Accepted Manuscript

One secure data integrity verification scheme for cloud storage

Yongkai Fan, Xiaodong Lin, Gang Tan, Yuqing Zhang, Wei Dong,
Jing Lei

PII: S0167-739X(18)31100-2
DOI: https://doi.org/10.1016/j.future.2019.01.054
Reference: FUTURE 4747

To appear in: *Future Generation Computer Systems*

Received date : 8 May 2018
Revised date : 19 January 2019
Accepted date : 27 January 2019

Please cite this article as: Y. Fan, X. Lin, G. Tan et al., One secure data integrity verification scheme for cloud storage, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2019.01.054

One Secure Data Integrity Verification Scheme for Cloud Storage

Yongkai Fan[1,2], Xiaodong Lin[1,2], Gang Tan[3], Yuqing Zhang[4], Wei Dong[5], JingLei[1,2]

[1]*Beijing Key Lab of Petroleum Data Mining, China University of Petroleum, Beijing, China*

[2]*Dept. of Computer Science and Technology, China University of Petroleum, Beijing, China*

[3]*Dept. of Computer Science and Engineering, Penn State University, PA, USA*

[4]*National CNIP Center, University of Chinese Academy of Sciences, Beijing, China*

[5]*Research Institute of Information Technology, Tsinghua University, Beijing, China*

Abstract. Cloud computing is a novel kind of information technology that users can enjoy sundry cloud services from the shared configurable computing resources. Compared with traditional local storage, cloud storage is a more economical choice because the remote data center can replace users for data management and maintenance, which can save time and money on the series of work. However, delivering data to an unknown Cloud Service Provider (CSP) makes the integrity of data become a potential vulnerability. To solve this problem, we propose a secure identity based aggregate signatures (SIBAS) as the data integrity checking scheme which resorts Trusted Execution Environment (TEE) as the auditor to check the outsourced data in the local side. SIBAS can not only check the integrity of outsourced data, but also achieve the secure key management in TEE through Shamir's (t,n) threshold scheme. To prove the security, security analysis in the random oracle model under the computational Diffie-Hellman assumption shows that SIBAS can resist attacks from the adversary that chooses its messages and target identities, experimental results also show that our solution is viable and efficient in practice.

Keywords: Trusted Execution Environment, Cloud Storage, Integrity Verification, Identity-Based Aggregate Signatures, Shamir's (t,n) threshold scheme

1 Introduction

Cloud computing was treated as a new network information technology architecture to meet the increasing need of computing owing to its own unprecedented characteristics: broad network access, on-demand self-service, resource pooling that location independence, the rapid elasticity of the resource and high-quality of measured services [1]. Different from the traditional technology, cloud computing allows individuals and IT enterprises to outsource the data to the cloud, which provides users with more flexible access services. Cloud computing is widely used in various networks, such as wireless sensor network[2,3]. The wireless sensors can be treated as the cloud nodes, the collected signals of which are transmitted to the cloud for secure storage.

As one of the core techniques in cloud computing, cloud storage was widely discussed because of its lower cost and higher efficiency. Together with the computing architecture called "software as a service" (SaaS), cloud data storage commits to switching data centers to pools of computing service on a large scale. At the meantime, by the rapid growth of the network bandwidth with the reliable and flexible network connection, it's possible that cloud users can enjoy high-quality cloud services from data that reside solely in the remote data centers [1]. Different from the traditional storage technology (direct attached storage, redu[illegible]e space and data access through independent geographical locations. Namely, cloud users can access the outsourced data easily anytime, anywhere, through any networked device that connected to the cloud.

Although the cloud data storage brings great convenience to end users, security issues should not be neglected for computer systems are subjected to an increasing range of attacks. While users deliver their data to the efficient yet unreliable CSP, due to lack of secure identity authentication and high-intensity access control on the identification, protection of the data integrity and privacy in remote cloud servers will be a great challenge. For example, a cross-VM-side-channel attack may be launched by sophisticated attackers and caused a data leakage of legitimate users [20]. Moreover, data loss could occur in any cloud infrastructure, even the cloud provider with the highest degree of protection is no exception. Sometime, several CSPs may choose to discard the data that has been accessed infrequently to save the storage space and maximize their profit. More abominable, they even concealed the fact that the data was lost to the user and pretended that the user's data was still intact and stored in the cloud [4,5]. Ultimately, users must bear these unnecessary losses by themselves. Security issues make users nervous and hesitate to outsource their data to the cloud. To some extent, the tension of losing data hinders the widespread of cloud technology.

To mitigate the tension, data integrity verification is proposed. A popular method is to resort an independent third-party auditor (TPA) services to check the integrity of the outsourced data, which is called "public verification" [24]. Such a concept was used in lots of work. Recently, such a concept has been applied to many different systems and security models [19,20,21]. In these research work, all the auditing tasks were done by TPA, it can interact with both CSPs and users to gain the information required for integrity verification. Throughout the process, users do not need to know how TPA performs the verification to check the integrity of outsourced data. Instead, they will receive an audit report of outsourced data from TPA, which hints at whether the integrity of the data has been destroyed. It seems that users can perform secure data integrity verification and save a lot of overhead by introducing TPA, but there are two fundamental requirements have to meet: 1) TPA is efficient to check the integrity of outsourced data without a data copy and eliminating the online burden of users. 2) TPA should not bring new vulnerabilities to users' security and privacy [5]. In other words, users must take default that TPA is trusted and will not deceive the users or infringe on the user's privacy. However, it is based on the assumption of the ideal state and difficult to realize in the commercial context since that it cannot avoid skillful attacks (i.e. the Man-in-the-Middle attack) [29]. Besides, the introduction of TPA in cloud infrastructure means that users need to pay extra fees in addition to the cloud service, as users also have to pay for the management and maintenance of TPA.

In this paper, we propose a new scheme to replace TPA with one secure environment on the client side which securely checks the integrity of outsourced data without using meta-data stored in the cloud. Specifically, we resort to Trusted Execution Environment (TEE) [17] that is running on local infrastructure acts as an auditor to verify the outsourced data and perform secure key management. A Trusted Execution Environment (TEE) holds its own independent running space that is isolated from a Rich Execution Environment (REE), which ensures that any adversary in REE cannot grab the privacy information or pry into the results of verification without the knowledge of the users. Therefore, users do not need to worry about disclosure of their private information while requesting for data verification. In addition, it can help users to save unnecessary costs while enjoying the cloud services.

In our approach, we present our scheme based on TEE which aims to protect user's assets, which is referred to SIBAS. Our contributions can be summarized into five main points as follows:

- To the best of our knowledge, we firstly put forwards integration of the advantages of TEE and identity-based encryption to construc … ons such as addition, deletion, and modification.

- We design and implement a secure terminal architecture based on Trusted Execution Environment. Since TEE is an independent execution space isolated from Rich Execution Environment (REE), such a mechanism can support adequate security in the process of data integrity verification.
- While outsourcing a large-scale file to the cloud, we utilize Shamir's (t, n) threshold scheme to encrypt the private key and transmit it to the cloud to reduce the storage consumption of the local side.
- Our scheme not only supports the integrity verification for a single file, but also achieves concurrent verification for multiple files.
- To test the feasibility and reliability of our scheme, theoretical and experimental analysis have been done.

The remainder of this paper is organized as follows: We first introduce some background information in section 2. In Section 3, the system model will be presented in detail. Then we prove that SIBAS is secure against the adversary that aim to deceive users and evaluate the performance of our proposal in section 4 and section 5 respectively. In section 6 we describe the related works that have been done. Finally, we conclude the paper in Section 7.

2 Problem formulation

Identity-based encryption was proposed by Shamir [18], then Boneh et al. [22] first proved that a fully functional and effective identity-based encryption scheme can be constructed through any bilinear map, a lot of research encryption schemes were proposed based on the bilinear map. In our work, we construct SIBAS base on the bilinear map as same as the other previous works. Before we describe the whole scheme in detail, some preparations of our scheme are shown in this section.

2.1 Preliminaries

2.1.1 Bilinear maps

Let G_1 , G_2 be the cyclic group of prime order p, e is a bilinear map that satisfies $e: G_1 \times G_1 \rightarrow G_2$ and meets the following properties.

1. Bilinear: There exists $e(aX, bY + cZ) = e(aX, bY) \cdot e(aX, cZ)$, for $\forall a, b, c \in Z_p$, $\forall X, Y, Z \in G_1$.

From the bilinear mapping, we can get the following equation: $e(aX, bY) = e(X, Y)^{ab}$, $\forall a, b \in Z_p$, $\forall X, Y \in G_1$.

2. Non-degenerate: There exists $\forall X, Y \in G_1$ to meet the inequality $e(X, Y) \neq 1$.

3. Computable: For $\forall X, Y \in G_1$, there exists an appropriate algorithm to compute $e(X, Y)$.

2.1.2 Gap Diffie–Hellman (GDH) groups

Let G be a cyclic multiplicative group generated by g with the prime order q. We can define the following cryptography problem in G.

(1) Discret ion $g = nh$ while the integer exists.

(2) Computation Diffie–Hellman problem (CDH): Given $P, aP, bP \in G$ for unknown $\forall a, b \in Z_p$ and a bilinear map $e: G_1 \times G_1 \rightarrow G_2$, then compute abP.

(3) Decision Diffie–Hellman problem (DDH): Given $P, aP, bP, cP \in G$ for unknown $\forall a, b, c \in Z_p$. Determine whether the following equation holds: $c \equiv ab \bmod q \iff e(P, cP) = e(aP, bP)$.

Given a bilinear map $e: G_1 \times G_1 \rightarrow G_2$, we call G_1 is a GDH group if the CDH problem in G_1 is believed to be hard, while the DDH problem in G_1 is easy to be calculated. Specially, if there is no polynomial time probabilistic algorithm $\mathcal{A}$ can solve the CDH problem with a negligible advantage ε, we say that it is computationally infeasible to solve the CDH problem in G_1.

2.1.3 Identity-based aggregate signatures

Identity-based aggregate signatures (IBAS) was firstly introduced by Gentry and Silverberg [23], which is consisted by five phases: Setup, Private key generation, Individual signing, Aggregation, Verification. In Gentry`s scheme, a location-independent private key generator (PKG, generally a trusted third party server) is first launched out and generates a master secret $s \in Z_p$ as well as the system parameter $params = (G_1, G_2, e, P, Q, H_1, H_2, H_3)$. For G_1 and G_2 are the bilinear group of prime order p, P is an arbitrary generator and sets $Q = sP$ $(P, Q \in G_1)$, H_i (for $i = \{1, 2, 3\}$) are cryptographic hash functions that satisfy $H_1, H_2 : \{0,1\}^* \in G_1$ and $H_3 : \{0,1\}^* \in Z_p$. Then the private key $sP_{i,j}$ is generated based on the user ID id_i, where $P_{i,j} = H_1(id_i, j)$ for $j \in \{0,1\}$. While the signing phase, the user id_i signs the message m_i to be processed with the private key as well as a "dummy message" ω. It first computes the two values $P_\omega = H_2(\omega)$ and $c_i = H_3(id_i, m_i, \omega)$, then it generates a random value $r_i \in Z_p$ and computes its signature (ω, S_i, T_i), where $S_i = r_i P_\omega + sP_{i,0} + c_i sP_{i,1}$, $T_i = r_i P$. After that, a collection of the individual signatures can be aggregated into one signature (ω, S_n, T_n), where $S_n = \sum_{i=1}^{n} S_i$ and $T_n = \sum_{i=1}^{n} T_i$. If anyone attempts to verify the aggregate signature, a corresponding information will be taken as input to proof the correctness of the signature by $e(S_n, P) =$

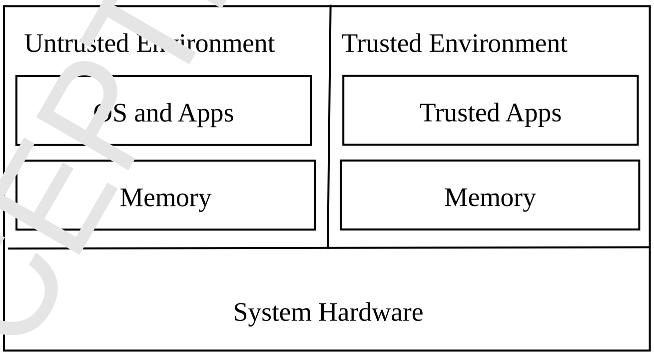

Fig. 1. Trusted Execution Environment (TEE) architecture [28].

$e(T_n, P_\omega) e(Q, \sum_{i=1}^{n} P_{i,0} + \sum_{i=1}^{n} P_{i,1})$.

2.1.4 Shamir's (t, n) threshold scheme

Shamir's (t, n) threshold scheme (Shamir's secret sharing scheme) is well-known in cryptography, which is used to share a secret among a group of participants. In this scheme, each participant holds partial information about the sharing secret, then the secret can be reconstructed if the number of shared participants meets the requirements. The mathematical description … $P_2, \cdots, P_n\}$) for sharing the secret s, and set a threshold value t, for $(t \leq n)$. Then given a finite field $\mathbb{F}_q^*$, each participant is allocated with an identification, which can be denoted as $x_1, x_2, \cdots, x_n \in \mathbb{F}_q^*$ and the value of x_i is publicly available. First, the system randomly chooses $a_i \in \mathbb{F}_q^*$ $(i = 1,2,\cdots,t-1)$ and constructs t time polynomial $f(x) = s + a_1 x + \cdots + a_{t-1}x^{t-1}$. After that, the system computes $f(x_i)$ and send them to each participant P_i as the sub-secret. While someone wants to recovery the secret s, he/she first requests any t participants from n participants to help. While the requester gets t sub-secrets $(x_i, f(x_i))$, he/she can reconstruct the t time polynomial $f(x)$ using the Lagrange interpolation by $f(x) = \sum_{i=1}^{t} f(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{t} \frac{x-x_j}{x_i-x_j}$. Then the secret s can be calculated as $s = f(0) = \sum_{i=1}^{t} f(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{t} \frac{x-x_j}{x_i-x_j} \ (mod\ q)$.

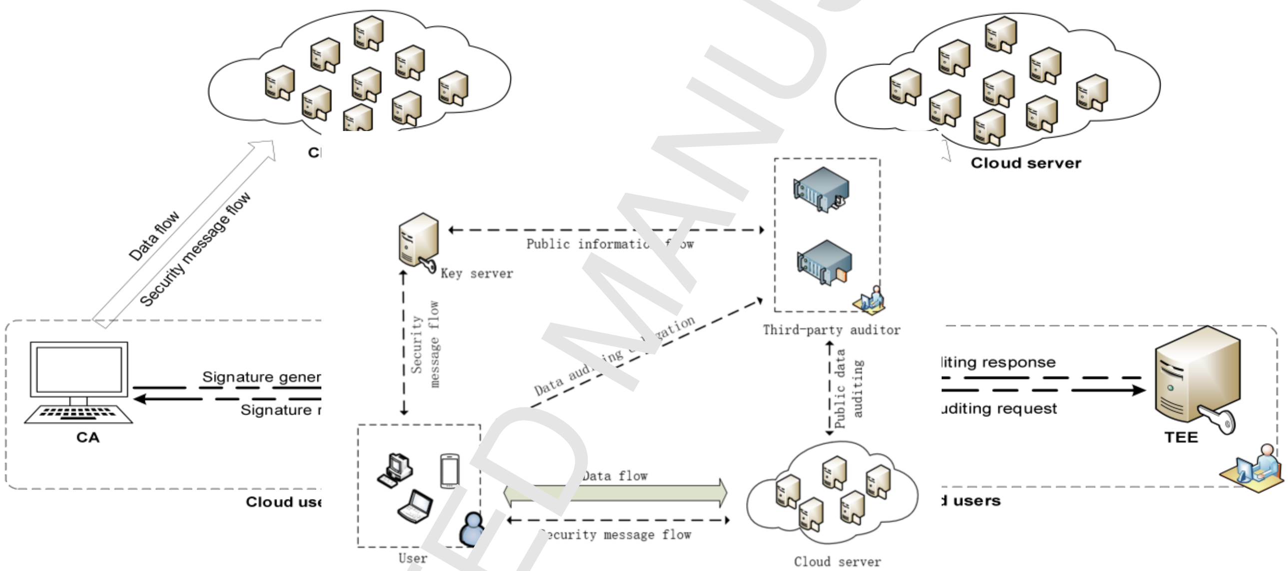

Fig. 3-(a), 3-(b). Th… re generation and challenge phases in (a) and the **signature aggregation** and **verify phases** in (b).

Fig. 2. Cloud architecture of NaEPASC [20]

2.1.5 Trusted Execution Environment (TEE)

The concept of TEE was proposed by the Global Platform in 2011. A complete system of TEE includes two execution environments that are physically separated: one is untrusted and responsible for hosting the main operating system and applications, while the other one hosts trusted applications and responsible for the operations data encryption or decryption. For these two separate environments, they are co-existing in the system. Meanwhile, each one maintains its own software stack. TEE is an isolated execution environment, for any application in the untrusted environment, which we call it a CA, it cannot communicate with the trusted environment without an allocated permission. Namely, TEE does not allow unauthenticated CAs in untrusted environments to access any data in TEE freely, which guarantees the security of the TEE. A TEE architecture can be abstracted as Fig. 1.

2.2 System model

SIBAS aim [illegible] nt approach from Gentry`s scheme [23]. While Tan et al. [20] consider splitting their verification scheme NaEPASC into 5 phases, we choose to divide our scheme into 6 phases (**setup**, **private key generation**, **signature generation**, **challenge**, **signature aggregation** and **verify**). At the same time, there are two different entities involved in SIBAS: cloud service provider (CSP), cloud user (it can be divided into the client application (CA) and a trusted execution environment (TEE)), while Tan`s scheme contains multiple entities. The comparison between the two schemes is shown as Fig. 2, Fig. 3-(a) and Fig. 3-(b) (Fig. 3-(a) contains the first three stages while Fig. 3-(b) includes the last three stages). Next, we describe the responsibilities of each entity and how they interact with each other.

CSP: CSP is responsible for storing the outsourced data of cloud users. We assume that CSP is “honest but curious”, which means it cannot be trusted totally.

CA: CA is responsible for interacting with the CSP. Besides, it can choose to issue a request to checking the integrity of outsourced data.

TEE: TEE is an isolated small-scale operating system. In our scheme, it replaces the PKG server and the trusted audit server to perform key management as well as auditing data integrity. Due to the unique design mechanism of TEE, we can ensure that key management is secure enough.

2.3 Design goal

In this paper, we aim at achieving the following goals from the aforementioned threaten:

Security: Ensuring that no one can obtain the master secret that users hold and the sensitive information storing in TEE.

Efficiency: checking the integrity of the outsourced data with a minimum overhead of computation.

Independence: Without involved the PKG server and the audit server to generating the secret key and verifying the outsourced data, we take TEE to replace them

Note that our scheme aims at verifying the integrity of outsourced data, so we do not take the metadata encryption into account.

3 Model statements

In this section, we present the SIBAS in detail. First, we introduce some useful information about TEE. TEE is an isolated system that enables to task the high level of security work while running in parallel to the ordinary operation system (CA). If the CA attempts to ask for the assistance of TEE, it first authenticates its identity to TEE by a *uuid*, which is the unique identifier for connecting TEE and CA. Without it, the communication will be failed. By such unique mechanism, the potential attackers are unable to obtain the sensitive information from TEE. In our study, we assume that there is a file F waiting for upload. To ensure the integrity while it is outsourcing, we launch a scheme with the following steps.

3.1.1 The basic construction of our scheme

In the first place, we present some related definitions in our proposal. Let G_1, G_2 be the cyclic group of order p, there exist a bilinear group $e: G_1 \times G_1 \to G_2$. We use three cryptographic hash functions $H_1, H_2: \{0,1\}^* \to G_1$, $H_3: \{0,1\}^* \to Z_p$ and

$H_4: \{0,1\}^* \rightarrow$

Setup: TEE generates the parameters and the secret as PKG. It first chooses an arbitrary generator $P \in G_1$, randomly pick a value $s \in Z_p$ to compute $Q = sP$. Constructing the system parameter: $\{G_1, G_2, e, H_1, H_2, H_3, P, Q\}$, and the master secret is $s \in Z_p$.

Private key generation: While CA is communicating with TEE, it first publishes its *uuid* as well as the data file to TEE, so TEE can take CA's *uuid* as input to compute the private key as follows:

For $j \in \{0,1\}$, it computes two hash values where $P_j = H_1(uuid, j)$, $P_j \in G_1$. Then outputs the secret key as $Q_j = sP_j$.

Signature generation: While the cloud user retrieves the outsourced file, the signature needs to be computed. For a file F can be split into n blocks: $F \rightarrow \{b_1, b_2, \cdots, b_{n-1}, b_n\}$, $n \in Z^*$. TEE can generate the signature pair $\{S_i, T_i\}$ of each file block b_i as follows:

(i) Computing two hash value as: $P_\omega = H_2(filename)$, $c_i = H_3(a_i, uuid, filename)$, where a_i is the index of the block b_i in the data file F , $1 \le i \le n$.

(ii) Initializing an instance of Shamir's (t, n) threshold scheme $\mathcal{INS}_{(t,n)}$ with $f(x) = P_\omega + a_1 x + \cdots + a_{t-1} x^{t-1}$ and computes $t-1$ points $pp = \{(v_1, f(v_1)),\ (v_2, f(v_2)),\ \cdots,\ (v_{t-1}, f(v_{t-1})) \mid v_i \in \{0,1\}^*\}$, pp is a public parameter. Then the algorithm computes $v' = H_4(uuid)$, $y = f(v')$, $enc = \frac{v' \parallel f(v')}{Q_0 \parallel Q_1}$.

(iii) Generating n random values as: $r_i \in Z_p$, $1 \le i \le n$, then computes $T_i = r_i P$.

(iv) Computing $S_i = r_i P_\omega + c_i Q_0 + b_i Q_1$, $(1 \le i \le n)$.

(v) TEE packs two values T_i and S_i into a signature $\psi_i = \{S_i, T_i, enc\}$, and then commits ψ_i with the file F to the cloud data center.

Challenge: Before the cloud user aggregates the signature to check the integrity of outsourced file, TEE picks a m elements subset $S_{sub} = \{s_1, s_2, \cdots, s_{m-1}, s_m\}$ in randomly for $S_{sub} \subseteq \{1, 2, \cdots, n-1, n\}$, where there exists a_j equal to an unique $s_i \in S_{sub}$ $(1 \le j \le n, 1 \le i \le m)$. Then, with the corresponding value $x_i \in Z_q$ $(q = p / 2)$ in randomly and sends the value $S_{sub} = \{s_1, s_2, \cdots, s_{m-1}, s_m\}$ and $X_{sub} = \{x_1, x_2, \cdots, x_{m-1}, x_m\}$ to CSP.

Aggregation: Upon CSP receives these two sets, it will search for the corresponding file blocks as $b_{S_{sub}} = \{b_{s_1}, b_{s_2}, \cdots, b_{s_{m-1}}, b_{s_m}\}$ and computes the linear value of the single block as $b_{s_i} x_i \in Z_p$. After that, a proof

$$\{enc, S_m = \sum_{i=1}^{m} x_i S_{s_i}, T_m = \sum_{i=1}^{m} x_i T_{s_i}, R_m = \sum_{i=1}^{m} x_i b_{s_i} \mid S_m, T_m \in G_1, R_m \in Z_p\}$$

is calculated by CSP and send back to the user as a response.

Verify: With the response proof from CSP, CA commits the proof to TEE in a secure way. First, the algorithm decrypts the secret enc as follow:

(i) It extracts $(v^* = v',\ f(v^*) = f(v'))$ from enc by $v' \parallel f(v') = enc(Q_0 \parallel Q_1)$.

(ii) The algorithm reconstructs the polynomial $f(x)$ of Shamir's (t, n) threshold scheme $\mathcal{INS}_{(t,n)}$ through Lagrange polyn … $_{t-1}))\}$ and $(v^*, f(v^*))$.

(iii) The algorithm recovers P_ω by $P_\omega = f(0)$.

Then, TEE can check the integrity of the outsourced data by the following equation:

$$e(S_m, P) = e(T_m, P_\omega)e(Q, \sum_{i=1}^{m} c_i x_i P_0 + R_m P_1) \quad [1]$$

for $P_j = H_1(uuid, j),\ j \in \{0,1\},\ P_\omega = H_2(filename),\ c_i = H_3(a_i, uuid, filename)$.

Remark 1. We give a proof of the correctness of the integrity verification when the equation [1] was established. The proof is listed as follows:

$$\text{Right} = e(T_m, P_\omega)e\left(Q, \sum_{i=1}^{m} c_i x_i P_0 + R_m P_1\right)$$

$$= e(T_m, P_\omega)e\left(Q, \sum_{i=1}^{m} c_i x_i P_0 + \sum_{i=1}^{m} b_i x_i P_1\right)$$

$$= e\left(\sum_{i=1}^{m} x_i r_i P, P_\omega\right)e\left(Q, \sum_{i=1}^{m} c_i x_i P_0 + \sum_{i=1}^{m} b_i x_i P_1\right)$$

$$= e\left(\sum_{i=1}^{m} x_i r_i P, P_\omega\right)e\left(sP, x_i \sum_{i=1}^{m}(c_i P_0 + b_i P_1)\right)$$

$$= e\left(P, \sum_{i=1}^{m} x_i r_i P_\omega\right)e\left(P, s x_i \sum_{i=1}^{m}(c_i P_0 + b_i P_1)\right)$$

$$= e\left(P, \sum_{i=1}^{m} x_i r_i P_\omega + \sum_{i=1}^{m}(c_i x_i s P_0 + b_i x_i s P_1)\right)$$

$$= e\left(P, \sum_{i=1}^{m} x_i(c_i Q_0 + b_i Q_1 + r_i P_\omega)\right),\ Q_0 = sP_0, Q_1 = sP_1$$

$$= e(S_m, P) = \text{Left}$$

Batch aggregation: With the popularity of cloud computing, the way of individual signature no longer meets the need of users. Therefore, we consider the situation of multiple files request for outsourcing concurrently, which can significantly improve the efficiency of integrity checking. Suppose the user attempts to verify K files concurrently, we take a dual aggregation signature scheme, which supports the aggregation of multiple signatures by the user on distinct outsourced files into a single signature. The B-aggregation signature is constructed as follows:

$$\{\sum_{j=1}^{K} enc_j, \sum_{j=1}^{K} S_{m,j} = \sum_{j=1}^{K}\sum_{i=1}^{m} x_i S_{s_i,j}, \sum_{j=1}^{K} T_{m,j} = \sum_{j=1}^{K}\sum_{i=1}^{m} x_i T_{s_i,j}, \sum_{j=1}^{K} R_m = \sum_{j=1}^{K}\sum_{i=1}^{m} x_i b_{s_i,j} \mid S_m, T_m \in G_1, R_m \in Z_p\}$$

Then, it can verify the equation as:

$$e\left(\sum_{j=1} S_{m,j}, P\right) = e\left(\sum_{j=1} T_{m,j}, \sum_{j=1} (P_\omega)_j\right) e\left(Q, \sum_{j=1}\sum_{i=1} c_{i,j} x_i P_0 + \sum_{j=1} R_{m,j} P_1\right) \qquad [2]$$

Remark 2. We give a proof of the correctness of the integrity verification when the equation [2] was established. The proof is listed as follows:

$$\text{Right} = e\left(\sum_{j=1}^{K} T_{m,j}, \sum_{j=1}^{K} (P_\omega)_j\right) e\left(Q, \sum_{i=1}^{m} x_i \cdot \sum_{j=1}^{K} c_{i,j} P_0 + \sum_{j=1}^{K} R_{m,j} P_1\right)$$

$$= e\left(\sum_{j=1}^{K} T_{m,j}, \sum_{j=1}^{K} (P_\omega)_j\right) e\left(Q, \sum_{i=1}^{m} \left(x_i \cdot \sum_{j=1}^{K} c_{i,j} P_0\right) + \sum_{j=1}^{K} b_{i,j} x_i P_1\right)$$

$$= e\left(\sum_{j=1}^{K}\sum_{i=1}^{m} x_i r_{i,j} P, \sum_{j=1}^{K} (P_\omega)_j\right) e\left(Q, \sum_{i=1}^{m} \left(x_i \cdot \sum_{j=1}^{K} c_{i,j} P_0\right) + \sum_{j=1}^{K} b_{i,j} x_i P_1\right)$$

$$= e\left(\sum_{j=1}^{K}\sum_{i=1}^{m} x_i r_{i,j} P, \sum_{j=1}^{K} (P_\omega)_j\right) e\left(sP, \sum_{j=1}^{K} \left(\sum_{i=1}^{m} c_{i,j} x_i P_0\right)_j + \sum_{j=1}^{K} b_{i,j} x_i P_1\right)$$

$$= e\left(P, \sum_{j=1}^{K}\sum_{i=1}^{m} x_i r_{i,j} (P_\omega)_j\right) e\left(P, sx_i \sum_{j=1}^{K}\sum_{i=1}^{m} (c_{i,j} P_0 + b_{i,j} P_1)\right)$$

$$= e\left(P, \sum_{j=1}^{K}\sum_{i=1}^{m} x_i r_{i,j} (P_\omega)_j + \sum_{j=1}^{K}\sum_{i=1}^{m} x_i (sc_{i,j} P_0 + sb_{i,j} P_1)\right)$$

$$= e\left(P, \sum_{j=1}^{K}\sum_{i=1}^{m} x_i \left(c_{i,j} Q_0 + b_{i,j} Q_1 + r_{i,j} (P_\omega)_j\right)\right), Q_0 = sP_0, Q_1 = sP_1$$

$$= e\left(\sum_{j=1}^{K} S_{m,j}, P\right) = \text{Left}$$

Therefore, SIBAS can verify the integrity of the outsourced file efficiently. However, while the user takes the batch aggregation scheme to check the integrity of multiple files at once, if the integrity of anyone in these files is compromised, then the output of file verification is a verification failure. In this case, we cannot detect which one is corrupted. Therefore, it may be convenient to use batch aggregation if the number of files is large, but it can also make the problem more troublesome if the accident occurs.

4 Security analysis

According to the above assumption, we consider the following adversaries in our integrity checking scheme: (1) an adversary who lurks in local side and attempts to obtain some privacy from CA. This kind of adversary may contain Trojans, malware and implantation … some useful information of the outsourced data in the cloud. In this type of attackers, we consider the attack model of revoked users and unauthorized users, who may apply for the sensitive information as the legal users. In this section, we will put forward a proof to show that our scheme is secure enough to resist the attack from the second one. Namely, there are not attackers can forge a correct signature to cheat TEE and output "true" only in the case that the attackers hold a correct user ID. After that, we apply the closedness of TEE to prove that our solution can defend against the first kind of adversaries.

Definition 1. Supposing that there is an adversary $\mathcal{A}$ can make q_E adaptive key extraction queries, q_S adaptive signature queries and q_H hash queries and forge an aggregation signature by the advantage ε over time t .We say that there exists an adversary $\mathcal{A}-(\varepsilon, \mathrm{t}, q_{H1}, q_{H2}, q_{H3}, q_E, q_S)$ is capable of breaking our scheme. Else, we say that our scheme is security and the signature is unforgeable.

Theorem 1. If the CDH problem is difficult to solve in bilinear group G_1, then our scheme is impossible to be broken by any adversary unless it can respond with the correct aggregation signature.

Proof: Assume that adversary $\mathcal{A}-(\varepsilon, \mathrm{t}, q_{H1}, q_{H2}, q_{H3}, q_E, q_S)$ can break our scheme, there is an algorithm $\mathcal{B}$ can solve the computational Diffie-Hellman (CDH) problem by interacting with the adversary $\mathcal{A}$. During the interaction, $\mathcal{B}$ must respond correctly to $\mathcal{A}$ to break our scheme or abort. Next, we describe how $\mathcal{B}$ can solve the CDH problem.

Given $X = xP \in G_1$, $Y = yP \in G_1$. The goal of $\mathcal{B}$ is output $XY = xyP$. Let $\mathcal{B}$ arbitrarily interacts with adversary $\mathcal{A}-(\varepsilon, \mathrm{t}, q_{H1}, q_{H2}, q_{H3}, q_E, q_S)$ as follows:

Setup: The algorithm $\mathcal{B}$ sets the public key as $X = xP$, and then transmits the key to the adversary $\mathcal{A}$. Now, $\mathcal{A}$ can make the hash queries from the random oracles (H_1, H_2, H_3) which are controlled by $\mathcal{B}$.

Hash Queries: $\mathcal{A}$ is allowed to make H_1-query, H_2-query, H_3-query at any time. Whenever $\mathcal{A}$ initiate its query, $\mathcal{B}$ must make a unique response to $\mathcal{A}$`s query.

Query on oracle H_1: In this phase, $\mathcal{B}$ maintains a list L_1 of tuples $\langle ID,\ t_0,\ t_1\rangle$ to respond to the query of H_1 oracle. While $\mathcal{A}$ submits its *ID* to H_1, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

(1) If the *ID* already exists, $\mathcal{B}$ searches list L_1 and recovers the value H_1, H_2 from L_1.

(2) Otherwise, $\mathcal{B}$ generates random values $t_0, t_1 \in Z_p$, and responds to $\mathcal{A}$ with $H_1(ID, j) = t_j yP$ for $j \in \{0,1\}$.

Query on oracle H_2: In order to ensure consistency, $\mathcal{B}$ also maintains with list L_2 of tuple $\langle filename, \lambda\rangle$ to responds to the query on oracle H_2. While $\mathcal{A}$ initiates its query to H_2, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

(1) If the filename already exists, $\mathcal{B}$ recovers λ from the L_2.

(2) Otherwise, $\mathcal{B}$ generates a random value $\lambda \in Z_p$, and log it with the filename to the dual tuple $\langle filename,\ \lambda\rangle$.

(3) $\mathcal{B}$ responds to $\mathcal{A}$ with $H_2(filename) = \lambda P$.

Query on oracle H_3: In this query, $\mathcal{B}$ maintains a list L_3 of tuples $\langle ID,\ filename, a_i, h_i\rangle$. When $\mathcal{A}$ initiates its query and submits a tuple $\langle ID,\ filename, a_{\mathcal{A}}\rangle$ to H_3, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

(1) If the tu

(2) Otherwise, $\mathcal{B}$ generates a random value $h_i \in Z_p$, and log it with the tuple $\langle ID,\ filename, a_i\rangle$ to L_3.

(3) $\mathcal{B}$ responds to $\mathcal{A}$ with $H_3(ID, filename, a_i) = h_i$.

Extraction queries: When $\mathcal{A}$ requests for the private key that bind with the unique ID, $\mathcal{B}$ searches for the corresponding tuple $\langle ID,\ t_0,\ t_1\rangle$ from the list H_1. If the ID does not exist, $\mathcal{B}$ outputs "failure" and halts. Otherwise, the values $\langle t_0 bX, t_1 bX\rangle$ is set as private key and return to $\mathcal{A}$.

Signature queries: When $\mathcal{A}$ queries for the single signature of a file block b_i, $\mathcal{B}$ first checks whether $\mathcal{A}$ has initiated such a query or not. If not, $\mathcal{B}$ allows to generate the signature of the block b_i by the private key $t_j bP$ for j $\in \{0,1\}$. It first generates two random values $r_{\mathcal{A}}, h_{\mathcal{A}} \in Z_p$. Then, $\mathcal{B}$ responds as follows:

(1) If the tuple $\langle ID,\ filename, a_{\mathcal{A}}\rangle$ exists. However, $h_{\mathcal{A}} \neq h_i$. $\mathcal{B}$ outputs "failure" and halts.

(2) If the tuple $\langle ID,\ filename, a_{\mathcal{A}}\rangle$ exists and $h_{\mathcal{A}} = h_i$. $\mathcal{B}$ computes the signature $(\grave{S}_i, \grave{T}_i)$ as $\grave{S}_i = h_i aP_0 + m_i aP_1 + r_i P_\omega$ and $\grave{T}_i = r_i P$ for $P_0 = t_0 bP,\ P_1 = t_1 bP$ and $P_\omega = \alpha P$.

(3) If the tuple $\langle ID,\ filename, a_{\mathcal{A}}\rangle$ does not exist in the list L_3. $\mathcal{B}$ computes the signature $(\grave{S}_i, \grave{T}_i)$ as $\grave{S}_i = h_i aP_0 + m_i aP_1 + r_i P_\omega$ and $\grave{T}_i = r_i P$ for $P_0 = t_0 bP,\ P_1 = t_1 bP$ and $P_\omega = \alpha P$.

Output: If $\mathcal{A}$ is successful and proceeding to forge the signature as $\{\grave{S}_m, \grave{T}_m, \grave{Cha}\}$ for $\grave{Cha} = \{\grave{s}_i, \grave{x}_i\}$. Obviously, $\{\grave{S}_m, \grave{T}_m, \grave{Cha}\}$ is satisfied the verification equation [1] as follows:

$$e(\grave{S}_m, P) = e(\grave{T}_m, P_\omega)e(X, \sum_{i=1}^{m} h_i \grave{x}_i P_0 + \sum_{i=1}^{m} \grave{b}_i \grave{x}_i P_1) \tag{3}$$

Additionally, $\mathcal{B}$ holds the correct signature $\{S_m, T_m, Cha\}$ from an honest prover, which is satisfied the following equation

$$e(S_m, P) = e(T_m, P_\omega)e(Q, \sum_{i=1}^{m} c_i x_i P_0 + \sum_{i=1}^{m} b_i x_i P_1) \tag{4}$$

If $h_i = c_i$, and $X = xP$ is the public key that equal to Q. Then we construct another equation that dividing equation [3] by equation [4]. We can get the following equation:

$$e(S_m - \grave{S}_m, P) = e(T_m - \grave{T}_m, P_\omega)e(Q, \sum_{i=1}^{m} c_i x_i P_0 + \sum_{i=1}^{m} (b_i x_i - \grave{b}_i \grave{x}_i) P_1)$$

$$\Rightarrow e(S_m - \grave{S}_m, P) = e(T_m - \grave{T}_m, \lambda P)e(xP, \sum_{i=1}^{m} c_i x_i t_0 yP + \sum_{i=1}^{m} (b_i x_i - \grave{b}_i \grave{x}_i) t_1 yP)$$

$$\Rightarrow e(S_m - \grave{S}_m, P) = e(\lambda(T_m - \grave{T}_m), P)e(P, xyP \sum_{i=1}^{m} (c_i x_i t_0 + (b_i x_i - \grave{b}_i \grave{x}_i) t_1))$$

$$\Rightarrow S_m - \grave{S}_m = \lambda(T_m - \grave{T}_m) + xyP \sum_{i=1}^{m} (c_i x_i t_0 + (b_i x_i - \grave{b}_i \grave{x}_i) t_1)$$

$$\Rightarrow xyP = \frac{(S_m - \grave{S}_m) - \lambda(T_m - \grave{T}_m)}{\sum_{i=1}^{m}(c_i x_i t_0 + (b_i x_i - \grave{b}_i \grave{x}_i) t_1)}$$

Thence, we say that xyP is solvable and the CDH problem can be solved by the algorithm $\mathcal{B}$.

Next, we explain the security of our integrity verification that running in TEE.

Access to security management. We take TEE as a trusted execution environment to perform security management, TEE is a trusted computing platform which can be regarded as a black box. In other words, it is invisible to the malicious software or virus that how TEE checks the integrity of outsourcing files. If the adversary attacks the client in the local device, the sensitive data that send from TEE to CA are encapsulated. Therefore, the adversary cannot obtain valuable data through the trusted interface.

Advantage of key storage. In this scheme, it is our advantage to ensure the confidentiality of secret keys while committing them to the cloud. The outsourced keys are divided into two parts, one is derived from the *uuid* of user, and the other one is calculated by the filename of outsourced files. Since that the *uuid* is transmitted over a trusted side-channel between TEE and CA, there are no adversaries can threaten the reliability of it. Next, we prove that how Shamir's (t, n) threshold scheme works to protect the confidentiality of P_ω.

Theorem 2. If the adversary cannot catch the value of $(v^*, f(v^*))$, it is unable to reconstruct $f(x)$ with non-negligible probabilities and then get the value of P_ω.

Proof. We consider the following two situations:

Question 1: If the adversary knows $pp = \{(v_1, f(v_1)), (v_2, f(v_2)), \cdots, (v_{t-1}, f(v_{t-1}))\}$ yet has no other information about $(v^*, f(v^*))$, whether it can reconstruct $f(x)$ through the information that it holds?

Answer 1: Suppose an adversary attempts to reconstruct $f(x)$ through $\{(v_1, f(v_1)), (v_2, f(v_2)), \cdots, (v_{t-1}, f(v_{t-1}))\}$. According to the formula $P_\omega = f(0) = \sum_{i=1}^{t} f(v_i) \prod_{\substack{j=1 \\ j \neq i}}^{t} \frac{v - v_j}{v_i - v_j} \ (mod\ q)$, we can construct the following equations:

$$\begin{cases} s_1 = f(v_1) = \sum_{i=0}^{t-1} a_1 v_1^i \ (mod\ q) \\ s_2 = f(v_2) = \sum_{i=0}^{t-1} a_i v_2^i \ (mod\ q) \\ \vdots \\ s_{t-1} = f(v_{t-1}) = \sum_{i=0}^{t-1} a_i v_{t-1}^i \ (mod\ q) \end{cases}$$

Which can convert to a matrix group as:

$$XA = S \quad \Rightarrow \quad \begin{bmatrix} 1 & v_1 & \cdots & v_1^{t-1} \\ 1 & v_2 & \cdots & v_2^{t-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & v_{t-1} & \cdots & v_{t-1}^{t-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{t-1} \end{bmatrix}$$

Obviously, the rank of matrix X is $r(X) \le (t-1)$. Therefore, it is impossible to get the value of P_ω by $\{(v_1, f(v_1)),$

$(v_2, f(v_2))$,

Question 2: What is the probability that an attacker would model $(v_t, f(v_t))$ to reconstruct $f(x)$?

Answer 2: Suppose the adversary mimics a point $(v_t, f(v_t))$. Then it can construct $f'(x) = s' + a'_1 x + \cdots + a'_{t-1} x^{t-1}$ by $pp = \{(v_1, f(v_1)),\ (v_2, f(v_2)),\ \cdots,\ (v_{t-1}, f(v_{t-1}))\}$ and $(v_t, f(v_t))$. It computes $P'_\omega = f'(0)$. Obviously, the probability of $P_\omega = P'_\omega$ is $1/q$, which means that the adversary cannot forge the value of P_ω with non-negligible probabilities.

5 Performance Evaluation

In this section, we assess the performance of the proposed integrity verification scheme. In order to show the feasibility and efficiency of our scheme, computation cost and the computation overhead are the main considerations for the experiment. The experiment is conducted using C on an Ubuntu 14.04 with an Intel Core 4 processor running at 2.60 GHz and 4096 MB of RAM as the client, while a 7200 RPM Western Digital 1 TB Serial ATA drive with an 8 MB buffer for the server. Our algorithm uses the Pairing-Based Cryptography (PBC) library version 0.5.14 and the OpenSSL version 1.0.2n for programming. Moreover, the elliptic curve which we apply is an MNT curve, with a base field size of 159 bits and an embedding degree of 6. All of our experimental data are the result of averaging over 50 trials.

5.1 Computation cost and communication overhead

First, we estimate the computation cost and the communication overhead of our scheme. In table I, we list the cost of calculation operations and basic cryptographic operations. Assuming that a file F is divided into c blocks as the experimental sample, the computation cost of our scheme is almost as same as the NaEPASC scheme on the server side. According to the calculation formula in section 3, the response proof $\{S_m, T_m, R_m, enc\}$ in challenge phase is the whole computation cost, it is quite obvious that the cost of S_m is equal to T_m as $\frac{c}{p}$-$\text{AddMult}^2_{G_1}(m)$, and the R_m`s cost is $\text{Add}^{c-1}_{Z_p} + \text{Mult}^{c}_{Z_p}$. Therefore, a total cost can be denoted as c-$\text{AddMult}^2_{G_1}(m) + \text{Add}^{c-1}_{Z_p} + \text{Mult}^{c}_{Z_p} + INS^t_{\{0,1\}^*}$, the corresponding communication is $c(|n| + |p|/2) + 3|p| + 1$ (for a set n, $|n|$ denotes the number of elements in n) in the whole verification procedure.

Table 1. Notation of cryptographic operations

$Hash^t_{G_1}$	hash t value into the group G_1
$Add^t_{G_1}$	*t* additions in the group G_1
$Mult^t_{G_1}$	*t* multiplications in the group G_1
r-$\text{AddMult}^t_{G_1}(\lvert a_i \rvert)$	*t* *r*- term multiplications $\sum_{i=1}^{r} a_i P$, $\lvert a_i \rvert$ denotes the number of elements in the set a_i
$Pair^t_{G_1,G_1}$	*t* pairings $e(U,V)$, where U and V is belong to G_1
$INS^t_{\{0,1\}^*}$	*t* points of interpolation calculation

5.2 Auditing efficiency and Comparison results

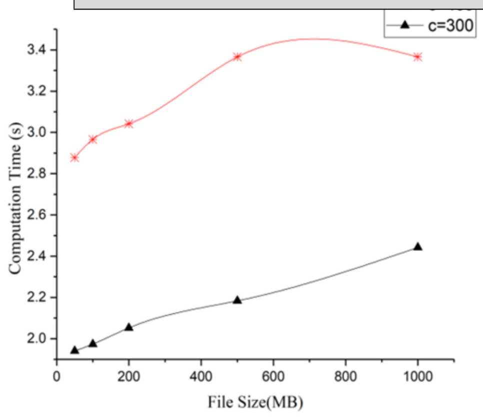

Fig. 4. Individual signatures computation time for TEE with file size as

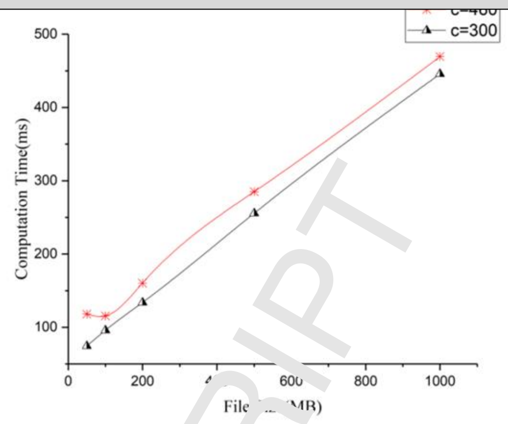

Fig. 5. Individual signatures computation time for cloud server with file size as 50 MB, 100 MB, 200MB, 500MB and 1000MB and $c = 300$ or 460.

In table 2, we compare our scheme with Wang`s [5] and Tan`s [20] in communication overhead and computation time. In their schemes, they take a file with the size of 1 GB and split it into c blocks as the experimental sample. According to Ateniese et al. [11], we know that it is more likely to detect the misbehavior when $c = 300$ or $c = 460$. Our experiment results show that the communication overhead is not much different from the two schemes. Furthermore, since that our auditing time contains the extra interacting time between TEE and CA, we can observe that our auditing time is longer than that of Tan`s yet it also performs better than Wang`s scheme. Considering the server computation time, our scheme is more rapid than the two schemes while the cloud server computes the response proof. Moreover, with the increasing of the sample blocks from 300 to 460, Wang`s scheme increases more than 300ms in both TPA computation time and server computation time, Tan`s also increases more than 600ms in server computation time, while the increasing time is less than 100ms in our scheme.

Table 2. Performance comparison between three different schemes

Three different schemes	Communication overhead (KB)		TPA computation time (ms)		server computation time (ms)	
	$c = 300$	$c = 460$	$c = 300$	$c = 460$	$c = 300$	$c = 460$
Wang`s scheme	*4.24*	*6.43*	*639.0*	*968.5*	*639.8*	*975.3*
NaEPASC	*4.16*	*6.34*	*35.2*	*40.6*	*1240.5*	*1902.6*
SIBAS	*4.20*	*6.40*	*62.0*	*77.4*	*445.7*	*469.4*

5.3 Computation cost of individual signature

To evaluate the effect of different file size on computation cost, we take 5 files in the size of 50MB, 100MB, 200MB, 500MB, and 1GB, and let the sample blocks as $c = 300$ and $c = 460$ for comparing, the experiment results are plotted as Fig. 4, Fig. 5, Fig. 6.

In Fig. 4, we show the computation time when the user generates the individual signatures in different file size. With the increasing of the file size, the computation time holds an approximately linear growth in two different cases. For $c = 300$, the computation r $c = 460$. Obviously, while $c = 460$, it takes about 1000ms more time than that $c = 300$ in the same file size. A total time of signatures computation includes the duration of the file slicing, the time of data-blocks signing and encrypting the secret key with Shamir's (t, n) threshold scheme. Then, we calculate the data-blocks signing and encrypt the secret, we find that the time cost between them is in a minimal difference. In other words, due to the difference of time cost in file slicing, it leads to a great different of time cost between them. In Fig. 5, the computation time of the cloud server is shown. For both two cases, the time costs are growing in a linear way. Furthermore, we can observe that the time disparity always less than 30ms between the two cases for the same file size. In Fig. 6, we compare the verification time under two cases. Because the verification time is independent of the file size, we find that the time cost is fluctuating around 60ms, but it is also within the margin of error. In both $c = 300$ and $c = 460$, the time cost is always stable in the range of 60ms.

5.4 Computation cost of batch Aggregation

In the case of large number of files, it is cumbersome to check the integrity of the files one by one. Therefore, we propose the batch aggregation scheme to check the integrity of multiple files concurrently. Compared with individual signature, batch aggregation requires for more operation of multiplications ($4K$ multiplications for multiple signatures aggregation). However, it reduces the operation of multiplications, which significantly improves the efficiency of data integrity checking. Following an experimental that sets file size as 500MB and $c = 300 \mid 460$, the average of per signature computing time which is obtained by dividing the total time cost by the number of files, is given in Fig. 7. In this experiment, the number of files is increased from 1 to 200 with intervals of 10. It can be shown that the computation cost of signing in the cloud is reduced for both $c = 300$ and $c = 460$. It cost about 255ms for individual signature while $c = 300$ and 285ms while $c = 460$. However, with the batch aggregation, the average time cost on each file drops to 90ms and 65ms, respectively.

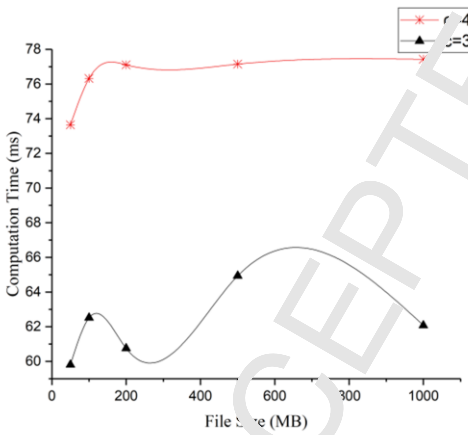

Fig. 6. Computation time of TEE to verify the signature with file size as 50

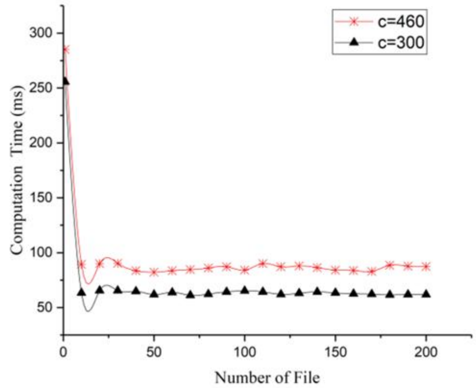

Fig. 7. Time computation of batch aggregation with file size as 500MB and

6 Related Works

In some research work, proof of retrievability (POR) was proposed to ensure the possession and retrievability of the data on remote storage nodes through spot-checking and error-correcting codes [9]. Ateniese et al. [11] proposed a model for provable data possession (PDP), which aimed to allow users who have stored data at an untrusted server to verify the original data without retrieving it. Without considering the data dynamic storage, they utilized RSA-based homomorphic tags for auditing outsourced data. Later, t pport public auditability.

Wang et al. [5,13] proposed the privacy-preserving public auditing protocol, which first achieved both public verifiability and dynamic data storage operations by manipulating the classic Merkle Hash Tree (MHT) [14] construction for block tag authentication. Then they introduced another scheme that utilizes the public key based homomorphic authenticator with random masking to implement the privacy-preserving public auditing [4]. However, it may lead to the disclosure of documents because the third-party auditor can obtain the private message of users easily. Wassim Itani et al. [15] introduced an energy-efficient protocol to ensure the integrity of storage services in mobile cloud computing, which utilizes a coprocessor to allocate an encryption key for mobile client, which can generate a message authentication code (MAC) [10] storing in local and update the MAC that used to implement integrity verification while client applied for the outsource data in the cloud. Kan Yang et al. [16] presented data access control for multiauthority cloud storage (DAC-MACS), a scheme that based on Ciphertext-policy attribute-based encryption (CP-ABE) [6] to apply effective data access control for multiauthority cloud storage systems. However, the analysis and investigation by J Hong et al. [7] show that there is a security vulnerability because a revoked user can still decrypt a new ciphertext which seems that only can be decrypted by the new-version secret keys.

In other related works, Wang et al. [25] first present the ID-based public auditing protocol, which was proved to be secure under the assuming the hardness of the computational Diffie–Hellman problem. Then, Tian et al. [20] constructed another data auditing scheme based on identity-based aggregate signatures. Li et al. [26] proposed a revocable IBE scheme that first introducing outsourcing computation into IBE to tackle the issue of identity revocation. Recently, Li and Yu et al. [27] introduced fuzzy identity-based auditing by utilizing biometrics as the fuzzy identity to achieve the goal of efficient key management. Yu et al. [8] proposed the protocol ID-CDIC, which is based on the user`s identity to eliminate the complex certificate management.

7 Conclusions

In this paper, we propose a scheme called SIBAS to verify the integrity of the outsourced data. In our scheme, we resort TEE to play the role of an auditor to check the correctness of the aggregate signature. Because of the closeness of TEE, it reduces the probability of key leakage and the cloud users do not have to fear that their secret information is embezzled by the other attackers. Furthermore, the extensive performance analysis and experiments are conducted, and the results show that it is feasible and efficient for our scheme while checking the data integrity of the outsourced data.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by CERNET Innovation Project-Research on Key Technologies of Data Security Access Control Mechanism Based on IPv6 (No. NGII20180406), by Beijing Higher Education Young Elite Teacher Project (No.YETP0683), by Beijing Higher Education Teacher Project (No. 00001149).

Reference

1. Mell T., Grance P.: Draft NIST Working Definition of Cloud Computing. Referenced on 53(6), 50-50 (2009).

2. Liang W., Xie Y., Xiao W., Chen X.: A Two-step MF Signal Acquisition Method for Wireless Underground Sensor Networks. Computer Science and Information Systems 13(2), 623-638(2016).

3. Liang W., Huang Y., Xu J., Xie S.: A Distributed Data Secure Transmission Scheme in Wireless Sensor Network. International Journal of Distributed Sensor Networks 12(4), 1-11(2017).

4. Kamara S., Lauter K.: Cryptographic cloud storage. In: International Conference on Financial Cryptograpy and Data Security. Springer-Verlag, pp.136-149 (2010).

5. Wang C., Wang Q., Ren K.: Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In: Proc. of IEEE INFOCOM'1, pp. 525-533. IEEE, San Dieg (2010).

6. Bethencourt J., Sahai A., Waters B.: Ciphertext-Policy Attribute-Based Encryption. In: IEEE Symposium on Security and Privacy, pp.321-334. IEEE Computer Society, Oakland (2007).

7. Hong J., Xue K., Li W.: Comments on "DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems"/Security Analysis of Attribute Revocation in Multiauthority Data Access Control for Cloud Storage Systems. IEEE Transactions on Information Forensics & Security 10(6),1315-1317 (2017).

8. Yu, Y., Xue, L., Man, H. A., Susilo, W., Ni, J., & Zhang, Y., et al.: Cloud data integrity checking with an identity-based auditing mechanism from RSA. Future Generation Computer Systems 52(C), 85-91 (2016).

9. Juels A.: Pors: proofs of retrievability for large files. In: ACM Conference on Computer and Communications Security, pp. 584-597. ACM, Alexandria (2007).

10. Bellare M, Ran C, Krawczyk H.: Message Authentication using Hash Functions--- The HMAC Construction. Cryptobytes, 2 (1996,).

11. Ateniese G, Burns R, Curtmola R.: Provable data possession at untrusted stores. In: ACM Conference on Computer and Communications Security. pp. 598-609. ACM, Alexandria (2007).

12. Ateniese G., Burns R., Curtmola R.: Remote data checking using provable data possession. Acm Transactions on Information & System Security 14(1), 12-12 (2011).

13. Wang Q., Wang C., Ren K.: Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. IEEE Transactions on Parallel & Distributed Systems 22(5), 847-859 (2011).

14. Merkle R C.: Protocols for Public Key Cryptosystems. IEEE Symposium on Security & Privacy (3), 122-122 (1980).

15. Itani W, Kayssi A, Chehab A.: Energy-efficient incremental integrity for securing storage in mobile cloud computing. In: International Conference on Energy Aware Computing. pp.1-2. IEEE, Cairo (2010).

16. Yang K, Jia X, Ren K. DAC-MACS: Effective data access control for multi-authority cloud storage systems. In: INFOCOM, 2013 Proceedings IEEE. pp. 2895-2903. IEEE, Turin (2013).

17. Global Platform: The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market. Global Platform white paper, pp. 1-26 (2011).

18. Shamir A.: Identity-Based Cryptosystems and Signature Schemes. Lect.notes Comput.sci 196(2), 47-53 (1985).

19. Liu H., Z 2), 373-380 (2013).

20. Tan S., Jia Y.: NaEPASC: a novel and efficient public auditing scheme for cloud data. Frontiers of Information Technology & Electronic Engineering 15(9), 794-804 (2014).

21. Kumar P S., Subramanian R.: RSA-based dynamic public audit service for integrity verification of data storage in cloud computing using Sobol sequence. IEEE Wireless Communications Letters 3(3), 289-292 (2012).

22. Boneh D., Franklin M.: Identity based encryption from the Weil pairing. Crypto 32(3), 213-229 (2001).

23. Gentry C., Ramzan Z.: Identity-Based aggregate signatures. In: Moti Y., International Conference on Theory and Practice of Public-Key Cryptography. LNCS, Vol.3958, pp.257-273. Springer-Verlag (2006).

24. Shacham, H., Waters, B.: Compact proofs of retrievability. In: International Conference on the Theory & Application of Cryptology & Information Security, LNCS. Vol.26, pp.90-107. Springer, Berlin, Heidelberg (2008).

25. H. Wang, J. Domingo-Ferrer, Q. Wu, and B. Qin: Identity-based remote data possession checking in public clouds. IET Information Security 8(2), pp. 114–121 (2014).

26. Li J., Li J., Chen X.: Identity-Based Encryption with Outsourced Revocation in Cloud Computing. IEEE Transactions on Computers 64(2) 425-437 (2015).

27. Li Y., Yu Y., Min G.: Fuzzy Identity-Based Data Integrity Auditing for Reliable Cloud Storage Systems. IEEE Transactions on Dependable & Secure Computing, pp. (99):1-1 (2017).

28. Fan Y, Liu S, Tan G, et al. Fine-grained access control based on Trusted Execution Environment[J]. Future Generation Computer Systems, 2018, In press. https://doi.org/10.1016/j.future.2018.05.062.

29. Hu F, Qiu M, Li J, et al. A review on cloud computing: Design challenges in architecture and security[J]. Journal of computing and information technology, pp. 19(1): 25-55 (2011).

Yongkai Fan received the Bachelor, Master And Ph.D. degrees from Jilin University, Changchun, China, in 2001, 2003, 2006, respectively. From 2006 to 2009, he was a assistant researcher in Tsinghua University, Beijing. His current appointment is an assistant professor in China University of Petroleum (Beijing) since 2010. His current research interests include theories of software engineering and software security.

Xiaodong Lin has received a bachelor's degree in Information and Computing Science from China University of Petroleum (East China), Qingdao, China, in 2016.And now is applying for master degree of Computer Science and Technology in China University of Petroleum (Beijing). His current research interests include theories of software engineering and software security.

Gang Tan Received his B.E. in Computer Science from Tsinghua University in 1999, and his Ph.D. in Computer Science from Princeton University in 2005. He is an Associate Professor in Penn State University, University Park, USA. He was a recipient of an NSF Career award and won James F. Will Career Development Professorship. He leads the Security of Software (SOS) lab at Penn State. He is interested in methodologies that help create reliable and secure software systems.

Yuqing Zhang is a professor and supervisor of Ph.D. students of Graduate University of Chinese Academy of Sciences. He received his B.S. and M.S. degree in computer science from Xidian University, China, in 1987 and 1990 respectively. He received his Ph.D degree in Cryptography from Xidian University in 2000. His research interests include cryptography, wireless security and trust management.

WeiDong is currently an associate professor in Department of Electronic Engineering, Tsinghua University, China. He received his Ph.D degree from Tsinghua University, China, in 2006, and received his bachelor degree from Lanzhou University, China, in 2000, respectively. His research interests include energy-efficient integrated perception systems for intelligent robots, and algorithm/hardware co-design for moving robots.

Jing Lei has received a bachelor's degree in software engineering from Shanxi Agricultural University , in 2017. And now is applying for master degree of Computer Science and Technology in China University of Petroleum (Beijing). Her current research interests include machine learning and Information Safety .