the challenged block. Then, using K_L and c_j, the witness calculates $\tau_j' = L(c_i||k_H)$.
3. Verify $\tau_j' \stackrel{?}{=} \tau_j$.

5. Proof of Security

5.1. Ind-Cca Security

Theorem 1. *Assuming the Decisional n-BDHE assumption holds, and considering L_α and L_ω as collision-resistant hash functions, the key sharing mechanism achieves IND-CCA security in the absence of random oracles.*

Lemma 1. *Suppose there exists an opponent H capable of breaking the security of the key sharing mechanism under the IND-O-CCA notion. In such a case, one might consider constructing a simulator $\mathcal{B}$ that is capable of solving the Decisional n-BDHE assumption.*

Proof. When presented with a Decisional n-BDHE instance $(l, d, d_1, \ldots d_k, d_{k+2}, \ldots, d_{2k}, T)$, $\mathcal{B}$ determines if T equals $v(d_{k+1}, l)$ or if T is an arbitrary element chosen randomly from D_T.

$\mathcal{B}$ starts with an empty table as its initial state:

Key^{List}: It keeps track of the tuples (ω, i, sk_i) which contain the details of the private keys.

$ReKey^{List}$: Saves the data produced by $RKGen(sk_i, S', W')$ in the $(\omega_1, i, S', W', rk_{i\to S', \mathcal{T}}, \mu, R, flag_1)$ tuple where the information is stored. $flag_1 = 1$ signifies the legitimacy of the re-encryption key, while $flag_1 = 0$ signifies that the re-cipher key is a randomly generated value. □

Initialize. The challenger H chooses a set of users S^* from $\{1, 2, \ldots, k\}$ and a set of conditions $W^* = \beta_1^*, \beta_2^*, \ldots, \beta_k^*$.

Setup. The simulator $\mathcal{B}$ selects a random non-zero value σ from Z_p^* and an element Z from D. It then defines the following to create users' private keys:

$$e = d^\sigma \cdot \left(\prod_{j \in S^*} d_{k+1-j}\right)^{-1} \stackrel{\Delta}{=} d^\varpi \tag{29}$$

H is provided with the public key $PK = (d, d_1, \ldots, d_k, d_{k+2}, \ldots, d_{2k}, e, Z, L_\alpha, L_\omega)$ and the secret key $sk = \varpi$, which are chosen by $\mathcal{B}$.

Query Phase I. $\mathcal{B}$ provides answers to the inquiries posed by H as follows:

- $Extract(i)$: After verifying that i is not an element of S^*, $\mathcal{B}$ proceeds to check Key^{List}. If the tuple (ω, i, sk_i) is present in Key^{List}, $\mathcal{B}$ will provide H with the corresponding sk_i. However, if the tuple does not exist, then $\mathcal{B}$ will generate a biased coin ω with a probability of $\Pr[\omega = 1]$ equal to δ.
  - If $\omega = 0$, then $\mathcal{B}$ termination.
  - If $\omega = 1$, then $\mathcal{B}$ calculates the following equation to obtain the private key sk_i of user i in S^*:

$$\begin{aligned} sk_i &= d_i^{\sigma} \cdot \left(\prod_{j \in S^*} d_{k+1-j} \right)^{-1} \\ &= \left(d^{\sigma} \cdot \left(\prod_{j \in S^*} d_{k+1-j} \right)^{-1} \right)^{\alpha^i} \\ &= e^{\alpha^i} \\ &= d_i^{\omega} \end{aligned} \tag{30}$$

- $RKGen(i, S', \mathcal{T})$: Set $i \in S^*$, $j \in S^*$, and $\mathcal{T}(W^*) = 1$. $\mathcal{B}$ ensures that there are no tuples in Key^{List} of the form $(*, j, sk_j)$, where $*$ is a placeholder. If such a tuple is found, $\mathcal{B}$ terminates the process. However, if there exists a tuple $(*, i, S', \mathcal{T}, rk_{i \to S', \mathcal{T}}, \mu, R, *)$ in $ReKey^{List}$, $\mathcal{B}$ returns the value of $rk_{i \to S', \mathcal{T}}$. If neither of these conditions is met, then $\mathcal{B}$ proceeds with the following steps:
  Suppose there is a tuple $(1, i, sk_i)$ present in Key^{List}. In that case, $\mathcal{B}$ employs sk_i to create the re-cipher key $rk_{i \to S', \mathcal{T}}$ using the $RKGen$ algorithm, following the same procedure as in the actual scheme. $\mathcal{B}$ then provides the re-cipher key to H, includes $(*, i, S', \mathcal{T}, rk_{i \to S', \mathcal{T}}, \mu, R, 1)$ in $ReKey^{List}$, and randomly selects r' and R during the $RKGen$ algorithm.
  Alternatively, $\mathcal{B}$ employs a biased coin $\mathcal{B}$ to make a decision. If ω equals 1, $\mathcal{B}$ interacts with $Extract(i)$ to obtain sk_i. Subsequently, $\mathcal{B}$ generates the re-encryption key $rk_{i \to S', \mathcal{T}}$ using the $RKGen$ algorithm, returns it to H, and adds $(1, i, sk_i)$ and $(*, i, S', \mathcal{T}, rk_{i \to S', \mathcal{T}}, \mu, R, 1)$ to Key^{List} and $ReKey^{List}$, respectively. In the case where ω equals 0, $\mathcal{B}$ sets $\left(A_\beta = \rho_\beta\right), \left(B_\beta = \rho'_\beta\right); \beta \in keyword(\beta)$ for randomly selected ρ_β and ρ'_β from D. Next, $\mathcal{B}$ constructs rk_1, rk_2, rk_3, rk_4 and selects μ' and R. Finally, $\mathcal{B}$ forwards the re-cipher key to H and then appends $(*, i, S', \mathcal{T}, rk_{i \to S', \mathcal{T}}, \mu, R, 0)$ to $ReKey^{List}$.
- $ReEnc(i, S, S', C)$: $\mathcal{B}$ executes the subsequent procedures:
  In the presence of $(*, i, S', \mathcal{T}, rk_{i \to S', \mathcal{T}}, \mu, R, *)$ in $ReKey^{List}$, $\mathcal{B}$ encrypts (PKm, S, φ, W) as C using the encrypt function. If $\mathcal{T}(W)$ equals 1, then $\mathcal{B}$ employs the re-encryption key $rk_{i \to S', \mathcal{T}}$ to generate C_R through the $ReEnc$. Following this, $\mathcal{B}$ appends $(i, S, S', C, C_R, *)$ to $ReEnc^{List}$ and returns C_R to H.
  If $(*, i, S', \mathcal{T}, rk_{i \to S', \mathcal{T}}, \mu, R, *)$ is not found in $ReKey^{List}$, $\mathcal{B}$ initiates an $RKGen(i, S')$ query to acquire the re-encryption key $rk_{i \to S', \mathcal{T}}$. Subsequently, $\mathcal{B}$ generates C_R and includes $(i, S, S', C, C_R, *)$ in the $ReEnc^{List}$.
- $Decrypt_O(i, S, C)$: $\mathcal{B}$ performs a validation check to confirm the fulfillment of Equations (16)∼(18). If these equations are unsatisfied, then $\mathcal{B}$ outputs $\perp$. Otherwise, B proceeds with the following steps:
  If there is an entry $(1, i, sk_i)$ in Key^{List}, then $\mathcal{B}$ utilizes sk_i to retrieve φ.
  In the absence of $(1, i, sk_i)$ in Key^{List}, B initiates an $Extract(i)$ query to acquire sk_i and applies sk_i to restore φ.
- $Decrypt_R(i, j, S, S', C_R)$: $\mathcal{B}$ validates the validity of Equations (22) and (23). If the aforementioned formulas are invalid, then $\mathcal{B}$ outputs $\perp$ and terminates. If they hold, then $\mathcal{B}$ continues with the following steps:
  If there is an entry $\left(1, j, sk_j\right)$ in Key^{List}, then $\mathcal{B}$ utilizes sk_j to retrieve φ.
  In the absence of $(1, j, sk_i)$ in Key^{List}, B initiates an $Extract(j)$ query to acquire sk_j and applies sk_j to restore φ.

Challenge. Upon completion of Query Phase I as determined by H, it produces two messages φ_0 and φ_1 of the same length. $\mathcal{B}$ randomly selects a value b from $\{0, 1\}$ and r^* from G_T. Let l be equal to d^{t^*}, where t^* is randomly chosen. $\mathcal{B}$ performs the following computation:

$$C_1^* = \mu^* \cdot T \tag{31}$$

$$C_2^* = l = d^{t^*} \tag{32}$$

$$\begin{aligned} C_3^* &= l^{\sigma} = d^{\sigma t^*} \\ &= \left(d^{\sigma} \cdot \left(\prod_{j \in S^*} d_{k+1-j} \right)^{-1} \left(\prod_{j \in S^*} d_{k+1-j} \right) \right)^{t^*} \\ &= \left(e \cdot \prod_{j \in S^*} g_{k+1-j} \right)^{t^*} \end{aligned} \tag{33}$$

$$C_4^* = L_{\alpha}(\beta)^{t^*}, \quad \beta \in W^* \tag{34}$$

$$C_5^* = [PRF(\mu^*, C_2^*)]^{K-k} || ([PRF(\mu^*, \ C_2^*)]_k \oplus \varphi_b \ ) \tag{35}$$

$$\mathcal{G}(\lambda) = (ssk^*, svk^*) \tag{36}$$

$$S^* = \mathcal{S}(svk^*, (C_2^*, C_4^*, C_5^*)) \tag{37}$$

If $T = v(d_{k+1}, l)$, then $C_1^* = \mu^* \cdot T = \mu^* \cdot v(d, d_{k+1})^{t^*}$. Let us denote C_3^* as a legitimacy challenge cryptographic text. If T represents a stochastic element in D_T, then the adversary's observation of C_3^* is unrelated to the value of b.

Query Phase II. While adhering to the constraints specified in the IND-O-CCA game, H proceeds to make further queries, following the identical pattern as in Query Phase I.

Guess. H provides the guess b', and if b' matches b, then the output is 1, indicating that $T = v(d_{k+1}, l)$. Otherwise, the output is 0, indicating that T is a random value selected from D_T.

5.2. Attack Prevention

This section will analyze sequentially the types of attacks that may be affected by the integrity verification section. It will also demonstrate the resilience of this program to these attacks.

Tag counterfeiting attack. The property of collision resistance is essential for a hash function. Due to the utilization of a hash function $L\left(c_j'\right)$ in generating tags τ_j', the likelihood of generating the same tag with different data is extremely low. Consequently, cloud service providers are unable to deceive witnesses by creating counterfeit tags.

Data deletion attack. When confronted with the witness challenge, the provider is unable to compute the witness wit_{b_j} by aggregating data blocks and tags $\prod_{i=0}^{k-1}\left(d_2^{s^i}\right)^{a^i}$ if the original data is misplaced or erased. Therefore, cloud service providers cannot use tags to generate legitimate proof of ownership in the event of the loss of raw data $\left(wit_{b_j}, b_j\right)$.

Substitution attack. In the event that the witness challenges the provider using a randomly selected block index, if the provider substitutes the corrupted or deleted data with an incompatible data block or label, then the token τ_j' calculated by the witness becomes unverifiable. Consequently, the provider is unable to employ alternative tactics to deceive the witness.

Replay attack. The provider's utilization of previously cached data to respond to the new challenge posed by the current authenticator holds no significance. Firstly, during the verification and challenge process, the likelihood of the verifier executing the challenge using the same random index j is negligible. This is due to the fact that the provider can solely compute the witness wit_{b_j} through the auxiliary value aux_2, whereas the auxiliary value aux_2 transmitted by the original data owner does not include $sk_{acc} = s$. Second, the witness generated by the challenge before caching needs to store the corresponding data block, which requires more storage space for the cloud service provider. In summary, integrity verification is unaffected by replay attacks.

Data leakage attack. Since the data owner has encrypted the data stored on the server, no third party can know the actual content of the outsourced data during integrity

verification and re-encryption. Therefore, even if leaking encrypted data, the system can still guarantee its security.

6. Performance

6.1. Performance of PBRE

For the proxy broadcast re-encryption subsystem, this section evaluates the time expenditure for each phase in the scheme with the prior scenario. The Golang-based PBC software package (version 0.5.14) implements the ciphertext conversion module of the mobile multimedia sharing system. The PBC software package not only comprises a cryptographic library based on bilinear pairings but also provides a framework for building cryptographic systems. The test was conducted on a system comprising an Intel Xeon X5365 @3.00 GHz processor, Centos 7.5 operating system, and Go 1.19 programming language. For the test, a 160-bit elliptic curve $Y^2 = X^3 + X$ was chosen. To minimize errors, the program was executed 10 times, and the average value was recorded as the test result. The test results are shown in Table 1.

Table 1. Ciphertext conversion performance comparison.

Scheme	RKGen (ms)	Encrypt (ms)	ReEncrypt (ms)	DecryptO (ms)	DecryptR (ms)
Scheme [32]	26.87	14.96	169.6	19.45	20.37
This system	27.93	19.58	53.06	21.88	25.97

Based on the test findings, it was observed that the time required for ciphertext conversion in the experimental scheme is significantly lower compared to the control scheme. This is because the experimental scheme only requires a single re-encryption operation to generate a collection of public ciphertexts. However, the remaining steps of the system take more time than the control scheme. This is attributed to the fact that the key length generated by the system is directly proportional to the magnitude of the user group, and the amount of data processed in each operation is substantially larger. Nevertheless, the overall efficiency of the subsystem is deemed satisfactory.

6.2. Performance of Integrity Verification

For the integrity verification subsystem, the experiment uses the DCLXVI library to calculate the elliptic curve, and SHA-3 to generate the 160-bit label. The software and hardware environment used for the test is the same as the environment of the ciphertext conversion subsystem. In the subsystem establishment phase, the original ciphertext C needs to be divided into multiple data segments. Through previous experiments, it was empirically determined that a data block size of 768 bytes yields optimal results. For the current testing phase, a dataset of 1 GB was used. The obtained test results are presented in Table 2.

Table 2. Integrity verification performance comparison.

Scheme	Setup (s)	Challenge (μs)	Proof (s)	Verify (Bytes)	Storage (MB)
Scheme [46]	9480	3.3	68.4	42.3	33.1
This system	18.9	0.64	53.06	0.0033	28.9

It can be seen from the comparison between this scheme and the control scheme that our scheme exhibits certain benefits across various performance aspects. The most crucial point is that the time complexity of performing the verification operation is a fixed value of $O(2)$, which does not change as the block size increases. Moreover, the time spent in the system establishment phase is much smaller than that of the control scheme.

This is because the input field of the RSA accumulator used in the comparison scheme is limited to prime numbers, and each data block needs to be bitwise shifted to avoid a collision. The bilinear pair accumulator used in this scheme does not need to preprocess the data during the initialization phase. Moreover, the scheme possesses the advantages of time cost and space cost in terms of storage overhead, challenges, and proof. In summary, the solution used by the integrity verification subsystem is suitable for computing-capable devices or frameworks, so it has a broader range of applications, such as multimedia mobile devices, edge computing, etc.

7. Conclusions

In this paper, we propose a conditional agent-based re-encryption key sharing mechanism for clustered federated learning. The scheme combines a proxy broadcast re-encryption mechanism and an integrity verification mechanism to protect the keys used for homomorphic encryption across clusters stored in the cloud. The proxy broadcast re-encryption subsystem can convert the keys generated and encrypted by the KMC into a new set of ciphertexts to be provided to the user without the server having the capability to access the valuable data. Thus, the KMC can control the access conditions through an access tree. The integrity verification subsystem ensures that the keys are not deleted or corrupted while reducing the computational and storage costs of the verification process. Experiments show that the proposed scheme has significant improvements in overall computing efficiency and communication cost, especially in terms of storage overhead, challenges and proofs, and has greater advantages in time cost and space cost. Therefore, the scheme can be applied well in environments with limited computing power.

Author Contributions: Y.Z., Z.Z. and S.W. were responsible for conceptual analysis, methodological analysis, and writing of the original draft. S.J., Z.Z. and S.H. were responsible for thesis revision and review. Y.Z. was responsible for review, supervision, and project administration. 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 (No. 62072249), the Natural Science Foundation of Zhejiang Province (No. LHY22E080004), the Open Fund of the Key Laboratory of Port, Waterway, and Sedimentation Engineering, Ministry of Communications, China (No. YK222001-7), the National Key Research and Development Program of Guangdong Province (No. 2020B0101090002), and the Natural Science Foundation of Jiangsu Province (No. BK20200418, BE2020106).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Yin, C.; Xi, J.; Sun, R.; Wang, J. Location privacy protection based on differential privacy strategy for big data in industrial internet of things. *IEEE Trans. Ind. Inform.* **2017**, *14*, 3628–3636. [CrossRef]
2. Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A.K.; Kim, H.J. An intelligent data gathering schema with data fusion supported for mobile sink in wireless sensor networks. *Int. J. Distrib. Sens. Netw.* **2019**, *15*, 1550147719839581. [CrossRef]
3. Ge, C.; Liu, Z.; Susilo, W.; Fang, L.; Wang, H. Attribute-based encryption with reliable outsourced decryption in cloud computing using smart contract. *IEEE Trans. Dependable Secur. Comput.* 2023, *early access*.
4. Liu, J.; Liang, T.; Sun, R.; Du, X.; Guizani, M. A privacy-preserving medical data sharing scheme based on consortium blockchain. In Proceedings of the GLOBECOM 2020–2020 IEEE Global Communications Conference, IEEE, Taipei, Taiwan, 7–11 December 2020; pp. 1–6.
5. Konečnỳ, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv* **2016**, arXiv:1610.05492.
6. Maurya, C.; Chaurasiya, V.K. Collusion-resistant and privacy-preserving data sharing scheme on outsourced data in e-healthcare system. *Multimed. Tools Appl.* **2023**, 82, 40443–40472. [CrossRef]
7. Yin, Y.; Xu, W.; Xu, Y.; Li, H.; Yu, L. Collaborative QoS prediction for mobile service with data filtering and SlopeOne model. *Mob. Inf. Syst.* **2017**, *2017*, 7356213. [CrossRef]

8. Li, Q.; Wen, Z.; Wu, Z.; Hu, S.; Wang, N.; Li, Y.; Liu, X.; He, B. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 3347–3366. [CrossRef]
9. Ge, C.; Susilo, W.; Liu, Z.; Baek, J.; Luo, X.; Fang, L. Attribute-based proxy re-encryption with direct revocation mechanism for data sharing in clouds. *IEEE Trans. Dependable Secur. Comput.* 2023, *early access*.
10. Zheng, T.; Luo, Y.; Zhou, T.; Cai, Z. Towards differential access control and privacy-preserving for secure media data sharing in the cloud. *Comput. Secur.* **2022**, *113*, 102553. [CrossRef]
11. Yeh, L.Y.; Shen, N.X.; Hwang, R.H. Blockchain-based privacy-preserving and sustainable data query service over 5g-vanets. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 15909–15921. [CrossRef]
12. Ren, Y.; Leng, Y.; Cheng, Y.; Wang, J. Secure data storage based on blockchain and coding in edge computing. *Math. Biosci. Eng.* **2019**, *16*, 1874–1892. [CrossRef]
13. Maiti, S.; Misra, S. P2B: Privacy preserving identity-based broadcast proxy re-encryption. *IEEE Trans. Veh. Technol.* **2020**, *69*, 5610–5617. [CrossRef]
14. Pu, Y.; Hu, C.; Deng, S.; Alrawais, A. R^2PEDS: A recoverable and revocable privacy-preserving edge data sharing scheme. *IEEE Int. Things J.* **2020**, *7*, 8077–8089. [CrossRef]
15. Ge, C.; Susilo, W.; Baek, J.; Liu, Z.; Xia, J.; Fang, L. Revocable attribute-based encryption with data integrity in clouds. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2864–2872. [CrossRef]
16. Zhu, H.; Xu, J.; Liu, S.; Jin, Y. Federated learning on non-IID data: A survey. *Neurocomputing* **2021**, *465*, 371–390. [CrossRef]
17. Ghosh, A.; Hong, J.; Yin, D.; Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv* **2019**, arXiv:1906.06629.
18. Ghosh, A.; Chung, J.; Yin, D.; Ramchandran, K. An efficient framework for clustered federated learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 19586–19597. [CrossRef]
19. Duan, M.; Liu, D.; Ji, X.; Wu, Y.; Liang, L.; Chen, X.; Tan, Y.; Ren, A. Flexible clustered federated learning for client-level data distribution shift. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 2661–2674. [CrossRef]
20. Ren, Y.; Leng, Y.; Qi, J.; Sharma, P.K.; Wang, J.; Almakhadmeh, Z.; Tolba, A. Multiple cloud storage mechanism based on blockchain in smart homes. *Future Gener. Comput. Syst.* **2021**, *115*, 304–313. [CrossRef]
21. Sun, J.; Xu, G.; Zhang, T.; Yang, X.; Alazab, M.; Deng, R.H. Verifiable, fair and privacy-preserving broadcast authorization for flexible data sharing in clouds. *IEEE Trans. Inf. Forensics Secur.* **2022**, *18*, 683–698. [CrossRef]
22. Zhang, X. Bilinear mapping and blockchain-based privacy-preserving and data sharing scheme for smart grid. *Int. J. Netw. Secur.* **2023**, *25*, 151–160.
23. Ge, C.; Susilo, W.; Baek, J.; Liu, Z.; Xia, J.; Fang, L. A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2907–2919. [CrossRef]
24. Ren, Y.; Huang, D.; Wang, W.; Yu, X. BSMD: A blockchain-based secure storage mechanism for big spatio-temporal data. *Future Gener. Comput. Syst.* **2023**, *138*, 328–338. [CrossRef]
25. Blaze, M.; Bleumer, G.; Strauss, M. Divertible protocols and atomic proxy cryptography. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Espoo, Finland, 31 May–4 June 1998; pp. 127–144.
26. Weng, J.; Deng, R.H.; Ding, X.; Chu, C.K.; Lai, J. Conditional proxy re-encryption secure against chosen-ciphertext attack. In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, Australia, 10–12 March 2009; pp. 322–332.
27. Fang, G.; Sun, Y.; Almutiq, M.; Zhou, W.; Zhao, Y.; Ren, Y. Distributed Medical Data Storage Mechanism Based on Proof of Retrievability and Vector Commitment for Metaverse Services. *IEEE J. Biomed. Health Inform.* 2023, *early access*.
28. Chu, C.K.; Weng, J.; Chow, S.S.; Zhou, J.; Deng, R.H. Conditional proxy broadcast re-encryption. In Proceedings of the Information Security and Privacy: 14th Australasian Conference, ACISP 2009, Proceedings 14, Brisbane, Australia, 1–3 July 2009; pp. 327–342.
29. Liu, Y.; Ren, Y.; Ge, C.; Xia, J.; Wang, Q. A CCA-secure multi-conditional proxy broadcast re-encryption scheme for cloud storage system. *J. Inf. Secur. Appl.* **2019**, *47*, 125–131. [CrossRef]
30. Ren, Y.; Qi, J.; Liu, Y.; Wang, J.; Kim, G.J. Integrity verification mechanism of sensor data based on bilinear map accumulator. *ACM Trans. Internet Technol. (TOIT)* **2021**, *21*, 1–19. [CrossRef]
31. Ge, C.; Liu, Z.; Xia, J.; Fang, L. Revocable identity-based broadcast proxy re-encryption for data sharing in clouds. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 1214–1226. [CrossRef]
32. Weng, J.; Chen, M.; Yang, Y.; Deng, R.; Chen, K.; Bao, F. CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. *Sci. China Inf. Sci.* **2010**, *53*, 593–606. [CrossRef]
33. Borcea, C.; Polyakov, Y.; Rohloff, K.; Ryan, G. PICADOR: End-to-end encrypted Publish–Subscribe information distribution with proxy re-encryption. *Future Gener. Comput. Syst.* **2017**, *71*, 177–191. [CrossRef]
34. Benaloh, J.; de Mare, M.; Accumulators, O.W. A Decentralized Alternative to Digital Signatures. In Proceedings of the Advances in Cryptology-Proceedings of Eurocrypt, Perugia, Italy, 9–12 May 1994; Volume 93.
35. Miers, I.; Garman, C.; Green, M.; Rubin, A.D. Zerocoin: Anonymous distributed e-cash from bitcoin. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, IEEE, Berkeley, CA, USA, 19–22 May 2013; pp. 397–411.
36. Ren, Y.; Lv, Z.; Xiong, N.N.; Wang, J. HCNCT:A Cross-chain Interaction Scheme for the Blockchain-based Metaverse. *ACM Trans. Multimed. Comput. Commun. Appl.* 2023, *accepted*. [CrossRef]

37. Wang, J.; Gao, Y.; Liu, W.; Wu, W.; Lim, S.J. An Asynchronous Clustering and Mobile Data Gathering Schema Based on Timer Mechanism in Wireless Sensor Networks. *Comput. Mater. Contin.* **2019**, *58*, 711–725. [CrossRef]
38. Wang, J.; Ju, C.; Gao, Y.; Sangaiah, A.K.; Kim, G.J. A PSO based energy efficient coverage control algorithm for wireless sensor networks. *Comput. Mater. Contin.* **2018**, *56*, 433–446.
39. Ren, Y.; Zhu, F.; Sharma, P.K.; Wang, T.; Wang, J.; Alfarraj, O.; Tolba, A. Data query mechanism based on hash computing power of blockchain in internet of things. *Sensors* **2019**, *20*, 207. [CrossRef] [PubMed]
40. Ge, C.; Susilo, W.; Liu, Z.; Xia, J.; Szalachowski, P.; Fang, L. Secure keyword search and data sharing mechanism for cloud computing. *IEEE Trans. Dependable Secur. Comput.* **2020**, *18*, 2787–2800. [CrossRef]
41. Barić, N.; Pfitzmann, B. Collision-free accumulators and fail-stop signature schemes without trees. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Konstanz, Germany, 11–15 May 1997; pp. 480–494.
42. Camenisch, J.; Lysyanskaya, A. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Proceedings of the Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference, Proceedings 22, Santa Barbara, CA, USA, 18–22 August 2002; pp. 61–76.
43. Nguyen, L. Accumulators from bilinear pairings and applications. In Proceedings of the Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, 14–18 February 2005; pp. 275–292.
44. Damgård, I.; Triandopoulos, N. Supporting Non-Membership Proofs with Bilinear-Map Accumulators. Cryptology ePrint Archive. 2008. Available online: https://eprint.iacr.org/2008/538 (accessed on 28 December 2008).
45. Barsoum, A.F.; Hasan, M.A. Provable multicopy dynamic data possession in cloud computing systems. *IEEE Trans. Inf. Forensics Secur.* **2014**, *10*, 485–497. [CrossRef]
46. Hao, Z.; Zhong, S.; Yu, N. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. *IEEE Trans. Knowl. Data Eng.* **2011**, *23*, 1432–1437.