JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

SPRPLDA: Supporting Strong Privacy Protection and Recovery through Progressive Lightweight Deduplication and Auditing Method for secure data storage in cloud environment

Sulthana ASR
Research Scholar
Rathinam college of arts and science

Dr. K. Juliana Gnanaselvi
Principal
St.Paul's college of arts and science

Abstract: Cloud data storage outsourcing has emerged as a service that can facilitate data owners (DOs) to store their data remotely. However, data storage security and integrity remains challenging due to lack of control and physical possession over the data by the DOs. On the other hand, all of the existing cloud storage auditing schemes with deduplication are vulnerable to brute-force dictionary attacks and content guessing attack well, which incurs the leakage of user privacy. In this work, focused on a new aspect of being against brute-force dictionary attacks on cloud storage auditing based on Strong Privacy Protection and Recovery through Progressive Lightweight Deduplication and Auditing Method (SPRPLDA). Also this research work propose the efficient recovery technique for recovering the deleted based on Hybridization of particle swarm optimization (PSO) and artificial bee colony (ABC) approaches, based on recombination procedure (HPSOABC) which is more competitive optimization approach. In addition to that a novel SPRPLDA method is designed to generate the file index for duplicate check, data recovery using and use a new Enriched Dual Integrity Convergent Encryption (EDICE) to generate the key for file encryption with hash and offset value. Finally, the Date Owner only needs to perform progressive lightweight computation to generate data authenticators, verify cloud data integrity, and retrieve the file from the cloud.

Keyword: Cloud computing, Security, Data storage, Integrity, Deduplication, Auditing, Recovery, EDICE, Artificial bee colony, Particle swarm optimization and Recombination procedure

1. Introduction

In recent years, cloud computing is becoming more and more popular in both academic and commercial world. Cloud computing offers resources such as computing power, storage, online applications and networking infrastructures shared as services over the Internet [1]. A wide variety of customers such as personal users, developers, IT companies, educational data users and enterprises are attracted by the possibility of reducing their costs by abandoning the traditional Capital Expenditure model in other words buying and maintaining dedicated hardware, and move to the Operational Expenditure model which is using shared cloud infrastructure and pay as much as they have used. An enormous amount of data is produced every day by people all over the world. With increase of affordable smart phones, tablets and other devices equipped with camera and recorder, the volume of this data is growing rapidly. Cloud storage is an essential service of cloud computing, which allows data owners (DOs) to move data from their local computing systems to the cloud. It provides low-cost, scalable, location-independent platform to store and manage users' data. Among the outsourced data, most of them are duplicated [2] and hence, cloud storage service providers can save huge amounts of storage space via deduplication, in which the cloud server maintains only a single copy of the redundant data and assigns a link of the data to all clients that own the same data. Moreover, through client-side cross-user deduplication, cloud servers can save not only storage space but also network bandwidth whereby the client directly checks data duplications to determine whether to transmit data. Therefore, cloud servers can save storage space and network bandwidth.

Despite these advantages, general client-side deduplication is vulnerable to a number of threats, as only a short and fixed identifier (e.g., hash value of the data) replaces the whole data. For example, cloud servers can be used as a content distribution network unintentionally, and malicious clients can launch a poison attack (Target Collison attack), etc. [3, 4]. Thus, cloud servers that provide the client-side cross-user deduplication have to confirm whether the client actually owns the upload requested data [4].

In addition, through various incidents such as cloud data leakages and corruption, the security of outsourced data has become an important issue. Since the data in clouds can be attacked by internal and external attackers, personal privacy or secret information of enterprises can be leaked or corrupted, which can be fatal. Therefore, the cloud server has to ensure confidentiality and integrity of the outsourced data. However, this can destroy the deduplication goal of using resources efficiently.

Firstly, in terms of confidentiality, if the client encrypts the data using conventional encryption systems, deduplication will fail. Since conventional encryption systems encrypt the data using different encryption keys for each user, it outputs different results even though the inputs are the same. To overcome this, convergent encryption was proposed, where the hash value of the data is used the encryption key [5]. With this, the same data becomes the same ciphertext after encryption, and it enables the cloud server to deduplicate while ensuring confidentiality. However, convergent encryption is vulnerable to brute-force attack.

Secondly, in terms of integrity, the cloud server does not intentionally damage the client's data, however outsourced data can be corrupted during the internal process of the cloud by unintentional error. The cloud server may hide the incident to the client to maintain their reputation. Hence, the client can require an audit for the integrity of outsourced data periodically. However, applying conventional integrity check techniques to the cloud system, such as message authentication codes, can create a huge burden to both the client and cloud server as it requires the local data to verify integrity. Recently, in order to achieve the goal of deduplication while ensuring integrity in cloud storage environments, various techniques have been proposed [6, 7]. However, they have heavy computational costs because the schemes are based on public key cryptography, and the client's privacy can potentially be leaked as they support publicly verifiable integrity auditing. This means that, with respect to a file, subsequent clients have to get the first uploader's public key to be used in integrity auditing, so they are still inefficient, and with large variations in efficiency depending on block and file size.

In order to resolve all these above issues, this work propose strong privacy protection with recovery storage scheme based progressive light weight storage and auditing scheme named as progressive, which ensures data confidentiality with brute-force attack resilience and supports integrity auditing based on EDICE with recovery mechanism. The main contribution of the work is as follows:

- Initially, investigate how to fully resist the brute- force dictionary attacks and realize deduplication with strong privacy protection in cloud storage auditing, and propose a concrete scheme satisfying this property.
- Also, the data recovery is done using hybrid PSO and ABC method and denoted as HPSOABC.
- In order to realize deduplication with strong privacy protection, design a novel method to generate the file index, and employ a new EDICE strategy to generate the key for file encryption.
- In the detailed design, the file index is generated with the help of an Agency Server (AS) instead of directly being produced by the hash value of file. The key for file encryption is generated with the file and the file label. The file label is kept by the user secretly. In this way, the privacy of the user's file is protected against the cloud and the AS.
- In order to improve the storage efficiency, the users, who own the same file, are able to generate the same ciphertext and the same authenticators. The proposed scheme effectively achieves data deduplication and authenticator deduplication.
- Furthermore, to reduce the computation burden on the user side, the user only needs to perform progressive lightweight computation to generate data authenticators, verify the integrity of the cloud data, and retrieve his file from the cloud.

The remainder of this paper is organized as follows: In Section 2, briefly review related work. In Section 3, presents the system construction of proposed SPRLDA. In Section 4, present the implementation of the scheme and compare with other schemes. Finally, in Section 5, conclude the paper with future work.

2. Related work

With the growing popularity of cloud computing, more and more enterprises and individuals tend to store their sensitive data on the cloud in order to reduce the cost of data management. However, new security and privacy challenges arise when the data stored in the cloud due to the loss of data control by the data owner. This section focuses on the techniques of verifiable data storage and secure data deduplication. In [8] proposed a public auditing scheme for cloud storage systems, in which deduplication of encrypted data and data integrity checking can be achieved within the same framework. The cloud server can correctly check the ownership for new owners and the auditor can correctly check the integrity of deduplicated data. However, these schemes either only support plaintext data file or have been proved insecure. In [9] studied the problem of integrity auditing and secure deduplication on cloud data. Specifically, aiming at achieving both data integrity and deduplication in cloud, propose two secure systems, namely SecCloud and SecCloud$^+$. SecCloud introduces an auditing entity with a maintenance of a MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud. In [10] proposed an integrity verification algorithm of different version files. The algorithm establishes the generic storage model of different version control methods to improve the universality of data verification. Then, the methods of verification tags and proofs generating are improved based on the index pointers corresponding to the storage relationship in the version groups and chained keys. Finally, the random diffusion extraction based on the random data sampling in the version group is proposed to improve the verification efficiency. However, this deduplication storage only focuses on file/chunk level to store one copy of the same data hosted by different data owners, and is not concerned with the same part of different data, e.g., a series of version files. In [11] an efficient auditing scheme is proposed for outsourced big data based on algebraic signatures and an XOR-homomorphic function, that can achieve numerous advantages, such as fewer challenges and proofs, non-block verification, data privacy preservation, and lower computational and communication costs. However this work not considered the deduplication mechanism. In [12] proposed an identity-protected secure auditing and deduplicating data scheme in this paper. In the proposed scheme, the cloud cannot learn any useful information on the relationship of data owners. Moreover, the auditing computation cost is high. In [13] improved the security and data storage with Self-Destruction mechanism by handling the applications independently built on reinforcement learning strategy. Data chunk fingerprint index and sketch index are included to support independent and parallel data destruction among multiple applications. Additionally, SePeCloud is further extended to prevent impersonation attacks from illegitimate users by introducing Modified Shamir secrete sharing scheme to handle user revocation policies with limited storage space. There is a need of developing paralleled SePeCloud systems specially for handling big data networks. However, the generation of massive data has brought great burden to users' storage and computing. In [14] proposed fuzzy cloud auditing protocol is such a mechanism running between data owners and cloud storage service providers. In these protocols, the data owner fuzzy challenges the cloud storage servers on the randomly chosen data blocks with random values, the servers need to response with corrected aggregated tag proof to pass through the auditing process. In [15] designed a combined technique, which performs both secure deduplication of encrypted data and public integrity auditing of data. To support the two functions, the proposed scheme performs challenge-response protocols using the BLS signature-based homomorphic linear authenticator. Here utilize a third party auditor for performing public audit, in order to help low-powered clients. However, if the original data holder helps an adversary to obtain the authentication tag, it is easy to mount the attack. In [16] proposed an identity-based public auditing (IBPA) scheme for cloud storage systems. In IBPA, the nonces in a blockchain are employed to construct unpredictable and easily verified challenge messages, thereby preventing the forging of auditing results by malicious TPAs to deceive users. However, certain TPAs might deviate from the public auditing protocol and/or collude with the cloud servers. In [17] proposed a confidentiality-preserving deduplication cloud storage with public cloud auditing (CPDA), which achieves secure file deduplication on encrypted file, which supports public integrity auditing for the unique copy in the deduplication cloud storage system also realizes secure authentication tag deduplication. In [18] proposed the first deduplicated data integrity auditing scheme supporting the ownership modification. It guarantees the integrity of the outsourced data and supports the dynamic access control over the outsourced data. However, this can destroy the deduplication goal of using resources efficiently. Grounded on these research findings a need for efficient and novel TPA which schedules and audits the files on the remote servers with reduced communication, computation and storage cost is evident. So this research propose an progressive light-weight novel and secure auditing protocol based on EDICE cryptographic primitives and optimization based recovery mechanism that ensure reduced communication and computation cost.

3. Proposed Methodology

Cloud storage is an essential service of cloud computing, which allows data owners (DOs) to move data from their local computing systems to the cloud. Moreover in cloud storage system, none of the sides, namely DO and cloud storage provider, is guaranteed to provide unbiased auditing result. Therefore third-party auditing is an intuitive choice for cloud storage auditing to save computation and communication cost. Third-party auditing provides a clear yet cost effective method for setting up the trust between DO and cloud server. In order to execute continuous integrity verification without the DOs local copies of the data and propose an efficient and novel auditing method for securing data in cloud storage based on EDICE algorithm. Here the recovery mechanism is done using HPSOABC. This method allows the auditor to check for the data availability and correctness in cloud storage, with minimum computational cost on the auditor and server when compared with the state of art data auditing methods. Furthermore this work extend the auditing method by utilizing a data structure that permits the auditor to execute dynamic operation like insert, delete, append and modify efficiently with fewer computational cost on DO and cloud storage server.

3.1. SPPRLDA System Model

The proposed system model Fig. 1 consists of entities such as DOs, cloud storage providers (CSP), TPA, and authorized users (AUs).

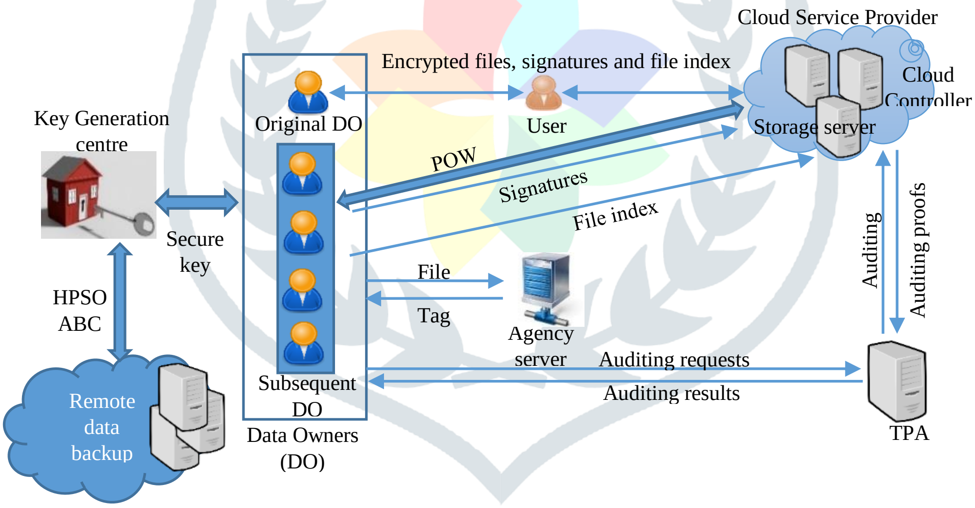

Fig.1.SPPRLDA System Model

Data owner (DO): It is a resource bounded entity which generates gargantuan amount of data to be outsourced to remote cloud storage. They depend on cloud storage providers for data maintenance and computation thus minimizing the storage cost and utilizing the computational resources of the service providers. The DOs are allowed to insert, modify, append or delete their data if required. DOs can be either individual customer or commercial organization.

Cloud service provider (CSP): This service provider, which has unlimited storage space and highly capable computational resources. It provides the storage space to DOs and is responsible for managing and maintaining the servers and data. CSP is an untrusted entity of hardware and software resource cluster.

Third party auditor (TPA): It is a trusted and reliable entity, with the expertise, capabilities and resources to perform data integrity auditing service on outsourced data under the deputation of DO. It is trusted by both the DO and CSP. TPA eases the burden of DO for security upkeep. Metadata manager of TPA is responsible for collection of metadata from the DO and indexes the basic information about the data in the index table. This enables the navigation and search process of a specific data to be at ease.

Authorized user (AU) users: AU possibly customers or coworkers of DO, who have been authorized to access stored data. They can be either inside or outside the cloud.

Cloud controller (CC): the CC checks for deduplication entries and moves the non-duplicated data from the DO to the cloud storage servers. The index for the deduplication data is updated with modification in the address pointers and the storage index table is generated for the data stored by the storage manager. The query generator in CC is responsible for generating query based on the scheduling policy in Table 1 and the proof as the response for the request is generated and send to the requester. The verifier in TPA checks if the received proof is valid or not and if any misbehaviour found will raise the audit report to the DO for further action. Initially the client uploads the data in the required file format and using random key generator the client generates the key to verify the data. **The key generation centre:** It generates the keys using pseudo random generator function PRG(.). The file to upload is split into smaller units as blocks $b_1, b_2, \ldots, b_n$. This EDICE encryption enables privacy of the user's data blocks and also enables efficient deduplication. After the entire data blocks are encrypted the tag for each data block is generated which will be used for integrity verification. The client sends the data to the CSP to store it in their server premises. The CSP controller receives the data and checks for the deduplication data entries if any. If there exists any duplicate entry of the blocks then discard the duplicated blocks and stores the unique data blocks. The duplicated blocks will have a pointer to the original data blocks storage address.

Agency Server: It is the storage manager generates index for each files with file id, rule exact location address for the blocks of the file document to be stored in the storage the address for the set of documents stored in the exact storage and storage nodes does the actual storage and this file server and the server checks for deduplication entry and store the data in the cloud. The tag generated will be send to the auditor to verify the data tag later with the server. Now the server receives the data from the client and stores it in the cloud into the table in the encrypted form. When the auditor asks for any proof from the server it generates the tag at that instant and send it to the auditor to verify. The auditor receives the tag which was generated by the client and stores it in the table in the auditor database. The auditor's scheduler ensures continuous auditing at periodic intervals to ensure the integrity of the data. The auditor sends the challenge to the server at several intervals of time and the server generates the proof and send it back to the auditor. The challenge is gathered from multiple clients and then processed in batches by the auditor to the server. Then the auditor checks for the integrity from the proof it received from the server and if the integrity is maintained then the auditor informs to the client that the data is secure. If there is any misbehaviour detected or corrupted block is identified the auditor will inform the storage client about it and also reports to the maintenance and monitoring manager for preventive measures to be enforced and SLA's to be updated to further prevent such inconsistencies and also to ensure prevention mechanism enabling the users ease of maintenance of data with high security in a remote storage cloud environment.

Remote data backup: The main objective of this proposed optimization model is using HPSOABC for efficient Disaster Recovery in a cloud environment, that shorten the data recovery time and data backup cost as much as possible.

Threat model: The security goal in the proposed scheme is to fully resist the brute-force dictionary attacks and provide strong privacy protection for the users' files. Here consider two types of adversaries: internal adversary and external adversary. The malicious cloud or the AS plays the role of the internal adversary, which is honest-but-curious. That is to say, the cloud performs the deduplication protocol honestly but tries to cheat the user about the data corrupt event or derive the file's content from the encrypted file. The AS faithfully performs the assigned operations, but intents to guess the contents of users' files and do not consider the collusion of the cloud and the AS. The user, who somehow knows the file's content or the file's hash value, acts as the external adversary. He intends to obtain a link of a file but does not keep the corresponding file.

3.2. Data Recovery Process using HPSOABC method

The cloud data backup and recovery using the HPSOABC has made the process more powerful and efficient. The data are considered as solutions and those solutions are taken as an initial particle. Fitness value is calculated by using the following equation (1)

$$F = \min\{\max p_k\} + \sum f(D)$$

where p_k represents the kthe node particle and D represents that distance between particle. In cloud, data are stored continuously on a Virtual Machine (VM) location. Based on the fitness value of the data, the retaining of data is decided. First tasks are scheduled and related resources are allocated to the data which is stored in the cloud. So that the data from the cloud can be accessed at any time. During the data recovery procedure, the following steps describe the execution flow.

1. A data recovery request is the initial step from the user to the data disaster recovery service provider CSP. This request is transmitted via cloud interface.

2. The second step is an immediate authentication process once the request reaches the CSP. It authenticates the user's details like privilege and account; and neglect the request once it is found to be not the legal one.

3. In order to find the backup locations of the data, every CSP has its own recovery manager that found the data in the Meta database.

4. If the CSP find the data in the database in three different locations, then it compares all the location and selects the one with highest recovery bandwidth.

5. Consider the selected location be CSP_x, and the recovery manager of CSP_x alerts the CSP to made the temporary access authorization which can be used by the recovery proxy of the user who send the request in begin to read the replica. Once the authorization started, the CSP_x informs CSP.

6. The recovery proxy of the user who gave request receives the authorization information and data location from the CSP and the data from the CSP_x is taken from its recovery proxy and intimates CSP.

7. Once the data is retrieved from the CSP_x, it abolish the temporary access authorization which was crested before according to the instruction of CSP_x. Then it records the information regarding the user request for later use.

8. In parallel, the CSP_x analyze and monitor the traffic consumption of the request to charge the CSP when needed.

3.2. HPSOABC based Data Recovery Mechanism

Upon analysing the ABC [19] algorithm, it is seen that basic ABC does not directly use the global best solution of the population. Also, when the particles in the PSO [20] get stuck in the local minima, there is not an ability which will get rid of local minimum in the basic PSO models. To cope with these disadvantages of the algorithms, propose a hybrid global optimization approach based on recombination procedure. Recombination or crossover operation is known as evolution operator. Here use it for generating a new solution called as TheBest. TheBest is taken as gbest for the PSO and neighbor of onlooker bees for the ABC. In order to obtain TheBest, fitness values of gbest of the PSO and the best solution of the ABC are calculated by using Eq. (1), and selection probabilities for two solutions are calculated by using these fitness values and Eqs. (2) and (3).

$$P_{ABCbest} = \frac{F_{ABCbest}}{F_{gbest} + F_{ABCbest}}$$

where $P_{ABCbest}$ is the selection probability of the best solution of the ABC, F_{gbest} and $F_{ABCbest}$ are the fitness values of the gbest of the PSO and the best solution of the ABC obtained by using Eq. (1).

$$P_{gbest} = \frac{F_{gbest}}{F_{gbest} + F_{ABCbest}}$$

Eher P_{gbest} is the gbest of the PSO. While the best is constructed, random numbers in the range of [0,1] are used for the dimensions of the problem. If the random number is less than or equals to $P_{ABCbest}$, the value for this dimension is taken from the best solution of the ABC; otherwise, the value for this dimension is taken from gbest of the PSO. This is formulated as follows:

$$Optimum_i = \begin{cases} ABCbest_i & if\,(r < P_{ABCbest}) \\ gbest_i & otherwise \end{cases}$$

where $Optimum_i$ is the ith dimension of final best values, $ABCbest_i$ is ith dimension of the best solution obtained by the employed bee population of the ABC, $gbest_i$ is the ith dimension of gbest of the PSO. By using the recombination procedure, the relationship is provided between the particles and bees in the HPSOABC. The social component of the HPSOABC is "optimum". By using the optimum, not only the exploitation ability of the ABC is improved through the direct use of global best information, but also the ability of the PSO to get rid of local minima (search ability) is improved. Gbest of the PSO is changed with the best and the best is given to onlooker bees of ABC as neighbor. The flowchart of the HPSOABC is displayed in Fig. 2. The HPSOABC is also an iterative method like the PSO and ABC, and its termination condition can be the maximum evolution number, maximum running time etc.

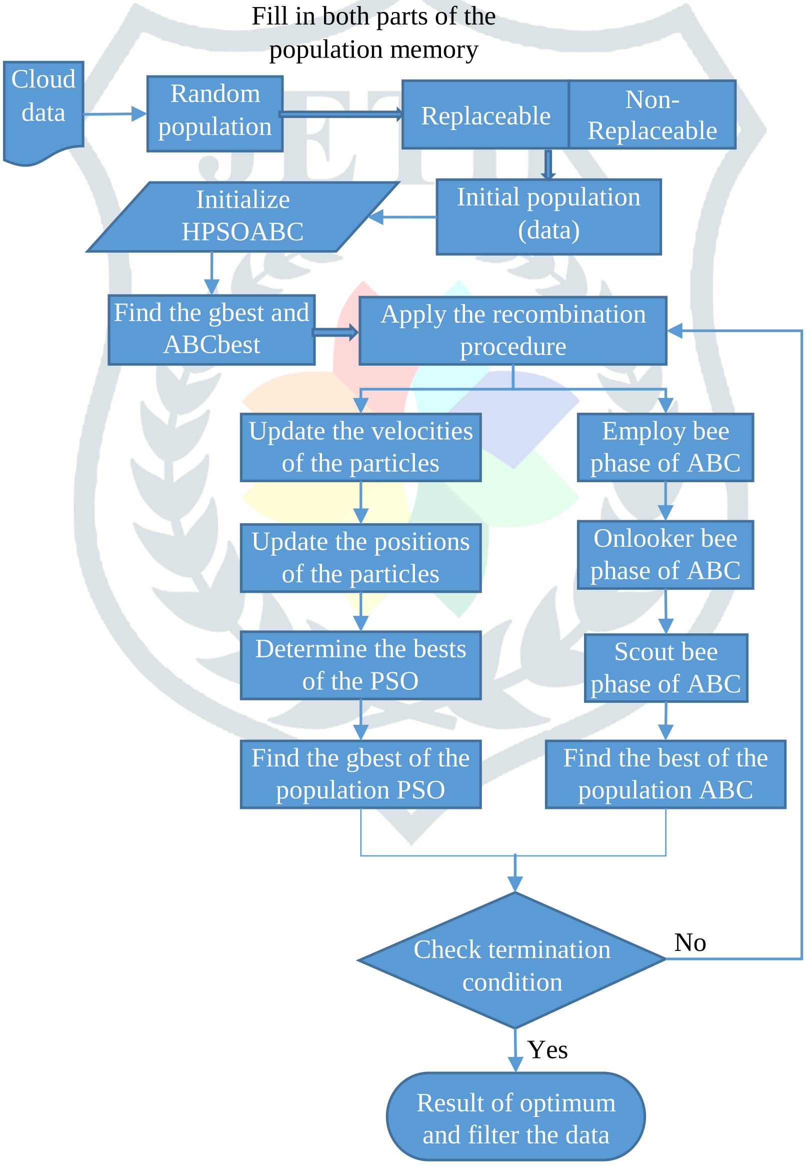

Fig.2.Flowchart of the HPSOABC

3.3. EDICE based Key Generation Method

This section introduces a new client-side deduplication protocol, called EDICE (Enriched Dual Integrity Convergent Encryption), in which perform tag checking and send the hash instead of the entire message over the network, resulting in a reduction of communication and computation cost without any loss of security. EDICE allows cloud storage services to deduplicate data and provides better privacy to traditional cloud storage. In EDICE, the hash value of f is sent to security proxy first. Then, the security proxy will generate a new key $K = G(h, offset)$ with hash h and the offset, and send k to client. The eventual ciphertext will be $C = E_{G(hash(f), offset(f))}$ at client. In the way, the attacker cannot calculate the key according to the plaintext, because the key is decided by the hash value together with the offset generated by security proxy. Attack cannot be carried out when plaintext and key are unknown, that is to say, our improved convergent can resist the content guessing attack well. Upon receiving C, the server (S) then generates the tag T based on the hash of the cipher text as T ← H(C). The server then matches T with stored tags and if there is a match, the metadata corresponding to the client is updated and the client is authorized to access C. Otherwise, the server stores both C and the tag T. In brief, the server keeps a log of the duplicated file through the tag T. Later on, whenever the client requests the cipher text, the server sends the tag T and the client stores T. For the download operation, the client first sends the stored tag T along with his identification information (ID) for the download. The server sends the cipher text C corresponding to T and the client is able to get back the original plain text M as $M \leftarrow D(K, C)$, where D is the decryption scheme. Upon receiving the tag T, the server first checks if there is a match with the stored tag T 0 . If there is a match found, it implies that the cipher text is already present at the cloud server. During the entire protocol, the client stores only K and the tag T, whereas the server stores T′ and C. Compared to the CE protocol, our protocol sends only the tag T. Hence, the bandwidth requirement is reduced.

3.4. Proposed SPRPLDA Model

A progressive lightweight cloud storage auditing with deduplication supporting strong privacy protection is composed by the following seven algorithms:

1) Setup algorithm. It takes as input a security parameter γ, and outputs the AS's private-public key pair (x, y) and the system public parameters $pubp$.

2) Initial Processing algorithm. It takes as input the AS's private-public key pair (x, y) and the file F, and outputs the file index α_1 and the file label α_2. The initial user computes the EDICE encryption key K_{enc} with the file label α_2 and the file F, then generates the ciphertext CF with the key K_{enc}.

3) Authenticator Generation algorithm. It takes as input the file index α_1, the file label α_2, the file identifier name, the file F and the Pseudo Random Function (PRF) key k_{prf} . It generates the initial user's private key sk and a file tag TF. The initial user generates an authenticator set φ for the ciphertext CF with the PRF key k_{prf}.

4) Subsequent Processing algorithm. It takes as input the AS's private-public key pair (x, y) and the file F, and generates α_1 and α_2. The subsequent user generates the encryption key K_{enc} with the file label α_2 and the file F, and outputs the ciphertext CF with the key K_{enc}. The subsequent user computes his private key sk according to the α_2, the file F and the file identifier name. Furthermore, the subsequent user executes the PoW protocol with the cloud to prove that he indeed owns the file with TPA.

5) Proof Generation algorithm. It takes as input the ciphertext CF, the corresponding authenticator set φ and the auditing challenge AC, and generates an auditing proof AP that is used to prove the cloud stores the entire ciphertext CF. The TPA first computes the hash values for all the challenged data blocks using the EDICE and computes the root challenge hash and then verifies the response from the server for matches of the hash of the blocks challenged.

6) Proof Verification algorithm. It takes as input the auditing challenge AC and the auditing proof AP, and returns ''true'' if the proof is valid; or ''false'', otherwise.

7) Data Retrieval algorithm. It takes as input the ciphertext CF, its corresponding authenticators φ and the file tag TF. The user verifies the validity of file tag TF, then checks the integrity of the ciphertext CF. If the ciphertext CF is intact, the user decrypts the ciphertext CF, then recovers the file F. The overall process is given in Fig.3.

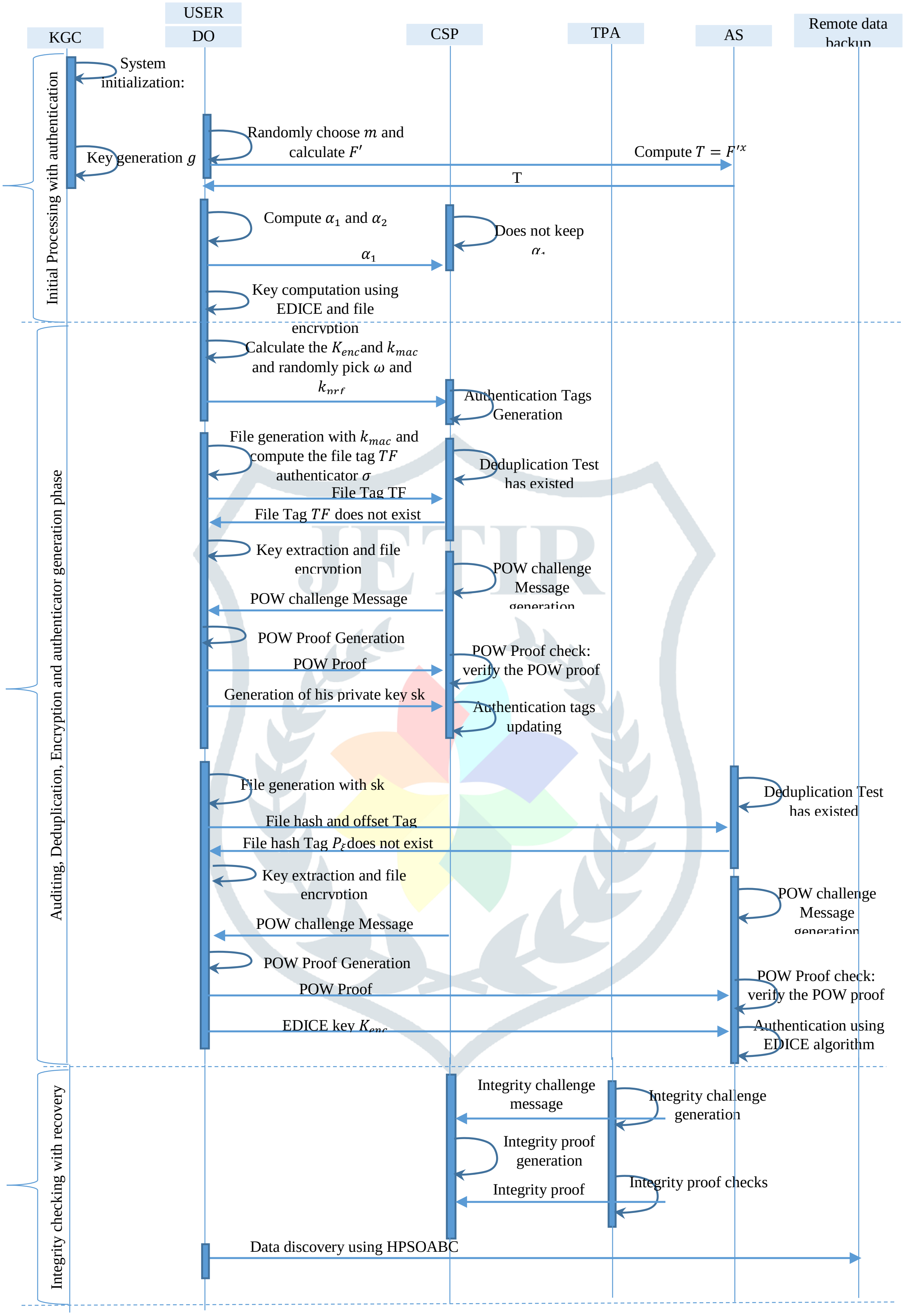

Fig.3. The overall process of SPRPLDA

The Proposed Scheme: The details of the proposed scheme are as follows.

1) Setup algorithm: Let CG_1 be a multiplicative cyclic group of the order p which is one large prime. Let g and u be two random generators of the group CG_1. Pick four EDICE based cryptographic hash functions $H1 : \{0,1\}^* \rightarrow CG_1, H2 : CG_1 \rightarrow \{0,1\}^{\lambda}$, $h_1: \{0,1\}^* \rightarrow \{0,1\}^{\lambda}$ and $h2: \{0,1\}^* \rightarrow Z_p^*$, where λ is a security parameter and Z_{pubp}^* is te prime field with nonzero elements. After received the hash values, CSP generates a random offset, and re-calculates the encryption key as $k = G(h, offset)$, the send back to client. Select a pseudo-random function : $\{0,1\} * \times Kprf \rightarrow Z_{pubp}^*$. AS selects a random value $x \in Z_{pubp}^*$ as his private key, and calculates $y = g^x$ as his public key. The system public parameters are $pubp = (CG_1, pubp, g, u, H1, H2, h1, h2, y, f, offest)$.

2) Initial Processing algorithm: The initial user interacts with AS to generate the file index α_1 and the file label α_2. The $\alpha_1$is used to verify duplicate file by the cloud. The $\alpha_2$is used to output some keys for encryption and authenticator generation. Then the initial user sends the file index α_1 to the cloud as the file upload request. If the cloud did not store this file index α_1, then the initial user encrypts his file F with the EDICE encryption key K_{enc}.

a) The initial user chooses $m \in Z_{pubp}^*$ at random, then computes the blinded hash value $F' = H_1(F)g^m$ and sends it to AS.

b) Upon receiving $F'$from the initial user, AS computes $T = F'^x$ with his private key x, and sends it to the initial user.

c) The initial user calculates $\tau = Ty^{-m}$ with AS's public key y, then computes $\alpha_1 = H2(\tau||1)$ and $\alpha_2 = H2(\tau||2)$. Set α_1 as the file index, which is used to check duplicate file by the cloud. Set α_2 as the file label, which is used to generate the encryption keys such as K_{enc} and MAC key K_{mac}.

d) The initial user sends the file index α_1 to the cloud as the file upload request. If the cloud did not keep α_1, the initial user computes the encryption key $K_{enc} = h_1(\alpha_2||F)$, then encrypts the file F as follows: $CF = Enc(F, K_{enc})$. The ciphertext CF is divided into n blocks $(c_1, c_2, \ldots, c_n)$, where $c_i \in Z_{pubp}^*$ denotes the i-th block of the ciphertext CF.

3) Authenticator Generation algorithm: The initial user generates his private key $sk = (K_{enc}, k_{mac})$, the file tag TF, and the authenticator set φ. The initial user uploads the ciphertext CF, the set of authenticators φ and the file tag TF to the cloud. The cloud makes a link to the file F for the initial user. The initial user holds the symmetric encryption key sk_{enc} and his private key $sk = (k_{enc}, k_{mac})$. The symmetric encryption key skenc is used to decrypt the ciphertext in Data

3) Retrieval algorithm: The private key $sk = (K_{enc}, K_{mac}.)$ is used to generate the file tag TF and encrypt the PRF key k_{prf} and the random value ω in the Authenticator Generation algorithm, and verify the validity of file tag TF, decrypt the encrypted portion and recover the PRF key kprf and the random value ω in the Proof Verification algorithm. This process is illustrated in Fig.2.

a) The initial user calculates the symmetric encryption key $K_{enc} = h1(name||\alpha_2||F||1)$ and the MAC key $k_{mac} = h1(name||\alpha_2||F||2)$, where name is the identifier of the file F. The initial user's private key is $sk = (K_{enc}, k_{mac})$. The initial user randomly picks $\omega \in Z_{pubp}^*$ and a PRF key $k_{prf} \in Z_{pubp}^*$ for the pseudo-random function prf.

b) The initial user computes the file tag $TF = T_0||MAC\_k_{mac}(T_0)$, where $T_0 = n\left||Enc(k_{prf})\right||\omega, K_{enc}$. For each block $c_i \in Z_{pubp}^*(i \in [1,n])$ of the ciphertext CF, the initial user computes the authenticator σ_i for block c_i as follows: $\sigma_i = f_{k_{prf}}(i) + \omega c_i$. Denote the set of authenticators as $\varepsilon_0 = \{\sigma_i\}_{1 \le i \le n}$. The initial user uploads $\{CF, \varphi\}$ along with the file tag TF to the cloud, then deletes these messages from local storage and holds the symmetric encryption key K_{enc} and his private key $sk = (k_{enc}, k_{mac})$. The cloud makes a link to the file F for the initial user.

4) Subsequent Processing algorithm: The subsequent user interacts with AS to generate the file index $\alpha_1$and the file label β2. Then the subsequent user sends the file index α_1 to the cloud as the file upload request. If the cloud has stored this file index α_1, then performs the PoW protocol with the subsequent user. With PoW, the subsequent user is able to prove to the cloud that he indeed keeps the file F without sending the entire file. The subsequent user generates his private key, and holds the symmetric encryption key and his private key. This process is illustrated in Fig.3.

a) The subsequent user chooses $m' \in Z_{pubp}^*$ at random, then computes the blinded hash value $F'' = H1(F)g^{m'}$ 0 and sends it to AS.

b) After receiving F'' from the subsequent user, AS computes $T' = F''^x$with his private key x, and sends it to the subsequent user.

c) The subsequent user calculates $\alpha = T'y^{-m'}$ with AS's public key y, then computes the file index $\alpha_1 = H2(\alpha||1)$ and the file label $\alpha_1 = H2(\alpha||1)$. The subsequent user sends the file index α_1 to the cloud as the file upload request. If the cloud has stored α_1, then he will perform the following PoW protocol with the subsequent user.

i) The cloud randomly picks a set of I with c elements, where $I \subseteq [1,n]$. For each $i \in I$, the cloud outputs a random value $v_i \in Z_{pubp}^*$, and sends a PoW challenge $PoWC = \{v_i\}i \in I$ to the subsequent user.

ii) The subsequent user computes the symmetric encryption key $K_{enc} = h1(\alpha_2||F)$, and encrypts the file F as follows: $CF = Enc(F, K_{enc})$. Then the subsequent user calculates the PoW proof $PoWp = \sum_{i \in I} v_i c_i$, and sends it to the cloud.

iii) Upon receiving the PoW proof, $p = \sum_{i \in I} v_i c_i$ then verifies whether the following equation holds or not: $PoWp = p$. If the equation holds, the cloud believes that the subsequent user indeed owns the file F, and provides the subsequent user with a link to the file F. It means that the subsequent user does not need to upload the ciphertext CF and its corresponding authenticators to the cloud any more.

d) The subsequent user generates his private key $sk = (K_{enc}, kmac)$ with the file label α_2, the file F and the file identifier name, where K_{enc} and kmac. The subsequent user holds the symmetric encryption key skenc and his private key sk = (K_{enc}, kmac).

5) Proof Generation algorithm: The user outputs and sends an auditing challenge to the cloud. Then the cloud generates an auditing proof to respond to the user.

a) The user (the initial user or the subsequent user) randomly selects a c-elements subset $I(I \subseteq [1,n])$. For each $i \in I$, outputs a random value $v_i \in Z_{pubp}^*$. Then, the user generates and submits an auditing challenge $AC = \{i, v_i\}_{i \in I}$ to the cloud.

b) Upon receiving the auditing challenge from the user, the cloud computes a linear combination of encrypted data blocks $\mu = \sum_{i \in I} v_i c_i$ and an aggregated authenticator $\sigma = \sum_{i \in I} v_i c_i$. The cloud sends the auditing proof $AP = \{\mu, \sigma\}$ along with the file tag TF to the user.

6) Proof Verification algorithm: The user firstly utilizes his private key k_{mac} to verify the validity of MAC on file tag TF. If the MAC is valid, the user parses T, then decrypts the encrypted portion $Enc(k_{prf} || \omega, K_{enc})$ under his private key $K_{enc}$and recovers the k_{prf} and the random value ω. Finally, the user checks whether the following verification equation holds or not, $\sigma = \sum_{i \in I} v_i c_i f_{kprf}(i) + \omega\mu$. If the verification equation holds, it means that the cloud indeed keeps the user's intact data.

7) Integrity verification: The TPA randomly picks a c-element subset $I(I \subseteq [1, n])$. For each $i \in I$, outputs a random value $v_i \in Z_{pubp}^*$. Then user calculates a random masking based on the DO's public key as $R \leftarrow y_t^r$, where $r \in Z_{pubp}^*$ is a random number, and sends the challenge message $CM = (\{i, v_i\}_{i \in I}, R)$ to the CSP. Upon receivingMC, the CSP would produce a response of data integrity, which consists of the tag proof Θ and the data proof. The tag proof 2 is essentially the aggregated authenticator of authentication tags of the challenged blocks, i.e.,

$$\Theta = \prod_{i \in I} \theta_i^{w_i}$$

Then, the CSP computes the linear combination of sampled blocks and aggregates all DOs' public keys except the DO's public key as follows:

$$C = \sum_{i \in I} v_i C_i \,, Y = \prod_{j \in [0,v] \setminus I} y_j$$

The data proofs are composed of ω_1 and ω_2, where $\omega_1 = e(u, R)^C, \omega_2 = e(u, Y)^C$. Finally, the CSP sends $P = (\Theta, \omega_1, \omega_2)$to the TPA as the integrity proof. Further, the TPA performs the integrity verification by checking

$$\omega_1 \cdot \omega_2^r \cdot e\left(\prod_{i \in I} H(T_F || i)^{w_i}, R \cdot Y^r\right) = \Theta^r$$

If Θ^r holds, it outputs Ture; otherwise, outputs False.

8) Data Retrieval algorithm: The user, who wants to access his file, will send a request to the cloud. The cloud returns the ciphertext CF, its corresponding authenticators φ and the file tag TF to the user. After receiving these messages, the user firstly verifies the integrity of the ciphertext CF, then recovers his file F with the EDICE encryption key K_{enc}.

a) The user, who wants to use his file, submits a request for the ciphertext CF to the cloud.

b) Upon receiving the request from the user, the cloud firstly verifies whether the user is the data owner of the ciphertext CF. If he is, the cloud sends the ciphertext CF, its corresponding authenticators φ and the file tag TF to the user. Otherwise, the cloud rejects the user's request.

c) After receiving the messages from the cloud, the user firstly verifies the validity of MAC on file tag TF with his private key kmac. If the MAC is valid, the user parses TF, then decrypts the encrypted portion $Enc(kprf || \omega, K_{enc})$ by using his private key K_{enc} and recovers the k_{prf} and the random value ω. The user checks whether the following verification equation holds or not.

$$\sum_{i \in I} \sigma_i = \sum_{i \in [1,n]} f_{k_{prf}}(i) + \sum_{i \in [1,n]} c_i$$

If the equation holds, the user believes the ciphertext CF stored in the cloud is intact, then utilizes the EDICE encryption key K_{enc} to decrypt the ciphertext CF , and recovers the file F: $F = Dec(CF, K_{enc})$.

Security Analysis: The SPRPLDA ensures confidentiality with a brute-force attack resilience if any adversary is not allowed to compromise with the management server. In this system, a management server generates the improved convergent key associated with a private key of the management server and assume that the key distribution process is run over a secure channel. Thus, no adversary who attempts to launch a brute force attack can generate the valid encryption key without the private key of the management server. Therefore, even if the file is predictable, the adversary cannot guess the plaintext by launching a brute force attack. However, the adversary can attempt a brute force attack via the management server as the management server cannot make a distinction between an honest and malicious client. Hence, by applying SPRPLDA can achieve confidentiality with a brute force attack resilience.

4. Experimental Results and Discussions

Numerically analyses and simulation experiments on the communication, computation and storage costs of our SPRPLDA scheme are conducted in this section. Then, give the comprehensive performance comparisons between our SPRPLDA scheme with the current related schemes PCAD [22], P-PCAD [23] and CTAPDA all of which realize deduplication with the public cloud auditing. For the sake of fairness of comparison, assume that the size of original file F in all schemes is equal, the communication costs of uploading F and its identifier and the storage costs of storing F are not considered for all schemes in the numerical and experimental analyses. The criteria considered for the evaluation of the LDAP system are communication cost, computation cost and storage cost. For calculating the computation cost, the time taken for creating the hash tables and time for verifying the metadata are considered. For communication cost, the time taken to transfer the data between CSP and TPA and also for the TPA to send the result to the DO will be measured. Data storage auditing is extensively a resource demanding service in terms of communication cost, computational resource, and storage space. In this subsection briefly analyze these performance metrics. In this experiments, set the base field size to be 512 bits, the size of an element to be |p| = 160 bits, the size of a data file to be 20MB composed by 1,000,000 blocks.

4.1. Communication Cost Comparison

This section compare SPRPLDA scheme with existing scheme PCAD, P-PCAD and CTAPDA about the communication costs in storage phase and auditing phase, and the results are shown in Fig. 3. In the PCAD scheme, each subsequent owner does not need to upload the authentication tags, because this scheme audits the integrity of file F by using the authentication tags uploaded by the original owner. However, the original owner needs heavy communication costs in storage phase and it is not able to ensure that each owner can audit the integrity of file F separately. The P-PCAD scheme can ensure that each owner audits the integrity of file F separately, the CSP needs to store all authentication tags from all owners of file F, which results in heavy storage overheads for the CSP. In SPRPLDA the communication costs during the auditing phase is minimal, however, the SPRPLDA scheme requires a smaller proof with constant size than the PCAD, P-PCAD and CTAPDA scheme. To further reduce the computation cost this method uses more of the hash operations using EDICE for the data location and deduplication checking. From the results it is obvious that, the SPRPLDA scheme is comprehensively more efficient in communication costs during storage and auditing phases.

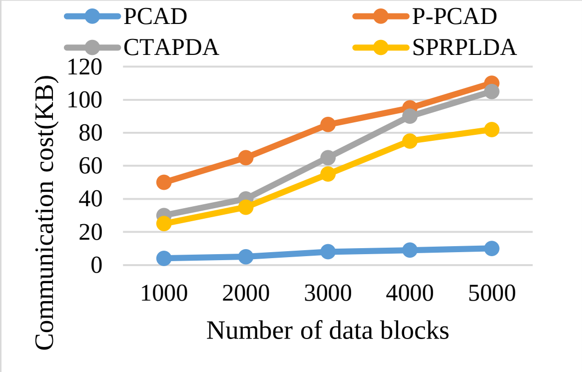

Fig.5.Comparisons of communication costs

4.2. Computation Cost

Now, the SPRPLDA scheme with the existing schemes such as PCAD, P-PCAD and CTAPDA, show them in Fig. 6. Without loss generality, the computation complexity only considers the multiplication operation, exponentiation operation and bilinear pairing operation on multiplicative group to describe computation costs simply and effectively. However, the PCAD scheme needs more computation costs in storage phase and generating the proof than the P-PCAD. In many other auditing schemes such as PCAD, P-PCAD and CTAPDA the dynamic auditing are not supported due to computation cost and difficulty in performing audit operation. This proposed method will solve these two problems for better performance of the cloud system and to improve users trust. This method reduces the computation cost at the user side by simply outsourcing the audit process to a TPA. To sum up, can get a conclusion that the SPRPLDA scheme is relatively computationally efficient during the storage phase and auditing phase.

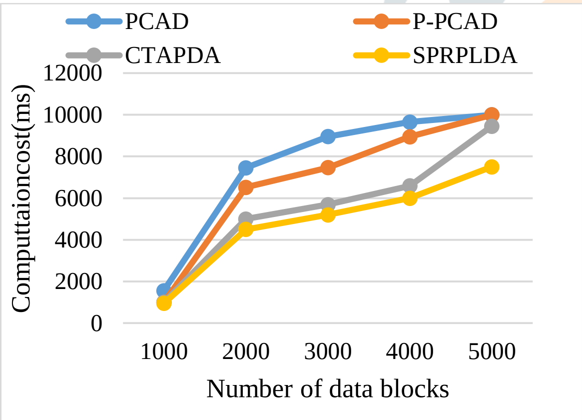

Fig.6.Computation Cost Comparison

4.3. Execution Time

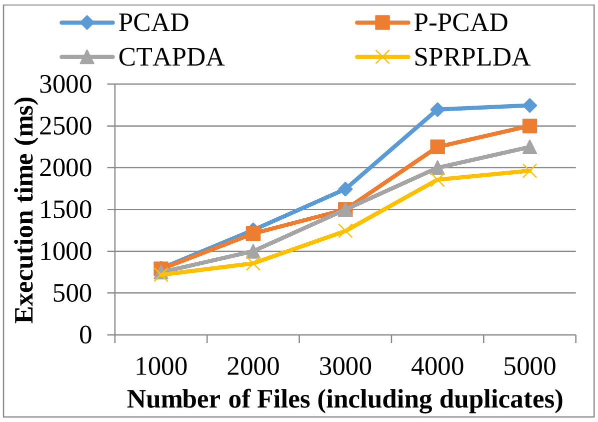

Fig.5. Energy consumption Vs No. of Nodes

Fig.5 illustrates the relationship between the execution time on communications and the number of files. It can be said that the proposed SPRPLDA approach consumes less time when compared with the other PCAD, P-PCAD and CTAPDA approach. From the figure, the time value will reduce significantly when malicious files initiate attacks in cloud. SPRPLDA can reduce the execution time value compared with PCAD, P-PCAD and CTPDA, since it secure against the poison attack and brute force and performs data deduplication faster EDICE. Attack cannot be carried out when plaintext and key are unknown, that is to say, this EDICE can resist the content guessing attack well and also performs data deduplication faster than the other schemes. Simultaneously, SPRPLDA reduce the communication bandwidth and the computation time required.

5. Conclusion and Future Work

In this work, proposed SPRPLDA to solve the problem of user's privacy leakage in cloud storage auditing with deduplication when brute-force dictionary attacks are launched. This SPRPLDA is a progrssive lightweight cloud storage auditing scheme with deduplication supporting strong privacy protection. In the proposed scheme, the privacy of user can be well preserved against the cloud and other parties. The user relieves the heavy computation burden for generating data authenticators and verifying data integrity. The security proof shows that the proposed scheme is secure. Also provide detailed comparisons among our proposed scheme and other existing schemes by experiments. Experimental results show the proposed scheme achieves higher storage efficiency and is more efficient in authenticator generation phase and auditing phase. In the future, plan to extend this protocol by testing it against other vulnerabilities beyond the poison attack, such as the identification, side-channel and dictionary attack.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)
2. J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," in Proceedings of the 1st IEEE International Conference on Communications and Network Security (CNS '13), October 2013.
3. D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," IEEE Security & Privacy, vol. 8, no. 6, pp. 40–47, 2010.
4. S. Halevi et al., "Proofs of ownership in remote storage systems," in Proceedings of the 18th ACM conference on Computer and communications security, ACM, 2011.
5. Jayapandian, N., & Md Zubair Rahman, A. M. J. (2018). Secure deduplication for cloud storage using interactive message-locked encryption with convergent encryption, to reduce storage space. Brazilian Archives of Biology and Technology, 61.
6. K. He, J. Chen, R. Du, Q. Wu, G. Xue, and X. Zhang, "DeyPoS: deduplicatable dynamic proof of storage for multi-user environments," IEEE Transactions on Computers, vol. 65, no. 12, pp. 3631–3645, 2016.
7. J. Li et al., "Secure auditing and deduplication data in cloud," *IEEE Transactions on Computers and Cloud Computing*, pp. 1–11, 2015.
8. He, K., Huang, C., Zhou, H. *et al.* Public auditing for encrypted data with client-side deduplication in cloud storage. *Wuhan Univ. J. Nat. Sci.* **20,** 291–298 (2015).

9. J. Li, J. Li, D. Xie and Z. Cai, "Secure Auditing and Deduplicating Data in Cloud," in IEEE Transactions on Computers, vol. 65, no. 8, pp. 2386-2396, 1 Aug. 2016.
10. Xu, G., Lai, M., Li, J. *et al.* A generic integrity verification algorithm of version files for cloud deduplication data storage. *EURASIP J. on Info. Security* **2018,** 12 (2018).
11. Gan, Q., Wang, X. & Fang, X. Efficient and secure auditing scheme for outsourced big data with dynamicity in cloud. *Sci. China Inf. Sci.* **61,** 122104 (2018).
12. Hou, H., Yu, J., Zhang, H. *et al.* Enabling secure auditing and deduplicating data without owner-relationship exposure in cloud storage. *Cluster Comput* **21,** 1849–1863 (2018).
13. Savitha, S., Thangam, P. & Latha, L. An enhancement to SePeCloud with improved security and efficient data management. *Cluster Comput* **22,** 3261–3269 (2019).
14. Zhang, J., Wang, B., He, D. *et al.* Improved secure fuzzy auditing protocol for cloud data storage. *Soft Comput* **23,** 3411–3422 (2019).
15. T. Youn, K. Chang, K. Rhee and S. U. Shin, "Efficient Client-Side Deduplication of Encrypted Data With Public Auditing in Cloud Storage," in IEEE Access, vol. 6, pp. 26578-26587, 2018.
16. Xue, J., Xu, C., Zhao, J. *et al.* Identity-based public auditing for cloud storage systems against malicious auditors via blockchain. *Sci. China Inf. Sci.* **62,** 32104 (2019).
17. J. Wu, Y. Li, T. Wang and Y. Ding, "CPDA: A Confidentiality-Preserving Deduplication Cloud Storage With Public Cloud Auditing," in IEEE Access, vol. 7, pp. 160482-160497, 2019.
18. Bai, J., Yu, J. & Gao, X. Secure auditing and deduplication for encrypted cloud data supporting ownership modification. *Soft Comput* (2020).
19. Karaboga, D. (2010). Artificial bee colony algorithm. *scholarpedia*, *5*(3), 6915.
20. Zhu, H. M., & Wu, Y. P. (2010). A PSO algorithm with high speed convergence. *Control and Decision*, *25*(1), 20-24.
21. J. Yuan and S. Yu, ‘‘Secure and constant cost public cloud storage auditing with deduplication,’’ in Proc. IEEE CNS, National Harbor, MD, USA, Oct. 2013, pp. 145–153,
22. C. Li and Z. Liu, ‘‘A secure privacy-preserving cloud auditing scheme with data deduplication,’’ Int. J. Netw. Secur., vol. 21, no. 2, pp. 199–210, Mar. 2019.