IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Special Issue-15, May 2019 E-ISSN: 2347-2693

Integrity Auditing and Data Sharing With Sensitive Information Hiding for Secure Cloud Storage

Divya. U[1*], Nagaveni. S[2], Pooja. S[3], Ramya. R[4], Supritha. N[5]

[1,2,3,4,5]Department of Computer Science and Engineering, EWIT, Visvesvaraya Technological University, Karnataka, India

DOI: https://doi.org/10.26438/ijcse/v7si15.9098 | Available online at: www.ijcseonline.org

***Abstract*—** Along with the development of cloud computing more and more applications are moved to the cloud. With cloud storage services, users can remotely store their data to the cloud and realize the data sharing with others on the condition that the sensitive information is hidden in order to guarantee the integrity and confidentiality of the data stored in the cloud. Remote data integrity auditing is proposed to guarantee the integrity of the data stored in the cloud. In some common cloud storage systems such as the electronic health records system, the cloud file might contain some sensitive information. The sensitive information should not be exposed to others when the cloud file is shared. Encrypting the whole shared file can realize the sensitive information hiding but will make this shared file unable to be used by others. How to realize data sharing with sensitive information hiding in remote data integrity auditing scheme with time seal management still has not been explored In order to address this problem, we propose a remote data integrity auditing scheme with appropriate time management thereby providing limited access that realizes data sharing with sensitive information hiding in this paper. In this scheme, a sanitizer is used to sanitize the data blocks corresponding to the sensitive information of the file and transforms these data blocks' signatures into valid ones for the sanitized file. These signatures are used to verify the integrity of the sanitized file in the phase of integrity auditing. As a result, our scheme makes the file stored in the cloud able to be shared and used by others on the condition that the sensitive information is hidden, while the remote data integrity auditing is still able to be execute efficiently. Meanwhile, the proposed scheme is based on Identity-based cryptography, which simplifies the complicated certificate management and also solves key exposure problem. The security analysis and the performance evaluation show that the proposed scheme is secure and efficient.

***Keywords*—** Cloud storage, key management, Remote data Integrity, Confidentiality, Sensitive Information Hiding**.**

I. INTRODUCTION

Cloud storage, which is one of various cloud services, serves as a practical tool and has made data outsourcing to the cloud an emerging trend. The rapid development of such a cloud service has various causes such as its on-demand outsourcing function, ubiquitous network access, and location independent resources [1]–[6]. For instance, data are no longer local with cloud storage, ensuring that data owners (DOs) do not have to worry about software or hardware failures. In addition, overhead resulting from maintenance, financial cost, time, and other resources would be greatly reduced, relieving the burden on DOs and local devices. However, data outsourced to the cloud are not kept securely and still suffer from a variety of security attacks both internal and external [7]–[12]. On the one hand, malicious network attacks, which are external and familiar to Internet users, threaten cloud data. Hackers might retrieve and steal cloud users' data or even corrupt and delete the data, destroying its confidentiality, integrity, and availability. On the other hand, the outsourced data might suffer from cloud service providers' (CSPs') illegal behaviours. In particular, a CSP could secretly delete some data in its storage cycle without authorization to save space for other clients' data. On top of this, a CSP might attempt to obtain the data outsourced to the cloud. Thus, whether the attacks are internal or external, the confidentiality and integrity of cloud data are in danger. Hence the design of Remote data Integrity auditing scheme helps to solve the issues by providing efficient security for the data.

In remote data integrity auditing schemes, the data owner firstly needs to generate signatures for data blocks before proving the cloud truly possesses these data blocks in the phase of integrity auditing. And then the data owner uploads these data blocks along with their corresponding signatures to the cloud. The data stored in the cloud is often shared across multiple users in many cloud storage applications, such as Google Drive, Dropbox and iCloud. Data sharing as one of the most common features in cloud storage, allows a few users to share their data with others. However, these shared data stored in the cloud might contain some sensitive information. For instance, the Electronic Health Records

(EHRs) [9] storedand shared in the cloud usually contain patients' sensitive information (patient's name, telephone number and ID number, etc.) and the hospital's sensitive information (hospital's name, etc.). If these EHRs are directly uploaded to the cloud to be shared for research purposes, the sensitive information of patient and hospital will be inevitably exposed to the cloud and the researchers. Besides, the integrity of the EHRs needs to be guaranteed due to the existence of human errors and software/hardware failures in the cloud. Therefore, it is important to accomplish remote data integrity auditing on the condition that the sensitive information of shared data is protected.

A potential method of solving this problem is to encrypt the whole shared file before sending it to the cloud, and then generate the signatures used to verify the integrity of this encrypted file, finally upload this encrypted file and its corresponding signatures to the cloud. This method can realize the sensitive information hiding since only the data owner can decrypt this file. However, it will make the whole shared file unable to be used by others. For example, encrypting the EHRs of infectious disease patients can protect the privacy of patient and hospital, but these encrypted EHRs cannot be effectively utilized by researchers any more.

Distributing the decryption key to the researchers seems to be a possible solution to the above problem. However, it is infeasible to adopt this method in real scenarios due to the following reasons. Firstly, distributing decryption key needs secure channels, which is hard to be satisfied in some instances. Furthermore, it seems very difficult for a user to know which researchers will use his/her EHRs soon when he/she uploads the EHRs to the cloud. As a result, it is impractical to hide sensitive information by encrypting the whole shared file. Thus, how to realize data sharing with sensitive information hiding in remote data integrity auditing with suitable time management is very important and valuable. If the user seems to be untrusted a time seal is been provided so that the authenticated users can access the file at appropriate time intervals this may provide limited access.

II. RELATED WORK

A potential method of solving the security problem is to encrypt the whole shared file before sending it to the cloud and then generate the signatures used to verify the integrity of the encrypted file.In order to verify the integrity of the data stored in the cloud, many remote data integrity auditing schemes have been proposed. To reduce the computation burden on the user side, a Third-Party Auditor (TPA) is introduced to periodically verify the integrity of the cloud data on behalf of user. Ateniese*et al.* [13] firstly proposed a notion of Provable Data Possession (PDP) to ensure the data possession on the untrusted cloud. In their proposed scheme, homomorphic authenticators and random sampling strategies are used to achieve blockless verification and reduce I/O costs. Juels and Kaliski [14] defined a model named as Proof of Retrievability (PoR) and proposed a practical scheme. In this scheme, the data stored in the cloud can be retrieved and the integrity of these data can be ensured. Based on pseudorandom function and BLS signature, Shacham and Waters [15] proposed a private remote data integrity auditing scheme and a public remote data integrity auditing scheme. In order to protect the data privacy, Wang *et al.* [16] proposed a privacy-preserving remote data integrity auditing scheme with the employment of a random masking technique.

Worku*et al.* [17] utilized a different random masking technique to further construct a remote data integrity auditing scheme supporting data privacy protection. This scheme achieves better efficiency compared with the scheme in [16]. To reduce the computation burden of signature generation on the user side, Guan *et al.* [18] designed a remote data integrity auditing scheme based on the indistinguishability obfuscation technique. Shen *et al.* [19] introduced a Third-Party Medium (TPM) to design a light-weight remote data integrity auditing scheme. In this scheme, the TPM helps user generate signatures on the condition that data privacy can be protected. In order to support data dynamics, Ateniese*et al.* [20] firstly proposed a partially dynamic PDP scheme. Erway*et al.* [21] used a skip list to construct a fully data dynamic auditing scheme.

Wang *et al.* [22] proposed another remote data integrity auditing scheme supporting full data dynamics by utilizing Merkle Hash Tree. To reduce the damage of users' key exposure, Yu *et al.* [23] and [24], and Yu and Wang [25] proposed key-exposure resilient remote data integrity auditing schemes based on key update technique [26]. The data sharing is an important application in cloud storage scenarios. To protect the identity privacy of user, Wang *et al.* [27] designed a privacy-preserving shared data integrity auditing scheme by modifying the ring signature for secure cloud storage. Yang *et al.* [28] constructed an efficient shared data integrity auditing scheme, which not only supports the identity privacy but only achieves the identity traceability of users. Fu *et al.* [29] designed a privacy-aware shared data integrity auditing scheme by exploiting a homomorphic verifiable group signature. In order to support efficient user revocation, Wang *et al.* [30] proposed a shared data integrity auditing scheme with user revocation by using the proxy re-signature. With the employment of the Shamir secret sharing technique, Luo *et al.* [31] constructed a shared data integrity auditing scheme supporting user revocation.
The schemes all rely on Public Key Infrastructure (PKI), which incurs the considerable overheads from the complicated certificate management. To simplify certificate management, Wang [32] proposed an identity based remote

data integrity auditing scheme in multi cloud storage. This scheme used the user's identity information such as user's name or e-mail address to replace the public key. Wang *et al.* [33] designed a novel identity-based proxy oriented remote data integrity auditing scheme by introducing a proxy to process data for users. Yu *et al.* [34] constructed a remote data integrity auditing scheme with perfect data privacy preserving in identity-based cryptosystems. Wang *et al.* [35] proposed an identity-based data integrity auditing scheme satisfying unconditional anonymity and incentive. Zhang *et al.* [36] proposed an identity-based remote data integrity auditing scheme for shared data supporting real efficient user revocation Other aspects, such as privacy-preserving authenticators [37] and data deduplication [38], [39] in remote data integrity auditing have also been explored. However, all of existing remote data integrity auditing schemes cannot support data sharing with sensitive information hiding. In this paper, we explore how to achieve data sharing with sensitive information hiding in identity-based integrity auditing for secure cloud storage.

III. METHODOLOGY

We design a practical Identity based shared data integrity auditing scheme with sensitive information hiding for secure cloud storage. A sanitizer is used to sanitize the data blocks corresponding to the sensitive information of the file. The proposed scheme introduces an efficient auditing scheme in order to provide efficient integrity of the data and it also simplifies the users task by auditing the integrity of the cloud by providing the auditing challenge by the TPA and response proof by the cloud. Also, key generation is also mandatory where unique private keys are generated by private key generator (PKG) based on the identity of the user.

A. NOTIONS AND PRELIMINARIESNOTIONS

We show some notations used in the description of our scheme in Table shown below. These notations helps to understand the meaning of the symbols used in equations and theorems which may increase the simplicity and readability.

Notation	Meaning
p	One large prime
G_1, G_2	Multiplicative cyclic groups with order p
g	A generator of group G_1
e	A bilinear pairing map $e: G_1 \times G_1 \rightarrow G_2$
Z_p^*	A prime field with nonzero elements
H	A cryptographic hash function: $H: \{0,1\}^* \rightarrow G_1$
x	An element in Z_p^*
$u', \mu_1, \mu_2, ..., \mu_i, u, g_2$	The elements in G_1
g_1	A public value
n	The number of data blocks of file F
$F = \{m_1, m_2, ..., m_n\}$	The original file F
$F^* = \{m_1^*, m_2^*, ..., m_n^*\}$	The blinded file F^* sent to the sanitizer
$F' = \{m_1', m_2', ..., m_n'\}$	The sanitized file F' stored in the cloud
ID	The user's identity
K_1	The set of the indexes of the data blocks corresponding to the personal sensitive information
K_2	The set of the indexes of the data blocks corresponding to the organization's sensitive information
msk	The master secret key
sk_{ID}	The private key of the user ID
$\Phi = \{\sigma_i\}_{1 \le i \le n}$	The signature set of the blinded file F^*
$\Phi' = \{\sigma_i'\}_{1 \le i \le n}$	The signature set of the sanitized file F'

PRELIMINARIES

In this section, we review some preliminary cryptography knowledge, including bilinear map, Computational Diffie-Hellman (CDH) problem and Discrete Logarithm (DL) problem.

1) **Bilinear Map** Let $G1$, $G2$ be two multiplicative cyclic groups of large prime order p, and g be a generator of $G1$. Bilinear map is a map e: $G1 \times G1 \rightarrow G2$ with the following properties:

a) *Bilinearity*: for all $u, v \in G1$ and $a, b \in Z^*p$, e $(u^a, v^b) = e(u,v)^{ab}$.

b) *Computability*: there exists an efficiently computable algorithm for computing map e.

c) *Non-degeneracy*: $e(g, g) \neq 1$.

2) **Computational Diffie-Hellman (CDH) Problem**

For unknown $x, y \in Z^*p$, given g, g^xandg^yas input, output $g^{xy} \in G1$.The CDH assumption in $G1$ holdsif it is computationally infeasible to solve the CDH problem in $G1$.

3) **Discrete Logarithm (DL) Problem**

For unknown $x \in Z^{*p}$, given g, g^xandg^yas input, outputs x. The DL assumption in $G1$ holds if it is computationally infeasible to solve the DL problem in $G1$.

B. SYSTEM MODEL AND SECURITY MODEL

SYSTEM MODEL

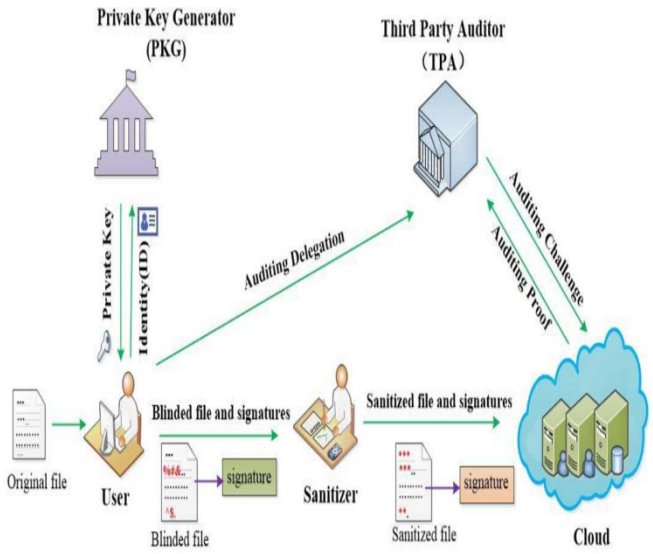


Fig1: The System model.

The system model involves five kinds of different entities: the cloud, the user, the sanitizer, the Private Key Generator (PKG) and the Third-Party Auditor (TPA)

(1) Cloud: The cloud provides enormous data storage space to the user. Through the cloud storage service, users can upload

their data to the cloud and shares their data with others.

(2) User: The user is a member of an organization, which has a large number of files to be stored in the cloud.

(3) Sanitizer: The sanitizer is in charge of sanitizing the data blocks corresponding to the sensitive information (personal sensitive information and the organization's sensitive information) in the file, transforming these data blocks' signatures into valid ones for the sanitized file and uploading the sanitized file and its corresponding signatures to the cloud.

(4) PKG: The PKG is trusted by other entities. It is responsible for generating system public parameters and the private key for the user according to his identity *ID*.

(5) TPA: The TPA is a public verifier. It is in charge of verifying the integrity of the data stored in the cloud on behalf of users.

The user firstly blinds the data blocks corresponding to the personal sensitive information of the file, and generates the corresponding signatures. These signatures are used to guarantee the authenticity of the file and verify the integrity of the file. Then the user sends this blinded file and its corresponding signatures to the sanitizer. After receiving the message from the user, the sanitizer sanitizes these blinded data blocks and the data blocks corresponding to the organization's sensitive information, and then transforms the signatures of sanitized data blocks into valid ones for the sanitized file. Finally, the sanitizer sends this sanitized file and its corresponding signatures to the cloud. These signatures are used to verify the integrity of the sanitized file in the phase of integrity auditing. When the TPA wants to verify the integrity of the sanitized file stored in the cloud, he sends an auditing challenge to the cloud. And then, the cloud responds to the TPA with an auditing proof of data possession. Finally, the TPA verifies the integrity of the sanitized file by checking whether this auditing proof is correct or not. Also, the user can download the file once in 24hours only there by providing the limited access, it also provides the user to access the file only at the appropriate time intervals.

SECURITY MODEL

To formalize the security model, we indicate a game between a challenger *C* and an adversary *A* to show how the adversary *A* is against the security of an identity-based shared data integrity auditing scheme with sensitive information hiding. The data owner is viewed as a challenger *C* and the untrusted cloud server is viewed as an adversary *A* in our security model. This game includes the following phases:

1) **Setup phase**. The challenger *C* runs the *Setup* algorithm to obtain the master secret key *msk*and the system public parameters *pp*, and then sends the public parameters *pp* to the adversary *A*.

2) **Query phase**. In this phase, the adversary *A* makes the following two queries to the challenger *C*.

a) *Extract Queries*: The adversary *A* queries the private key for the identity *ID*. The challenger *C* runs the *Extract* algorithm to generate the private key *skI D*, and sends it to the adversary *A*.

b) *Sig Gen Queries*: The adversary *A* queries the signatures of the file *F*. By running the *Extract* algorithm, the challenger *C* gets the private key. And then the challenger *C* runs the *Sig Gen* algorithm to calculate the signatures of the file *F*. Finally, the challenger *C* sends these signatures to the adversary *A*.

3) **Challenge phase**. In this phase, the adversary *A* acts as a prover and the challenger *C* plays the role of a verifier.

The challenger *C* sends the challenge *chal*= {*i, vi} i*€*I*to the adversary *A*, where *I* € {γ_1, γ_2, . . ., γ_c} (γj € [1, *n*], *j* €[1, *c*] and *c* € [1,*n*]). Meanwhile, it requeststhe adversary *A* to provide a data possession proof *P* for the data blocks {$m_{\gamma 1}$, $m_{\gamma 2}$, . . ., $m_{\gamma c}$} under the *chal*.

4) **Forgery phase**. After receiving the challenge from the challenger *C*, the adversary *A* generates a data possession proof *P* for the data blocks indicated by *chal*to reply the challenger *C*. If this proof *P* can pass the verification of the challenger *C* with non-negligible probability, we say that the adversary *A* succeeds in the above game. This represents the final phase of the security model.

C. DESIGN GOALS

To efficiently support data sharing with sensitive information hiding in identity-based integrity auditing for secure cloud storage, our scheme is designed to achieve the following goals:

1) The correctness:

a) Private key correctness: to ensure that when the PKG sends a correct private key to the user, this private key can pass the verification of the user.

b) The correctness of the blinded file and its corresponding signatures: to guarantee that when the user sends a blinded file and its corresponding valid signatures to the sanitizer, the blinded file and its corresponding signatures he generates can pass the verification of the sanitizer.

c) Auditing correctness: to ensure that when the cloud properly stores the user's sanitized data, the proof it generates can pass the verification of the TPA.

2) Sensitive information hiding to ensure that the personal sensitive information of the file is not exposed to the sanitizer, and all of the sensitive information of the file is not exposed to the cloud and the shared users.

3) Auditing soundness: to assure that if the cloud does not truly store user's intact sanitized data, it cannot pass the TPA's verification.

IV. THE PROPOSED SCHEME

In order to achieve data sharing with sensitive information hiding, we consider making use of the idea in the sanitizable signature [40] to sanitize the sensitive information of the file by introducing an authorized sanitizer. Nonetheless, it is infeasible if this sanitizable signature is directly used in remote data integrity auditing. Firstly, this signature in [40] is constructed based on chameleon hashes [41]. However, a lot of chameleon hashes exhibit the key exposure problem. To avoid this security problem, the signature used in [40] requires strongly unforgeable chameleon hashes, which will inevitable incur huge computation overhead [41]. Secondly, the signature used in [40] does not support blockless verifiability. It means that the verifier has to download the entire data from the cloud to verify the integrity of data, which will incur huge communication overhead and excessive verification time in big data storage scenario. Thirdly, the signature used in [40] is based on the PKI, which suffers from the complicated certificate management.

In order to address above problems, we design a new efficient signature algorithm in the phase of signature generation. The designed signature scheme supports blockless verifiability, which allows the verifier to check the integrity of data without downloading the entire data from the cloud. In addition, it is based on identity-based cryptography, which simplifies the complicated certificate management. In our proposed scheme, the PKG generates the private key for user according to his identity ID. The user can check the correctness of the received private key. When there is a desire for the user to upload data to the cloud, in order to preserve the personal sensitive information of the original file from the sanitizer, this user needs to use a blinding factor to blind the data blocks corresponding to the personal sensitive information of the original file. When necessary, the user can recover the original file from the blinded one by using this blinding factor. And then this user employs the designed signature algorithm to generate signatures for the blinded file. These signatures will be used to verify the integrity of this blinded file. In addition, the user generates a file tag, which is used to ensure the correctness of the file identifier name and some verification values.

The user also computes a transformation value that is used to transform signatures for sanitizer. Finally, the user sends the blinded file, its corresponding signatures, and the file tag along with the transformation value to the sanitizer. When the above messages from user are valid, the sanitizer firstly sanitizes the blinded data blocks into a uniform format and also sanitizes the data blocks corresponding to the organization's sensitive information to protect the privacy of organization,and then transforms their corresponding signatures into valid ones for sanitized file using transformation value. Finally, the sanitizer uploads the sanitized file and the corresponding signatures to the cloud. When the data integrity auditing task is performed, the cloud generates an auditing proof according to the challenge from the TPA. The TPA can verify the integrity of the sanitized file stored the cloud by checking whether this auditing proof is correct or not. The details will be described in the following subsection.

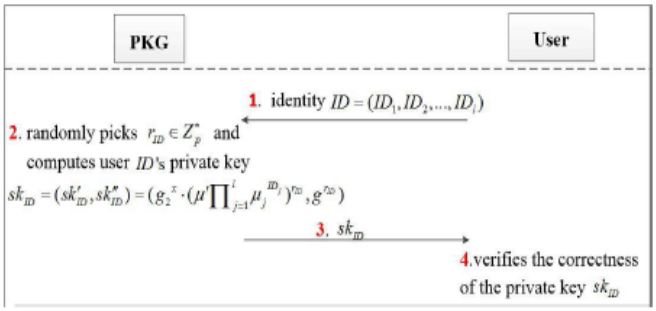


Fig2: The process of private key Generation.

Description of the Proposed Scheme

1. **Algorithm Setup**(1k)

a) The PKG chooses two multiplicative cyclic groups G1 and G2 of prime order p, a generator g of G1, a bilinear map e: G1 ×G1 → G2 and a pseudorandom function f: Z∗ p × Z∗ p → Z∗p.

b) The PKG randomly chooses an element x ∈ Z∗ p, elements µ, µ1, µ2..., µ l, u, g2 ∈ G1 and a cryptographic hash function H: {0,1} ∗ → G1.

c) The PKG computes the public value g1=gx and the master secret key msk= g2x. d) The PKG publishes system

parameters pp=(G1, G2, p, e, g, μ, μ1, μ2,...,μ l,u,g1,g2, H, f) and holds the master secret key msk.

2. **Algorithm Extract (pp, msk, ID)**

a) After receiving the user's identity ID = (ID 1, ID 2,...,ID l) ∈{ 0,1}l, the PKG randomly picks a value rID∈ Z∗ p and computes skID = (skID,skID) = (g2x · (μlj=1 μj IDj)rID,grID) as the private key of the user ID. The PKG sends it to the user ID.

b) The user ID verifies the correctness of the received private key skID by checking whether the following equation holds or not. e (skID, g) = (g1, g2) ·e (μlj=1 μj IDj, skID).

If above equation does not hold, the user ID refuses the private key skID; otherwise, accepts it.

3. **Algorithm SigGen (F, skID, ssk, name)**

a) The user ID randomly chooses a value r ∈ Z∗ p, and calculates a verification value gr. Then the user ID randomly chooses a seed k1 ∈ Z∗ p as the input secret key of pseudo-random function f. The user ID employs the secret seed k1 to calculate the blinding factor αi = fk1(i, name)(i∈ K1) which is used to blind the data blocks corresponding to the personal sensitive information, where name∈ Z∗ p is a random value chosen as the file identifier.

b) In order to preserve the personal sensitive information from the sanitizer, the user ID should blind the data blocks corresponding to the personal sensitive information of the original file F before sending it to the sanitizer. The indexes of these data blocks are in set K1. The user ID computes the blinded data block m∗i =mi +αi for each block mi ∈ Z∗ p (i∈ K1) of the original file F. The blinded file is F∗=m∗ 1, m∗2,..., m∗ n, where m∗i =mi only ifi∈ [1, n] and i / ∈ K1; otherwise, m∗i =mi.

c) For each block m∗i∈ Z∗ p (i∈[1,n]) of the blinded file F∗, the user ID calculates the signature σi on block m∗i as follows: σi = g2x(μlj=1 μj IDj)rID(H(name||i)·um∗i)r. Let = { σi}1≤i≤n be the set of signatures for the blinded file F∗.

d) The user ID sets τ0 = name||grID||gr and calculates the file tag by computing τ = τ0||SSigssk(τ0), where SSigssk(τ0) is the signature on τ0 under the signing private key ssk.

e) The user ID calculates a transformation value β=ur which is used to transform the signature in Sanitization algorithm. He sends {F∗,,τ,K1} along with β to the sanitizer, and then deletes these messages from local storage. In addition, when the user ID wants to retrieve his file F, he can send a request to the sanitizer. And then the sanitizer downloads and sends the blinded file F∗ to the user. The user ID can recover the original file F using the blinding factor.

4. **Algorithm Sanitization(F∗, ,)**

a) The sanitizer checks the validity of the file tag τ by verifying whether SSigssk(τ0) is a valid signature. If it is a valid signature, the sanitizer parses τ0 to obtain file identifier name name and verification values grID and gr, and then does the following steps.

b) The sanitizer respectively verifies the correctness of signature σi (i∈ [1, n]) as follows: e(σi,g)=e(g1,g2)·e(μlj=1 μj IDj ,grID) ·e(H(name||i)·um∗i ,gr). (2) If the equation (2) does not hold, the sanitizer thinks the signatures invalid; otherwise, goes to the step c.

c) The sanitizer verifies the correctness of the transformation value β by checking whether e(u,gr) = e(β,g) holds or not.

If the above equation holds, the sanitizer will sanitize the blinded data blocks and the data blocks corresponding to the organization's sensitive information. The indexes of these data blocks are in sets K1 and K2. In SigGen algorithm, the data blocks whose indexes are in set K1 have been blinded by the user ID, which will make the contents of these data blocks become messy code. In order to unify the format, the sanitizer can use wildcards to replace the contents of these data blocks. For example, in an EHR, Bob is a user's name. After blinded by the medical doctor, the contents of this sector will become messy code. To unify the format, the sanitizer replaces these messy code with ***. In addition, to protect the privacy of organization, the sanitizer also sanitizes the data blocks whose indexes are in set K2. For example, in an EHR, the sanitizer replaces the information such as hospital's name with ***. And then the sanitizer transforms the signatures of data blocks in sets K1 and K2 into valid ones for sanitized file F as follows:

$$\sigma_i' = \begin{cases} \sigma_i(\beta)^{m_i'-m_i^*} & i \in K_1 \cup K_2 \\ \sigma_i & i \in [1,n] \text{ and } i \notin K_1 \cup K_2 \end{cases}$$

$$= g_2{}^x(\mu' \prod_{j=1}^{l} \mu_j{}^{ID_j})^{r_{ID}}(H(name||i) \cdot u^{m_i'})^r$$

Let $\Phi' = \{\sigma_i'\}_{1 \le i \le n}$ be the set of sanitized file's signatures.

d) The sanitizer sends {F,} to the cloud, and then sends the file tag τ to the TPA. Finally, he deletes these messages from local storage.

5. **Algorithm ProofGen(F, ,chal)**

a) The TPA verifies the validity of the file tag τ. The TPA will not execute auditing task if the file tag τ is invalid; otherwise, the TPA parses τ0 to obtain file identifier name name and verification values grID and gr, and then generates an auditing challenge chal as follows:

i) Randomly picks a set I with c elements, where I⊆ [1, n].

ii) Generates a random value vi ∈ Z∗ p for each i∈ I.

iii) Sends the auditing challenge chal = {i,vi}i∈I to the cloud.

b) After receiving an auditing challenge from the TPA, the cloud generates a proof of data possession as follows:

i) Computes a linear combination of data blocks λ=i∈Imivi.

ii) Calculates an aggregated signature σ = i∈I σ i vi. iii) Outputs an auditing proof P ={λ,σ} to the TPA.

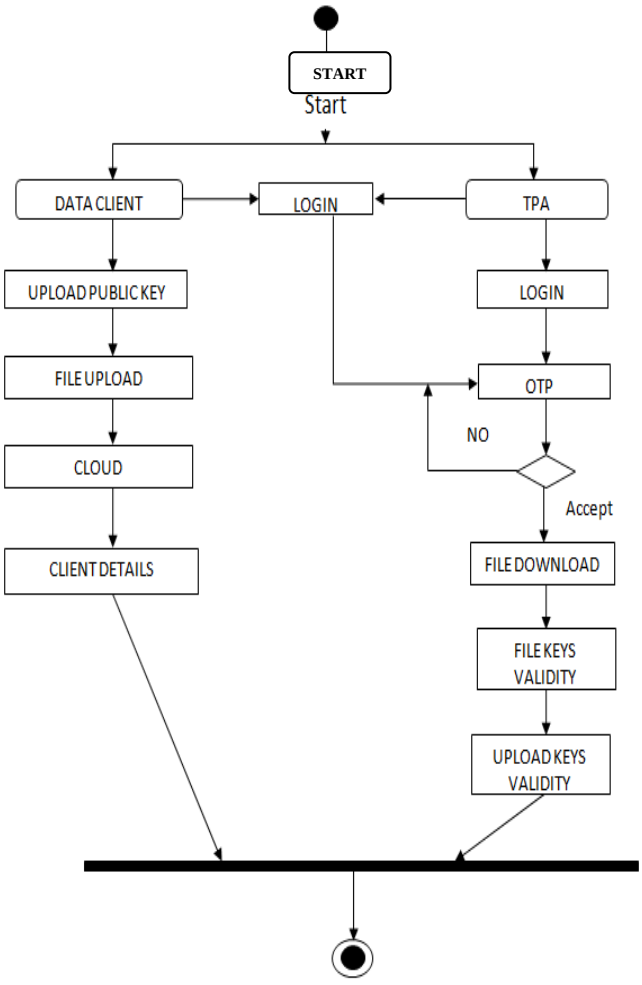
6. **Algorithm Proof Verify(chal,pp,P)**

The TPA verifies the correctness of auditing proof as follows:

$$e(\sigma, g) = e(g_1, g_2)^{\sum_{i\in I} v_i} \cdot e(\mu' \prod_{j=1}^{l} \mu_j^{ID_j}, g^{r_{ID}})^{\sum_{i\in I} v_i} \cdot e(\prod_{i\in I} H(name||i)^{v_i} \cdot u^{\lambda}, g^{r}). \quad (3)$$

If the equation (3) holds, the sanitized file stored in the cloud is intact; otherwise, it is not.

FLOW CHART



Flow charts are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, the flowcharts can be used to describe the business and operational step-by-step workflows of components in a system. An flowchart shows the overall flow of control.

Once the data owner or the third-party auditor login, the data owner allow to upload public key into the cloud. Also data owner upload the file into the cloud server. Before uploading, data owner makes sure that the file is in encrypted form in order to achieve data integrity and confidentiality. Also the client details means who can view and who all are accessible and who can upload the file into the cloud. Once the third-party auditor login he receives OTP, if that OTP is authenticates it allows to download a file and also he can check file integrity by using sanitized signature during auditing phase. otherwise received OTP is not authenticated so that he should again login with proper authentication.

V. RESULTS AND DISCUSSION

We evaluate the performance of the proposed scheme by several experiments. We run these experiments on a Linux machine with an Intel Pentium 2.30GHz processor and 8GB memory. All these experiments use C programming language with the free Pairing-Based Cryptography (PBC) Library and the GNU Multiple Precision Arithmetic (GMP) . In our experiments, we set the base field size to be 512 bits, the size of an element in Z*p to be |p| = 160 bits, the size of data file to be 20MB composed by 1,000,000 blocks, and the length of user identify to be 160 bits.

1) The Performance of Different Processes: To effectively evaluate the performance in different processes, we set the number of data blocks to be 100 and the number of sanitized data blocks to be 5 in our experiment. The private key generation and private key verification spend nearly the same time, which are nearly 0.31s. The time consumed by the signature generation is 1.476s. The time of signature verification and that of sensitive information sanitization respectively are 2.318s and 0.041s. So we can conclude that in these processes, the signature verification spends the longest time and the sensitive information sanitization spends the shortest time.

To evaluate the performance of signature generation and signature verification, we generate the signatures for different number of blocks from 0 to 1000 increased by an interval of 100 in our experiment. The time cost of the signature generation and the signature verification both linearly increases with the number of the data blocks. The time of signature generation ranges from 0.121s to 12.132s. The time of signature verification ranges from 0.128s to 12.513s.

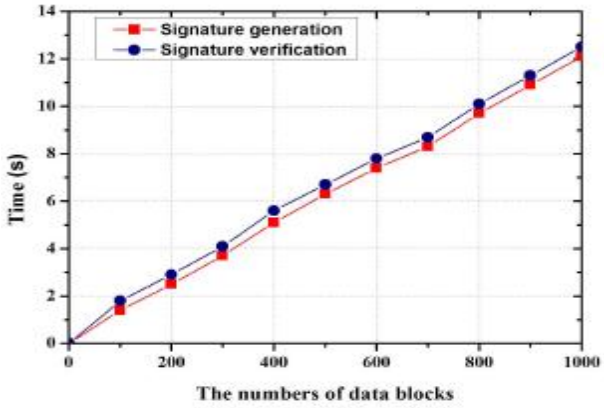


Fig3: The computation overhead in the process of signature generation and signature verification.

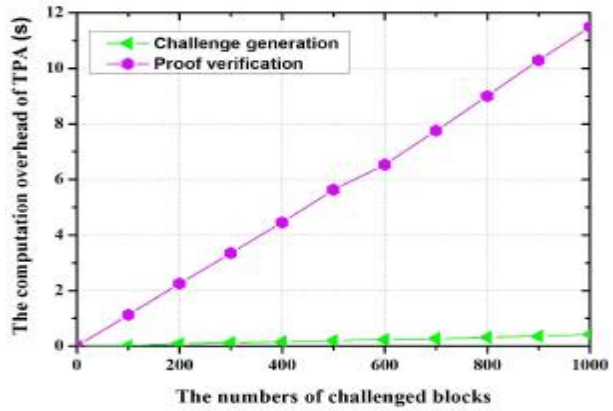


Fig4: The computation overhead of the TPA in the phase of integrity auditing.

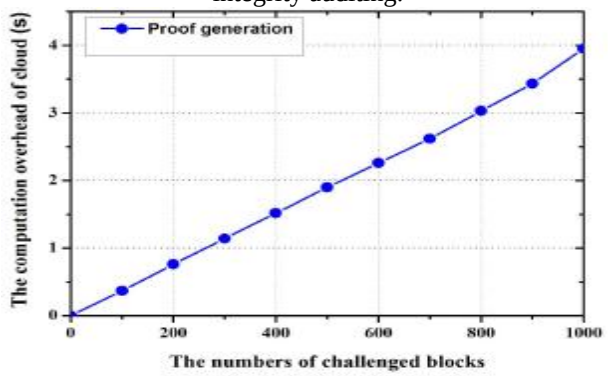


Fig5: The computation overhead of the cloud in the phase of integrity auditing.

2) Performance of Auditing: With the different number of challenged data blocks, we respectively show the computation overhead of the TPA and that of the cloud in integrity auditing phase in Fig. 4 and Fig. 5. In our experiment, the number of challenged data blocks varies from 0 to 1,000. As shown in Fig. 9, we see the that the computation overheads of challenge generation and proof verification on the TPA side linearly increase with the number of challenged data blocks. The computation overhead of proof verification varies from 0.317s to 11.505s. Compared with the time of proof verification, the time of challenge generation increases slowly, just varying from 0.013s to 0461s. From Fig. 10, we have the observation that the computation overhead of proof generation on the cloud side varies from 0.021s to 3.981s. So we can conclude that, with the more challenged data blocks, both the TPA and the cloud will spend the more computation overheads.

VI. CONCLUSION AND FUTURE SCOPE

In this paper, we proposed an identity-based data integrity auditing scheme for secure cloud storage, which supports data sharing with sensitive information hiding. In our scheme, the file stored in the cloud can be shared and used by others on the condition that the sensitive information of the file is protected. Besides, the remote data integrity auditing with time management is still able to be efficiently executed. The security proof and the experimental analysis demonstrate that the proposed scheme achieves desirable security and efficiency. Also sanitization process further increases the performance of the auditing scheme where the access of the sanitized file is more difficult only the authorized users are provided for the access.

ACKNOWLEDGEMENT

The completion of Project brings with great sense of satisfaction, but it is never completed without thanking the persons who are all responsible for its successful completion. First and foremost, I wish to express our deep sincere feelings of gratitude to my Institution, **EAST WEST INSTITUTE OF TECHNOLOGY**, for providing me an opportunity to do our education. I extend my deep sense of sincere gratitude to **Dr. K Channakeshavalu**(Principal), I express my heartfelt sincere gratitude to **Dr. Arun Biradar**(HOD),I would like to thank my guide **Mrs. Supritha N** Assistant Professor, Department of Computer Science and Engineering. Finally, I would like to thank all the Teaching, Technical faculty and supporting staff members of Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru, for their support

REFERENCES

[1] J. Yu, R. Hao, H. Xia, H. Zhang, X. Cheng, and F. Kong, "Intrusionresilient identity-based signatures: Concrete scheme in the standard model and generic construction," Inf. Sci., vols. 442–443, pp. 158–172, May 2018.

[2] W. Shen, G. Yang, J. Yu, H. Zhang, F. Kong, and R. Hao, "Remote data possession checking with privacy-preserving authenticators for cloud storage," Future Gener. Comput. Syst., vol. 76, pp. 136–145, Nov. 2017.

[3] Y. Yu et al., "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," IEEE Trans. Inf. Forensics Security, vol. 12, no. 4, pp. 767–778, Apr. 2017.
[4] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," IEEE Trans. Inf. Forensics Security, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.
[5] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," J. Netw. Comput. Appl., vol. 82, pp. 56–64, Mar. 2017.
[6] J. Hur, D. Koo, Y. Shin, and K. Kang, "Secure data deduplication with dynamic ownership management in cloud storage," IEEE Trans. Knowl. Data Eng., vol. 28, no. 11, pp. 3113–3125, Nov. 2016.
[7] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," IEEE Trans. Comput., vol. 65, no. 8, pp. 2386–2396, Aug. 2016.
[8] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," IEEE Trans. Inf. Forensics Security, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.
[9] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," J. Syst. Softw., vol. 113, pp. 130–139, Mar. 2016.
[10] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," IEEE Trans. Inf. Forensics Security, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.
[11] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multikeyword fuzzy search over encrypted outsourced data with accuracy improvement," IEEE Trans. Inf. Forensics Security, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.
[12] Q. Jiang, M. K. Khan, X. Lu, J. Ma, and D. He, "A privacy preserving three-factor authentication protocol for e-health clouds," J. Supercomput., vol. 72, no. 10, pp. 3826–3849, 2016.
[13] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multikeyword ranked search scheme over encrypted cloud data," IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 2, pp. 340–352, Feb. 2016.
[14] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," IEICE Trans. Commun., vol. 98, no. 1, pp. 190–200, 2015.
[15] J. Shen, H. Tan, S. Moh, I. Chung, Q. Liu, and X. Sun, "Enhanced secure sensor association and key management in wireless body area networks," J. Commun. Netw., vol. 17, no. 5, pp. 453–462, 2015.
[16] C. Guan, K. Ren, F. Zhang, F. Kerschbaum, and J. Yu, "Symmetrickey based proofs of retrievability supporting public verification," in Computer Security—ESORICS. Cham, Switzerland: Springer, 2015, pp. 203–223.
[17] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," IEEE Trans. Inf. Forensics Security, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.
[18] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," IEEE Trans. Serv. Comput., vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.
[19] Y. Luo, M. Xu, S. Fu, D. Wang, and J. Deng, "Efficient integrity auditing for shared data in the cloud with secure user revocation," in Proc. IEEE Trustcom/BigDataSE/ISPA, Aug. 2015, pp. 434–442.
[20] H. Wang, "Identity-based distributed provable data possession in multicloud storage," IEEE Trans. Serv. Comput., vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.
[21] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacypreserving public auditing scheme for cloud storage," Comput. Electr. Eng., vol. 40, no. 5, pp. 1703–1713, 2014.
[22] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, "Security issues in cloud environments: A survey," Int. J. Inf. Secur., vol. 13, no. 2, pp. 113–170, Apr. 2014.
[23] L. F. B. Soares, D. A. B. Fernandes, J. V. Gomes, M. M. Freire, and P. R. M. Inácio, Cloud Security: State of the Art. Berlin, Germany: Springer, 2014.
[24] H. Shacham and B. Waters, "Compact proofs of retrievability," J. Cryptol., vol. 26, no. 3, pp. 442–483, Jul. 2013.
[25] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreserving public auditing for secure cloud storage," IEEE Trans. Comput., vol. 62, no. 2, pp. 362–375, Feb. 2013
[26] M. Green, "The threat in the cloud," IEEE Security Privacy, vol. 11, no. 1, pp. 86–89, Jan./Feb. 2013.
[27] K. Yang and X. Jia, "Data storage auditing service in cloud computing: Challenges, methods and opportunities," World Wide Web, vol. 15, no. 4, pp. 409–428, 2012.
[28] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in Proc. IEEE 5th Int. Conf. Cloud Comput. (CLOUD), Jun. 2012, pp. 295–302.
[29] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., vol. 53, no. 6, p. 50, 2011.
[30] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847–859, May 2011.
[31] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype,and reality for delivering computing as the 5th utility," Future Generat. Comput. Syst., vol. 25, no. 6, pp. 599–616, 2009.
[32] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. 16th ACM Conf. Comput. Commun. Secur., 2009, pp. 213–222
[33] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. 4th Int. Conf. Secur. Privacy Commun. Netw., 2008, Art. no.9.
[34] G. Ateniese et al., "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 598–609.
[35] A. Juels and B. S. Kaliski, Jr., "Pors: Proofs of retrievability for large files," in Proc. 14th ACM Conf. Comput. Commun. Secur., 2007, pp. 584–597.
[36] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, "Sanitizable signatures," in Proc. 10th Eur. Symp. Res. Comput. Secur. Berlin, Germany: Springer-Verlag, 2005, pp. 159–177.
[37] G. Ateniese and B. de Medeiros, "On the key exposure problem in chameleon hashes," in Security in Communication Networks. Berlin, Germany: Springer, 2005, pp. 165–179.
[38] Q. Jiang, J. Ma, and F. Wei, "On the security of a privacy-aware authentication scheme for distributed mobile cloud computing services," IEEE Syst. J., to be published.
[39] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," IEEE Trans. Big Data, to be published, doi: 10.1109/TBDATA.2017.2701347.
[40] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," IEEE Trans. Serv. Comput., to be published, doi: 10.1109/TSC.2016.2633260.
[41] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," IEEE Trans. Depend. Sec. Comput., to be published, doi: 10.1109/TDSC.2018.2829880.

