IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 November 2022, accepted 17 December 2022, date of publication 26 December 2022,
date of current version 1 February 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3231604

RESEARCH ARTICLE

A Data Deduplication Scheme Based on DBSCAN With Tolerable Clustering Deviation

YAN TENG[1,2], HEQUN XIAN[1,2], QUANLI LU[3,4], AND FENG GUO[3,4]
[1]College of Computer Science and Technology, Qingdao University, Qingdao 266071, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[3]Shandong Zhengzhong Information Technology Company Ltd., Jinan 250014, China
[4]Shandong Digital Applied Science Research Institute Company Ltd., Jinan 250101, China

Corresponding author: Hequn Xian (xianhq@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702294, in part by the Provincial Natural Science Foundation of Shandong under Grant ZR2019MF058, and in part by the Open Project of the State Key Laboratory of Information Security under Grant 2020-MS-09.

ABSTRACT To protect data privacy, users prefer to store encrypted data in cloud servers. Cloud servers reduce the cost of storage and network bandwidth by eliminating duplicate copies. To address the potential internal data leakage problem, the concept of clustering deviation is proposed for the first time. We improve the DBSCAN algorithm to tolerate clustering deviation. A data deduplication scheme is built upon the new algorithm, which considers users as clustering samples. Instead of immediately re-clustering new users, a certain deviation is tolerated to assign the users to the existing classes. We determine the popularity of the data according to user clustering results and apply different encryption schemes to protect the security of unpopular data more effectively. The performance of the algorithm is analyzed and compared with other methods through experiments, and the results verify the feasibility and efficiency of the proposed deduplication scheme.

INDEX TERMS Deduplication, cloud storage, data popularity, DBSCAN.

I. INTRODUCTION

The development and application of cloud computing have led more and more users to store their data on the cloud server (CS) [1], [2], [3]. To save bandwidth and storage space, servers usually use data deduplication techniques, i.e., they maintain only a single copy of data and remove redundancy [4].

However, when uploading data to the CS, users want to encrypt the data to protect their privacy and prevent the data content from being obtained by CS or other attackers [5], [6]. However, with traditional encryption schemes, users randomly select keys and encrypt the plaintext. This makes the ciphertext stored on the CS different even for the same plaintext, which makes the deduplication operation very difficult. Conversely, when users encrypt the data with the same key, this can significantly reduce the security of the system.

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Mehmood.

Convergent encryption (CE) was proposed to solve this problem effectively [7]. In CE, the key is derived from the plaintext, so that the same plaintext produces the same key, which in turn produces the same ciphertext. This allows data deduplication of encrypted data. However, CE has security flaws and is vulnerable to offline brute-force attacks [8], because the key derivation process is deterministic.

In recent years, many researchers have worked on designing various Message-Locked Encryption (MLE)-based deduplication schemes [9], [10]. In response to the above attacks, Stanek et al. proposed a deduplication scheme based on popularity division [11]. Data with different popularity are encrypted using different encryption methods to further save cloud storage space and network bandwidth. Puzio et al. proposed a ClouDedup scheme with a metadata manager and an additional server defined in the CS: the server adds an encryption layer to prevent attacks against CE and thus protects the confidentiality of data [12]. DupLESS used a key manager to generate the key and applies the oblivious pseudorandom function (OPRF), which is a high-security algorithm [13]. The scheme of Zhang et al. used elliptic curve encryption algorithm to achieve data confidentiality, and different encryption methods were used for popular and unpopular data to reduce the computational overhead [14]. Liu et al. proposed a secure data deduplication scheme that does not require a third-party server [15]. This scheme adopts password-authenticated key exchange (PAKE) to implement cross-user key passing, thus achieving cross-user data deduplication. And it also eliminates the dependence on third-party servers and improves security. However, it requires all users involved in the protocol to be online when exchanging keys, which significantly increases the communication overhead and reduces the practicality.

Various existing data deduplication schemes focus on the protection and delivery of encryption keys and the identification of duplicate data while ignoring the impact of users on deduplication. Among the many schemes that differentiate data according to their popularity, for the data with fewer holders, a semantic security-compliant encryption scheme with higher security is used. When the number of data holders increases, the system considers that the data is less sensitive and uses a less secure encryption protection scheme such as CE. However, if the data belong to users from the same organization, such as a company's internal address book, the case will be different. That is, the increase in the number of data holders does not mean that its sensitivity decreases. If the system uses a less secure encryption protection scheme, it will cause the potential internal data leakage.

A. CONTRIBUTION

To address the above problems, we consider the effect of user attributes on the popularity recognization and propose a data deduplication scheme based on DBSCAN with tolerable clustering deviation (TCD-DBSCAN) algorithm. The contributions of this paper are as follows.

1) Propose a TCD-DBSCAN algorithm and use it for user clustering in data deduplication scheme to reduce the risk of internal data leakage.
2) Dynamic counting is performed based on the clustering results, and then the popularity of the data is determined based on whether the "number" of data holders reaches the popularity threshold, so as to avoid the premature conversion of internal privacy data to popular data.
3) A bilinear mapping is adopted to construct data tags for identifying the original plaintext. For unpopular data, an RSA-based blind signature algorithm is used between the user and key manager.

B. ORGANIZATION

The rest of this paper is organized as follows. In Section II, the required preliminaries for this paper are presented. In Section III, we describe the system model and definitions. In Section IV, we propose the construction of the scheme. Section V presents the security analysis. We describe the simulation and experimental analysis of the scheme in Section VI. In Section VII, we summarize the paper.

II. PRELIMINARIES

In this section, we briefly introduce the basic knowledge about RSA-based blind signature algorithm, bilinear mapping,DBSCAN algorithm and attribute similarity calculation.

A. RSA-BASED BLIND SIGNATURE ALGORITHM

This scheme adopts an RSA-based blind signature algorithm to generate the encryption key by the interaction between the Key manager and the user without any information disclosure [16]. Key manager holds the private key d and publishes the public key e, where $e \cdot d \equiv 1 mod \varphi(n)$. The user chooses random value $r \in \mathrm{Z}^*_N$ and generates $x \equiv h \cdot \mathrm{r}^e mod\ n$, where h is the hash value of the plaintext M. Then, the user sends x to Key manager. Once received x, Key manager calculates $y \equiv \mathrm{x}^d mod\ n$ and returns it to the user, the user removes the blind value r and obtains the secret value $z \equiv y \cdot \mathrm{r}^{-1} mod\ n$. Finally, the user can use the encryption key $K \leftarrow \mathrm{H}_2(z)$ to encrypt the data.

B. BILINEAR MAPPING

Let G,G_T be 2 multiplicative cyclic groups of order q, where q is a large prime, g is a generating element of the group G. Define the mapping relation e: $G \times G \to \mathrm{G}_T$, and satisfy the following properties [17]:

1) Computability:For $\forall\ \mathrm{g}_1, \mathrm{g}_2 \in G$,there are valid and efficient algorithms to calculate $e(\mathrm{g}_1, \mathrm{g}_2)$
2) Bilinearity:For $\forall\, \mathrm{g}_1, \mathrm{g}_2 \in G$,and $a, b \in Z_q$,$e({\mathrm{g}_1}^a, {\mathrm{g}_2}^b) = e(\mathrm{g}_1, \mathrm{g}_2)^{ab}$;
3) Non-degeneracy: $\exists\ \mathrm{g}_1, \mathrm{g}_2 \in G$, $e(\mathrm{g}_1, \mathrm{g}_2) \neq 1$.

C. DBSCAN ALGORITHM

Density-based spatial clustering of applications with noise (DBSCAN) is an unsupervised machine learning clustering algorithm [18].There are two important parameters in the DBSCAN algorithm: $Eps(\epsilon)$ and $MinPts$, the former being the neighborhood radius when defining the density and the latter being the threshold value when defining the core point [19].

DBSCAN classifies sample points into three classes:

1) Core point:if at least $MinPts$ samples exist in the ϵ-neighborhood of sample p,i.e.$|\ N\ \in(p)| \geq MinPts$, sample p is a core point.
2) Border point:if the number of samples in the ϵ-neighborhood of sample p is less than $MinPts$, then sample p is a border point.
3) Noise point:a point that is neither a core point nor a border point.

D. ATTRIBUTE SIMILARITY CALCULATION

When discussing the calculation of attribute distances, attributes are classified as "ordinal attribute" and "non-ordinal attribute" depending on whether the attributes define an "ordered" relationship. For example, an attribute such as a user's IP address can determine the distance directly on the attribute value, which is referred to as an ''ordinal attribute''; an attribute such as a user's domain name cannot calculate the distance directly on the attribute value, which is referred to as a ''non-ordinal attribute''. Mixed attributes are handled by combining the Minkowski distance and value difference metric (VDM). Users $x_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ and $x_j = \{x_{j1}, x_{j2}, \ldots, x_{jn}\}$ have n_c ordinal attributes and n-n_c non-ordinal attributes, then the distance between them is calculated as shown in Eq.(1).

$$\mathrm{MinkovDM}_p(x_i, x_j) = \Big(\sum_{u=1}^{n_c} |x_{iu} - x_{ju}|^p + \sum_{u=n_c+1}^{n} \mathrm{VDM}_p(x_{iu}, x_{ju})\Big)^{\frac{1}{p}} \quad (1)$$

III. SYSTEM MODEL AND DESIGN GOALS

In this section, we describe system model, threat model and the security goals.

A. SYSTEM MODEL

The system model of this scheme, including the cloud server, key manager and the user. As shown in Fig.1.

1) CLOUD SERVER (CS)

CS, an entity with huge computing power and storage capacity, mainly provides cloud storage services. Apply the TCD-DBSCAN algorithm to count the legal upload users and deduplicate the data to improve storage utilization.

2) KEY MANAGER (K-MAN)

K-man, a semi-trusted third party which is responsible for helping users generate encryption keys for unpopular data by performing the RSA-based blind signature algorithm. Users have access to K-man, and we will further explain the security of user interaction with K-man in Section 5.

3) USER (U)

Users, as the clients of the cloud storage system, require to upload and download data and other operations securely and conveniently on the CS. When uploading unpopular data, a blind signature operation with K-man is required to obtain encryption keys to ensure their data security.When uploading popular data, users only need to perform simple and efficient convergent encryption.

B. THREAT MODEL

We consider the following two types of attackers:

1) INTERNAL ATTACKERS

Meaning CS and potentially malicious users. We consider that CS is honest but curious and can have arbitrary access to their stored user data, potentially malicious users can interact with CS following all the protocols but they want to illegally access the data of other users.

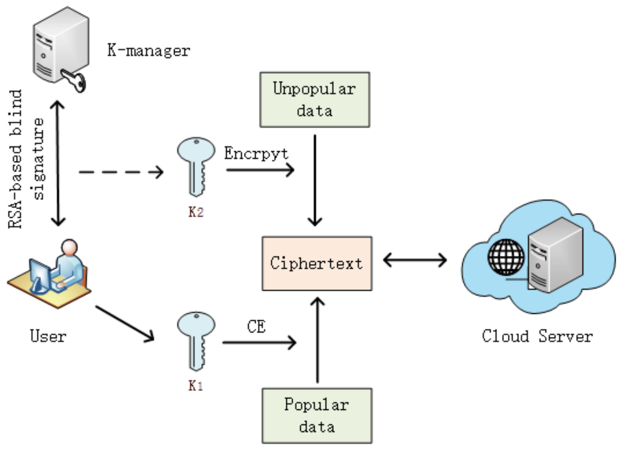

FIGURE 1. System model.

2) EXTERNAL ATTACKERS

Meaning unauthorized users. They obtain information about part of the uploaded data by tapping the public channel of the internet, and their main purpose is to illegally obtain plaintext information about the data stored on the CS.

C. SECURITY GOALS

The security goals of the scheme in this paper are as follows:

1) Users are classified according to their attributes, so that internal data, which come from the same organization will not be prematurely turned into popular data.
2) All data are protected by encryption methods and the attackers should not get any plaintext information about them.
3) This scheme is resistant to online and offline brute-force attack.

IV. PROPOSED SCHEME

We propose a TCD-DBSCAN algorithm to solve the problem of the potential internal data leakage in deduplication. The algorithm classifies users based on their attributes. The data is classified into popular data and unpopular data according to whether the counting of data owners reaches the popularity threshold. Different encryption methods are adopted for data with different popularity, so as to balance security and efficiency.

A. TCD-DBSCAN

Conventional DBSCAN algorithm require re-clustering when the data set is changed, which will greatly increase the computational overhead of the system. We first propose the concept of clustering deviation, i.e., ignoring certain errors and preferentially classifying newly added points into their nearby classes or treating them as noise points. Based on this, we designed a DBSCAN algorithm with tolerable clustering deviation algorithm. First, a conventional DBSCAN clustering is performed on the currently existing data set; thereafter, whenever a new point q appears, instead of re-clustering, the operation is performed in the following steps. The specific process is shown in Algorithm 1:

Algorithm 1 TCD-DBSCAN Algorithm

```
Input: $D$,$\epsilon$,$minPts$
Output: $D^{\star}$,$x_1$,$x_2$
 1: Flag=0
 2: for each $a \in D$ do
 3:     $d$=distance<a,q>
 4:     if $d \leq \epsilon$ then
 5:         $C$.add($q$);$x_1 = d$;$x_2$=Count($C$)
 6:         //Break, Flag=1
 7:     end if
 8: end for
 9: if Flag=0 then
10:     for each $p \in D$ do
11:         $d$ = distance<p,q>
12:         if $d \leq \epsilon$ then
13:             $C$.add($q$);$x_1$=$d$;$x_2$=Count($C$)
14:             //Mark q as a deviation point $dev$, Flag=1
15:         end if
16:     end for
17:     if $|N \in (q)| \geq minPts$ then
18:         //Create a new class whose core point is q
19:     else
20:         //Mark q as a noise point
21:     end if
22: end if
23: if Count($dev$)$\geq \Omega$ then
24:     //Run a DBSCAN algorithm
25: end if
```

If q is a point near the core point a, it is classified in the class where a is located. If a border point p exists near q, assign it to the class in which p exists, and record q as a deviation point. Determine whether q meets the core point condition and if so, assign it to the class whose density is connected. If none of the above conditions is met, q is treated as a noise point. In order to reduce the error, when the number of deviation points exceeds a threshold Ω, it will be re-clustered.

B. NOTATIONS

Table 1 shows some notations used in the scheme.

C. SYSTEM INITIALIZATION

In the initialization phase, a public-private key pair $\{P_U, P_R\}$ is assigned to K-man. And a unique identity ID_i is determined for each U_i who joins the system and assigned a unique public-private key pair $\{pk_i, sk_i\}$. The file tag list *File_List* is stored on the *CS* and the file information table $DB[T_F]$ associated with it contains four records: $DB[T_F].data$ is the stored ciphertext, $DB[T_F].user$ is the list of legal users, $DB[T_F].Count$ is the number of legal users, and $DB[T_F].Tag$ is the file popularity tag.

TABLE 1. Notations.

Notations	Meanings
M	Plaintext of file F
K_1	Hash value of the plaintext, $K_1 \leftarrow H_1(M)$
K_2	Key for unpopular files, obtained after blind signature,$K_2 \leftarrow KGen(M)$
C_1	Ciphertext of file F,$C_1 \leftarrow Enc(K_1,M)$
C_2	Ciphertext of file F,$C_2 \leftarrow Enc(K_2,M)$
H_i	Secure hash functions,H_1:$\{0,1\}^{\star} \rightarrow Z^{\star}{}_N$,$H_2$:$Z^{\star}{}_N \rightarrow \{0,1\}^{\star}$
$\{pk_i, sk_i\}$	A public-private key pair for U_i
$\{P_U, P_R\}$	A public-private key pair for K-man
T_F	File tag, $T_F \leftarrow TGen(sk,C_1)$
re	Result of the tag finding function takes the value of 1 or 0
$\varphi(x_1, x_2)$	The growth curve function
$countT_F$	Size of the weight occupied by the file F, in the range of 0 to 1
t	Popularity threshold of file F
num_t	Maximum of RSA blind signatures executed in a time period
num_c	Maximum of RSA blind signature requests sent when uploading data M
α	Length of the hash $H(M)$
β	Minimum entropy of M
x	Number of potential users who have M

D. FILE UPLOAD

File upload means that the user uploads data to CS, which can be divided into unpopular file upload and popular file upload. First, U_i send the *upload_request* $\|$ ID_i $\|$ $T_{F,i}$ $\|$*user attribute* to CS. After CS receives the request from U_i, it performs data duplication detection and runs the tag finding function $re \leftarrow TCheck(T_{F,i}, File\_List)$. Fig.2 is the file upload process.

1) UNPOPULAR FILE UPLOAD

If the uploaded file F does not exist in the CS, F is the initial file. CS initialize the corresponding file information table to store the file ciphertext and update the user list. When the uploaded file F exists in CS and $DB[T_F]$. $Count$ $< t$, F is an unpopular file.CS updates the information in the corresponding *File_List* and adds U_i to the list of legitimate users.The detailed process is described in Fig.3. In particular, CS applies the TCD-DBSCAN algorithm to transform the process of unpopular file uploads and adopts the *Growth Curve function* to calculate the weight number occupied by the current uploader. The growth curve model φ can be expressed in Eq.(2):

$$y = \frac{k}{x_1 + a \cdot a^{b \cdot x_2}} \tag{2}$$

where x_1 and x_2 are the independent variables, y is the dependent variable, and a,b,k are the parameters.

2) POPULAR FILE UPLOAD

The uploaded file F exists in CS and $DB[T_F].Count = t$. Then the file F undergoes a popularity conversion. The CS updates the information in the corresponding file information table asks the user to upload convergent encrypted ciphertext. If the uploaded file F exists in the CS and $DB[T_F]$. $Count$ $>$ t, F is a popular file. CS adds U_i to the list of legal users, and does not require the user to upload the convergent encrypted ciphertext. The detailed process is described in Fig.4.

V. SECURITY ANALYSIS

In this section, we formally prove that our scheme is correct and secure.

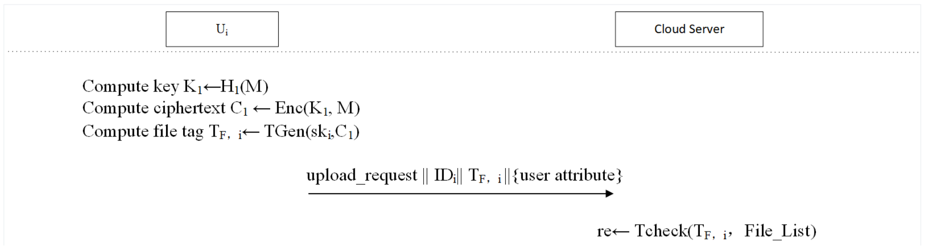
U_i
Cloud Server
Compute key K1←H1(M)
Compute ciphertext C1 ← Enc(K1, M)
Compute file tag TF, i← TGen(ski,C1)
upload_request || IDi|| TF, i||{user attribute}
re← Tcheck(TF, i, File_List)

FIGURE 2. File upload process.

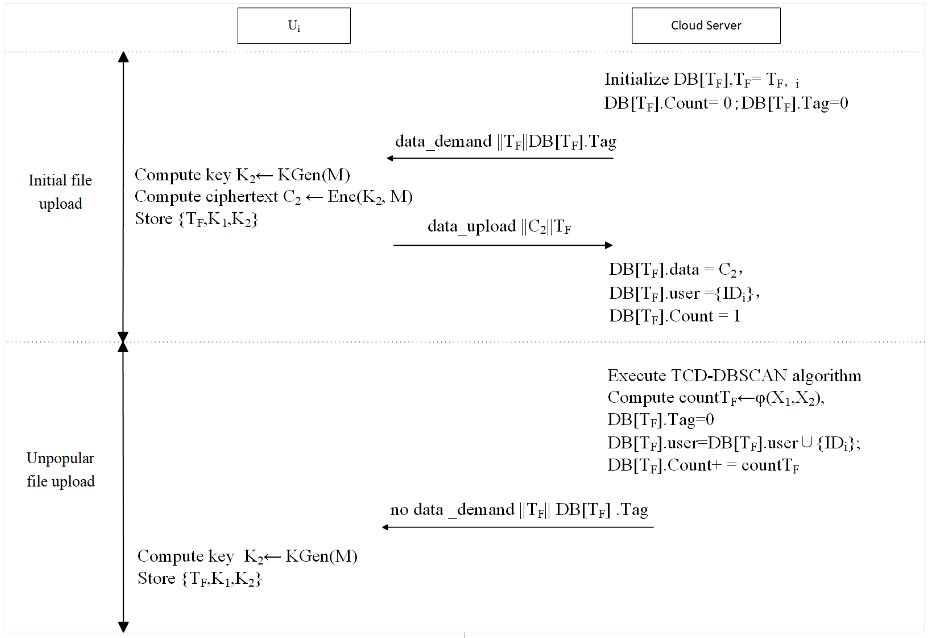

FIGURE 3. Unpopular file upload process.

A. TAG SECURITY

In this scheme, the user calculates the file tag, and the CS detects the file repeatability based on the bilinear mapping. We analyze and prove the correctness and uniqueness of the file tag.

1) TAG CORRECTNESS

Theorem 1: The initial uploader of file F, U_i, first computes the file tag $\mathrm{T}_{F,i} =< \mathrm{L}_i,\&_i >$, and uploads it to the CS to save it as the unique identifier of F. Then, the subsequent uploader of F, U_j, computes the file tag $\mathrm{T}_{F,j} =< \mathrm{L}_j,\&_j >$, which must satisfy $e(\mathrm{L}_i,\&_j) = e(\mathrm{L}_j,\&_i)$.

Proof: According to the tag generation function, $\mathrm{L}_i = \mathrm{g}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_i}$,$\&_i = \mathrm{g}^{\mathrm{sk}_i}$, $\mathrm{L}_j = \mathrm{g}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_j}$, $\&_j = \mathrm{g}^{\mathrm{sk}_j}$, where C_1 is the convergent ciphertext corresponding to F. By the nature of bilinear mapping, the following equation can be obtained:

$$\begin{aligned} e(\mathrm{L}_i, \&_j) &= e(\mathrm{g}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_i}, \mathrm{g}^{\mathrm{sk}_j}) \\ &= \mathrm{e(g,g)}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_i\cdot \mathrm{sk}_j} \\ &= \mathrm{e(g,g)}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_j\cdot \mathrm{sk}_i} \\ &= e(\mathrm{g}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_j}, \mathrm{g}^{\mathrm{sk}_i}) = e(\mathrm{L}_j, \&_i). \end{aligned}$$

■

2) TAG UNIQUENESS

Theorem 2: Let the initial uploader of file F, U_i, calculate the file tag $\mathrm{T}_{F,i} =< \mathrm{L}_i,\&_i>$, and upload it to the CS to save it as the unique identifier of F. When the user U_j uploads the file F', the file tag $\mathrm{T}_{\mathrm{F}',j} =< \mathrm{L}_j,\&_j >$ is calculated and uploaded to the CS for data duplicity detection. The scheme guarantees the uniqueness of data tag, i.e., there exists $F \neq \mathrm{F}'$ such that the probability that $e(\mathrm{L}_i,\&_j) = e(\mathrm{L}_j,\&_i)$ is negligible.

Proof: The proof is carried out using the converse method. Suppose there exists $F \neq \mathrm{F}'$ such that $e(\mathrm{L}_i,\&_j) = e(\mathrm{L}_j,\&_i)$.

$$\begin{aligned} e(\mathrm{L}_i, \&_j) &= e(\mathrm{L}_j, \&_i) \\ &\Leftrightarrow e(\mathrm{g}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_i}, \mathrm{g}^{\mathrm{sk}_j}) = e(\mathrm{g}^{H(\mathrm{C}_{1'})\cdot \mathrm{sk}_j}, \mathrm{g}^{\mathrm{sk}_i}) \\ &\Leftrightarrow \mathrm{e(g,g)}^{H(\mathrm{C}_1)\cdot \mathrm{sk}_i\cdot \mathrm{sk}_j} = \mathrm{e(g,g)}^{H(\mathrm{C}_{1'})\cdot \mathrm{sk}_j\cdot \mathrm{sk}_i} \\ &\Leftrightarrow H(\mathrm{C}_1) = H\left(\mathrm{C}_{1'}\right) \end{aligned}$$

when $F \neq \mathrm{F}'$,$H(\mathrm{C}_1) \neq H(\mathrm{C}_{1'})$. Without loss of generality, if $e(\mathrm{L}_i, \&_j) = e(\mathrm{L}_j, \&_i)$, then $H(\mathrm{C}_1) = H(\mathrm{C}_{1'})$, which contradicts the assumption, so the assumption is not valid. That is, $e(\mathrm{L}_i, \&_j) = e(\mathrm{L}_j, \&_i)$ is satisfied when and only when $F = \mathrm{F}'$. This proves the uniqueness of the file tag. ■

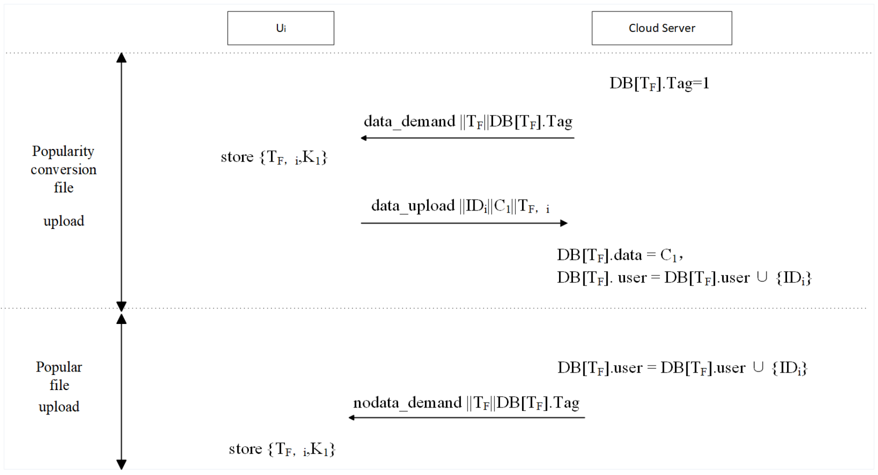
Ui
Cloud Server
DB[TF].Tag=1
data_demand ||TF||DB[TF].Tag
Popularity conversion file upload
store {TF, i,K1}
data_upload ||IDi||C1||TF, i
DB[TF].data = C1,
DB[TF]. user = DB[TF].user ∪ {IDi}
Popular file upload
DB[TF].user = DB[TF].user ∪ {IDi}
nodata_demand ||TF||DB[TF].Tag
store {TF, i,K1}

FIGURE 4. Popular file upload process.

B. KEY SECURITY

The key generation for unpopular files is acquired by user interaction with K-man using a blind signature algorithm based on RSA. This scheme takes full advantage of the nature of blind signatures to ensure the security of the scheme itself: No individual except K-man can generate a valid blind signature in its name. The user obtains z after decrypting the blind signature y and determines whether y is a signature of K-man by verifying $V(h,z) = TURE \Leftrightarrow h \equiv Z^e mod\ n$.

C. DATA PRIVACY

1) CHANNEL EAVESDROPPING ATTACK

By listening to the communication channel, the malicious user MU gets the file tag T_F and ciphertext C. Based on the information acquired, MU performs a brute-force attack on the data M.

Game 5.3:

1) MU lists all possible data sets $\{M_i\}$, and compute $\{H(M_i)\}, |M| = mSize$, where $i \in [1, 2^{mSize}]$;
2) MU lists all sets of random number $\{r_j\}, |r| = rSize$, where $j \in [1, 2^{rSize}]$;
3) According to the key generation algorithm, $K_1 = \{k | k = H(M_i)\}, K_2 = \{k | k = KGen(r_j, H(M_i))\}$;
4) Listing all possible sets of popular data ciphertext, $C_1 = \{C_t | C_t = Enc(k, M_i)\}$, $k \in K_1$, where $t \in [1, 2^{mSize}]$; Unpopular data ciphertext sets, $C_2 = \{C_n | C_n = Enc(k, M_i)\}$, $k \in K_2$, where $n \in [1, 2^{mSize+rSize}]$.

If there exists $C_t = C$ or $C_n = C$ in the set of exhaustive ciphertexts, the adversary wins the game to obtain the plaintext data. The time complexity of the data for MU to successfully crack the data is $O(2^{mSize})$ or $O(2^{mSize+rSize})$, which is computationally infeasible, so the scheme is resistant to attacks from malicious users.

2) OFFLINE BRUTE-FORCE ATTACK

CS attempts an offline brute-force attack on the user uploaded file tag $T_F = e(g^{H(C_1) \cdot sk}, g^{sk})$ with CS knows g, public key pk, prime p. The attack is as follows:

1) Exhaust the data sets $\{M_i\}$ and compute $H(Enc(M_i, H(M_i)))$ denoted as $H(C_i)$, $|M| = n$, where $i \in [1, n]$;
2) Exhaust set of private keys $\{sk_j\}, |sk_j| = s$, where $j \in [1, s]$, and use the bilinear mapping to calculate the set of tags, as shown in Eq.(3), at the bottom of the next page.
3) Compare the results of the above calculation with T_F one by one and determine whether there is a value equal to it, if there is an equal value then proves that $M_i = M$.

Lemma 1: Discrete logarithm problem (DL problem): Let G be the multiplicative group of large prime number P, where g is the generating element, and for a given $Q = x^P \in G$, compute $x \in Z^\star_N$

Lemma 2: Diffie-Hellman Problem (CDH problem):Let G be the multiplicative group of large prime numbers P, where g is the generating element. the CDH problem can be described as follows: for a given $g, g^a, g^b \in G$, compute $Q = g^{ab} \in G$, where a, b are unknown integers.

Theorem 3: The scheme is resistant to offline brute-force attack.

Proof: Without loss of generality. From **Lemma 1**, the public key pk_i of U_i is known and guessing the secret key sk_i is difficult. From **Lemma 2**, it is known that $e(g^{sk_i}, g^{sk_i})$and $e(g^{H(C_1)}, g^{sk_i})$ to compute $e(g^{H(C_1) \cdot sk_i}, g^{sk_i})$ is infeasible. Therefore, CS cannot perform the computation to get the tag set merged for comparison. ■

3) ONLINE BRUTE-FORCE ATTACK

To prevent online brute-force attacks by malicious users, our scheme also adopts a rate-limiting policy to limit the frequency of user-K-man interactions [15]. When Eq.(4)

$$num_t + num_c < \frac{2^{\beta-\alpha}}{x} \tag{4}$$

is satisfied, malicious users can be effectively prevented from performing online brute-force attacks on U_i.

VI. SIMULATION AND EXPERIMENTAL ANALYSIS

The experiments adopt the OpenSSL [20], PBC [21], and GMP [22] cryptographic function libraries and implement

TABLE 2. Communication overhead comparison.

	Initial file upload	Unpopular file upload	Popular file upload	File download
Stanek [11]	$2C_T+C_C+C_K+C_{ID}$	$2C_T+C_C+C_K+C_{ID}$	$2C_T+C_C+C_{ID}$	$C_T+C_C+C_K+C_{ID}$
Gao [24]	$C_T+C_C+C_{rk}+C_{ID}$	$C_T+C_{rk}+C_{ID}$	C_T+C_{ID}	$C_T+C_C+C_{ID}$
Ours	$2C_T+C_C+C_{ID}$	C_T+C_{ID}	C_T+C_{ID}	$C_T+C_C+C_{ID}$

TABLE 3. Storage overhead comparison.

	Cloud Server Side	User side
Stanek [11]	$NC_T+NC_C+NC_{ID}$	$2C_T+2C_K+C_{sk}$
Gao [24]	$C_T+C_C+C_{rk}+NC_{ID}$	$C_T+2C_K+C_{sk}$
Ours	$C_T+C_C+NC_{ID}$	$C_T+2C_K+C_{sk}$

the client and server software using the C++ programming language.We adopt the MD5 hash function to generate the data tags, SHA-256 is used to generate the convergent encryption key, and 256-bit strength AES is used to encrypt and decrypt the data.To simulate the Internet application environment, more than 500 files are stored on the *CS* and have been classified according to their attributes. We divide this scheme into four cases, i.e. initial file upload, unpopular file upload, popularity conversion file upload, and popular file upload. Each part of the operation is repeated 10 times and the average value is taken as the final result. The *Employee Attrition Analytics* dataset [23] is adopted for the experiment, and the dataset is normalized and the features are downscaled to present better experimental results.

A. PERFORMANCE ANALYSIS

We analyze the performance of the communication overhead and compare it with Stanek et al. [11] and Gao et al. [24], as shown in Table 2. For file F, C_T,C_C denote data tag size and ciphertext size respectively, C_K, C_{rk},C_{sk} denote data encryption key size, random key size and user private key size respectively. C_{ID} denotes the user identification size. Table 2 shows that our scheme is prior to other schemes in terms of communication overhead.

In Table 3, we compare the storage overhead of unpopular data with Stanek et al. [11] and Gao et al. [24]. N is the number of legitimate users of unpopular data. From Table 3, we can see that our scheme outperforms other schemes in terms of storage overhead for unpopular data.

B. TCD-DBSCAN EFFICIENCY

For the problem of selecting the parameters of the algorithm in this experiment, we follow the method proposed in [25]. The value of *Eps* can be obtained using the method of drawing k-distance graph, and there is a rule of thumb for the selection of *MinPts*, $MinPts \geq dim + 1$,where *dim* denotes the data to be clustered dimensionality of the data to be clustered. Finally, after continuously adjusting the parameters, the value of *Eps* for this dataset is determined to be 1.85 and the value of *MinPts* is 12.

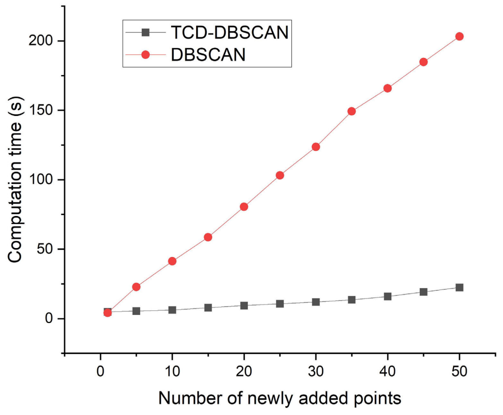

FIGURE 5. Time overhead comparison.

To simulate a real Internet application scenario, we store more than 500 different files in CS, each file is associated with multiple users. Based on the attributes of the uploaders, we clustered the users for each file. Then, new users with different attributes are simulated to perform an upload operation on a specific file, and the running time of the algorithm is recorded and compared with the conventional DBSCAN algorithm, and the test results are shown in Fig.5.

It can be seen that the TCD-DBSCAN algorithm is much better than the conventional DBSCAN algorithm in terms of computational efficiency. And the advantage of the TCD-DBSCAN algorithm becomes more obvious as the size of subsequent uploaders increases.

C. COMMUNICATION OVERHEAD

According to the characteristics of the scheme, we test the computational time overhead for four cases, each case includes tag generation, key generation, data encryption, and file uploading. We upload files of different sizes to CS and

$$\begin{pmatrix} e(g^{H(C_1)\cdot sk_1}, g^{sk_1}) & e(g^{H(C_1)\cdot sk_2}, g^{sk_2}) & e(g^{H(C_1)\cdot sk_3}, g^{sk_3}) & \ldots & e(g^{H(C_1)\cdot sk_s}, g^{sk_s}) \\ e(g^{H(C_2)\cdot sk_1}, g^{sk_1}) & e(g^{H(C_2)\cdot sk_2}, g^{sk_2}) & e(g^{H(C_2)\cdot sk_3}, g^{sk_3}) & \ldots & e(g^{H(C_2)\cdot sk_s}, g^{sk_s}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e(g^{H(C_n)\cdot sk_1}, g^{sk_1}) & e(g^{H(C_n)\cdot sk_2}, g^{sk_2}) & e(g^{H(C_n)\cdot sk_3}, g^{sk_3}) & \ldots & e(g^{H(C_n)\cdot sk_s}, g^{sk_s}) \end{pmatrix} \tag{3}$$

TABLE 4. Comparison of scheme features.

Scheme	Data popularity division	Unpopular data deduplication	Data integrity	Consider user attributes
Stanek [11] PerfectDedup [26]	Yes	No	No	No
Zhang [14] Gao [24]	Yes	Yes	Yes	No
Yuan et al. [27]	No	Yes	Yes	No
Wang et al. [28]	No	Yes	Yes	No
Ours	Yes	Yes	Yes	Yes

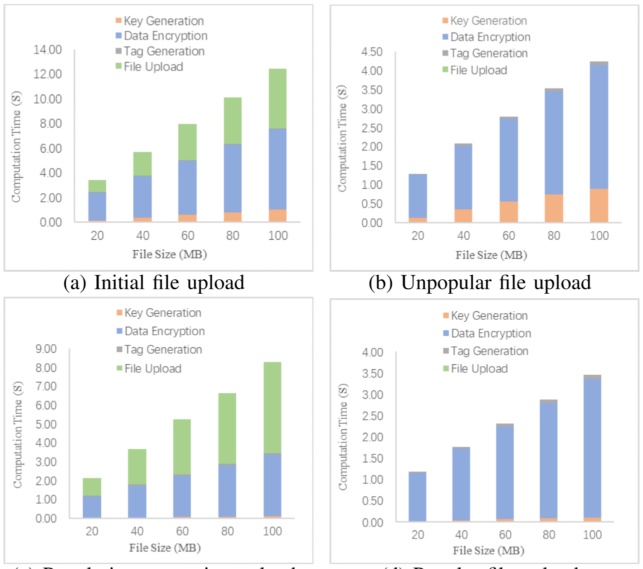

FIGURE 6. Time overhead of each case.

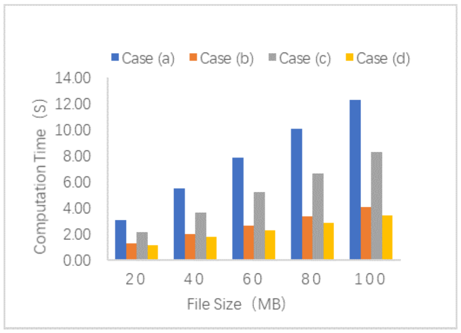

FIGURE 7. Total time overhead.

perform simulation experiments. Fig.6 shows the time overhead of different file size in each case. And the total time overhead statistics for the four cases are shown in Fig.7.

When compared with similar schemes such as Stanek et al. [11], PerfectDedup [26], Zhang et al. [14], and Gao et al. [24] in the same network environment the same file size (20MB), the result is shown in Fig.8. Overall, the scheme we proposed is prior to other schemes in terms of time overhead.

D. CHARACTERISTICS

The proposed scheme is compared with Stanek [11], PerfectDedup [26], Zhang [14], Gao [24], Yuan [27] and

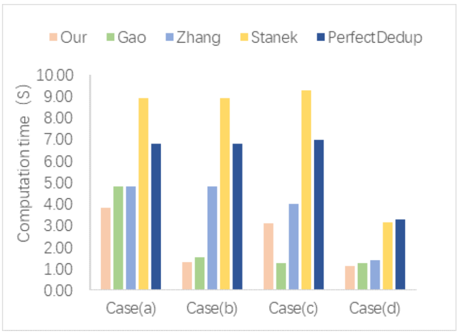

FIGURE 8. Time overhead comparison.

Wang [28] in terms of scheme characteristics, and the results are shown in Table 4.

Table 4 shows that this scheme considers the impact of user attributes on data popularity classification and deduplicates the unpopular data. It prevents internal data leakage, reduces storage overhead and improves deduplication efficiency.

VII. CONCLUSION

In this paper, we study the issue of encrypted data deduplication and propose a TCD-DBSCAN algorithm. The concept of clustering deviation is proposed, and our algorithm is applied in the deduplication process to reduce the risk of internal data leakage. Premature conversion of unpopular data is eliminated even if the data are uploaded by users from the same organization. For unpopular data, symmetric encryption is adopted and the encryption key is obtained by blind signature protocol, between the Key manager and the users. So that, deduplication of unpopular data can be achieved without user passing their keys online, which further improves the efficiency of deduplication. Security analysis and performance evaluation demonstrate that the proposed scheme is secure and of great practical value.

REFERENCES

[1] Y. Fan, X. Lin, W. Liang, G. Tan, and P. Nanda, "A secure privacy preserving deduplication scheme for cloud computing," *Future Gener. Comput. Syst.*, vol. 101, pp. 127–135, Dec. 2019.
[2] Z. Yan, W. Ding, X. Yu, H. Zhu, and R. H. Deng, "Deduplication on encrypted big data in cloud," *IEEE Trans. Big Data*, vol. 2, no. 2, pp. 138–150, Jun. 2016.
[3] Y. Zhai, M. Ibrahim, Y. Qiu, F. Boemer, Z. Chen, A. Titov, and A. Lyashevsky, "Accelerating encrypted computing on Intel GPUs," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2022, pp. 705–716.
[4] X. Yang, R. Lu, J. Shao, X. Tang, and A. A. Ghorbani, "Achieving efficient and privacy-preserving multi-domain big data deduplication in cloud," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1292–1305, Sep. 2021.

[5] C. Guo, X. Jiang, K.-K.-R. Choo, and Y. Jie, "R-dedup: Secure client-side deduplication for encrypted data without involving a third-party entity," *J. Netw. Comput. Appl.*, vol. 162, Jul. 2020, Art. no. 102664.

[6] X. Tang, L. Zhou, B. Hu, and H. Wu, "Aggregation-based tag deduplication for cloud storage with resistance against side channel attack," *Secur. Commun. Netw.*, vol. 2021, pp. 1–15, Feb. 2021.

[7] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *Proc. 22nd Int. Conf. Distrib. Comput. Syst.*, 2002, pp. 617–624.

[8] L. Wang, B. Wang, W. Song, and Z. Zhang, "A key-sharing based secure deduplication scheme in cloud storage," *Inf. Sci.*, vol. 504, pp. 48–60, Dec. 2019.

[9] Y. Zhao and S. S. M. Chow, "Updatable block-level message-locked encryption," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 449–460.

[10] H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, and R. H. Deng, "Secure cloud data deduplication with efficient re-encryption," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 442–456, Jan. 2022.

[11] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, *A Secure Data Deduplication Scheme for Cloud Storage*. Berlin, Germany: Springer, 2013.

[12] P. Puzio, R. Molva, M. Onen, and S. Loureiro, "ClouDedup: Secure deduplication with encrypted data for cloud storage," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2013, pp. 363–370.

[13] M. Bellare, S. Keelveedhi, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. Usenix Conf. Secur.*, 2013, pp. 1–17.

[14] S. G. Zhang, H. Q. Xian, Y. Z. Wang, H. Y. Liu, and R. T. Hou, "Secure encrypted data deduplication method based on offline key distribution," *J. Softw.*, vol. 29, no. 7, pp. 1909–1921, 2018.

[15] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 874–885.

[16] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme," *J. Cryptol.*, vol. 16, no. 3, pp. 185–215, 2003.

[17] V. S. Miller, "The Weil pairing, and its efficient calculation," *J. Cryptol.*, vol. 17, no. 4, pp. 235–261, 2004.

[18] N. Gholizadeh, H. Saadatfar, and N. Hanafi, "K-DBSCAN: An improved DBSCAN algorithm for big data," *J. Supercomput.*, vol. 77, no. 6, pp. 6214–6235, Jun. 2021.

[19] A. Starczewski and A. Cader, "Grid-based approach to determining parameters of the DBSCAN algorithm," in *Proc. Int. Conf. Artif. Intell. Soft Comput.*, 2020, pp. 555–565.

[20] H. U. Xiao-Ting, "Research and improved implementation of AES algorithm in OpenSSL," *Microcomput. Inf.*, vol. 25, no. 12, pp. 83–85, 2009.

[21] *The Pairing-Based Cryptographic Library*. [Online]. Available: http://crypto.Stanford.edu/pbc/

[22] M. Lukides, "Programming with gnu software," *J. Educ. Psychol.*, vol. 86, no. 86, pp. 350–359, 1995.

[23] D. K. Srivastava and P. Nair, "Employee attrition analysis using predictive techniques," in *Proc. Int. Conf. Inf. Commun. Technol. Intell. Syst.*, 2018, pp. 293–300.

[24] G. Wen-Jing, X. He-Qun, T. Cheng-Liang, L. Zeng-Peng, and H. Yun-Long, "A cloud storage deduplication method based on double-layered encryption," *J. Cryptologic Res.*, vol. 7, no. 5, p. 698, 2020.

[25] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise*. Palo Alto, CA, USA: AAAI Press, 1996.

[26] P. Puzio, R. Molva, M. Oenen, and S. Loureiro, "Perfectdedup: Secure data deduplication," in *Proc. 7th Int. Workshop Data Privacy Manage.*, 2015.

[27] H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, and R. H. Deng, "Secure cloud data deduplication with efficient re-encryption," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 442–456, Jan. 2022.

[28] Y. Wang, M. Miao, J. Wang, and X. Zhang, "Secure deduplication with efficient user revocation in cloud storage," *Comput. Standards Interface*, vol. 78, Oct. 2021, Art. no. 103523.

YAN TENG is currently pursuing the master's degree with the College of Computer Science and Technology, Qingdao University, China. Her research interests include cloud storage security and cloud computing security.

HEQUN XIAN received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 2009. He was a Visiting Scholar at the College of Information Science and Technology, Pennsylvania State University. His research interests include cryptography, cloud computing security, and network security.

QUANLI LU is the Chief Technology Officer of Shandong Zhengzhong Information Technology Company Ltd., China. His research interests include cloud computing, big data, and cryptography.

FENG GUO is the Product Manager of Shandong Zhengzhong Information Technology Company Ltd., China. His research interests include asymmetric cryptographic algorithm and privacy-preserving computing.