Accepted Manuscript

Title: Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing

Author: Yuchuan Luo, Ming Xu, Kai Huang, Dongsheng Wang, Shaojing Fu

PII: S0167-4048(17)30266-3
DOI: https://doi.org/10.1016/j.cose.2017.12.004
Reference: COSE 1252

To appear in: *Computers & Security*

Received date: 9-10-2017
Revised date: 30-11-2017
Accepted date: 6-12-2017

Please cite this article as: Yuchuan Luo, Ming Xu, Kai Huang, Dongsheng Wang, Shaojing Fu, Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing, *Computers & Security* (2017), https://doi.org/10.1016/j.cose.2017.12.004.

Efficient Auditing for Shared Data in the Cloud with Secure User Revocation and Computations Outsourcing

Yuchuan Luo[a], Ming Xu[a,*], Kai Huang[a,b], Dongsheng Wang[a], Shaojing Fu[a,b]

[a]*College of Computer, National University of Defense Technology, Changsha, China*

[b]*State Key Laboratory of Cryptography, Beijing, China*

Biographical sketch

Yuchuan Luo received the B.S. degree and M.S. degree from National University of Defense Technology (NUDT) in 2009, 2015 respectively, both in Computer Science and Technology. He is currently working towards the Ph.D degree in Computer Science and Technology in NUDT. He is currently studying at the City University of Hong Kong for a year. His research interests include Cloud Computing Security, Crowdsensing Security and Privacy.

Ming Xu is currently a professor and director of the Network Engineering Department in the College of Computer, National University of Defense Technology, China. Prof. Xu is a senior member of CCF and member of IEEE and ACM. His major research interests include mobile computing, wireless network, cloud computing and network security.

Kai Huang received the B.S. degree in Network Engineering and the M.S. degree in Cryptography from Naval University of Engineering, China, in 2007 and 2009, respectively. Currently, he is working towards the Ph.D. degree in the College of Computer at National University of Defense Technology. His research interests are in the areas of Cloud Computing security.

Dongsheng Wang received the B.S. degree from Tsinghua University, Beijing, China and the M.S. degree from National University of Defense Technology, Changsha China, in 2009 and 2012, respectively. He is currently a Ph.D. candidate at National University of Defense Technology, Changsha, China. His research interests focus on network and cloud computing security.

Shaojing Fu received his Ph.D. degree in applied cryptography from National University of Defense Technology in 2012 and has studied at University of Tokyo for one year during his Ph.D. He is currently a lecturer in the College of Computer, National University of Defense Technology. His research interests include cryptography theory and application, cloud storage security.

Abstract

With the cloud storage services, users can easily form a group and share data with each other. Considering the cloud is untrusted, public auditing is needed to ensure the integrity of the shared data. Once a user is revoked from the group, signatures from this revoked users need to be re-computed by an existing user, which may incurs heavy communication and computation cost. Proxy re-signatures can be used here to allow the cloud to compute re-signatures on behalf of the group. However, a malicious cloud is able to arbitrarily convert signatures from one user to

*Corresponding author (e-mail: mingxu@nudt.edu.cn)

another using the re-signing keys. Moreover, collusion between revoked users and malicious clouds will disclose the secret values of the remaining users. In this paper, we propose a novel public auditing scheme for the integrity of shared data with efficient collusion-resistant user revocation. In addition, we extend the proposed scheme to support securely signature and verification outsourcing, which allow more efficiency for group users and the auditor. The numerical analyses and experimental results demonstrate that our scheme is provably secure and highly efficient, the outsourcing algorithms make the signatures generation and verification process more efficient and affordable for mobile devices.

Keywords: Cloud Storage, Public Auditing, User Revocation, Collusion-resistant, Outsourcing, Mobile environment.

1. Introduction

Cloud storage is one of the most crucial services of cloud computing, which provides users with elastic storage space. Users are allowed to modify and share their outsourced data in the cloud anywhere and anytime [1]. However, data security has become a critical problem that prevents this new paradigm of data hosting service from widespread adoptions. One of the biggest concerns of the data security is the integrity of the outsourced data in the cloud. Although the cloud storage providers commit a reliable and secure storage service to users, the integrity of outsourced can still be corrupted due to carelessness of humans or failures of hardware/software [2, 3]. Besides internal threats, external adversaries may also destroy the integrity of the outsourced data in the cloud. Therefore, public integrity auditing is needed to convince the users that the outsourced data is correctly stored in the cloud.

To ensure the integrity of outsourced data in an untrustable cloud, a number of protocols have been proposed based on various techniques [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. In all these protocols, the data owner computes a signature on each block of the data and outsources the data together with the corresponding signatures to the cloud, which allows not only the data owner, but also a third party auditor (TPA) to check the integrity of data in the cloud without downloading the entire outsourced data. However, most of the previous works only consider the case of *personal data,* which means that there is only one modifier, namely the data owner, who holds the secret key and can modify the data.

Different from these works, Wang *et al.* [12, 13, 14, 15] focus on how to audit the integrity of the *shared data*. In this scenario, users can easily modify and share data as a group

with the cloud services, every user in the group is able to not only access and modify the shared data, but also share the version that he/she has modified with the rest of the group. In Ref. [12] Wang *et al.* proposed *Qurta*, a public auditing scheme for shared data in the cloud utilizing ring signature-based homomorphic authenticators, which preserves identity privacy of group users from the TPA during auditing. However, this scheme suffers from an inefficient user revocation and the size of the signature is linear to the group size. To overcome the shortcomings of their previous work, Wang *et al.* [13] utilized the advantages of group signatures and proposed *Knox*, which can preserve the identity privacy of users from the TPA. Besides, the size of the verification information in *Knox*, as well as the time it takes to check the verification information, are independent from the group size. However, *Knox* needs users to share a secret value with the TPA and the user revocation of it is also costly. Recently, Wang *et al.* [15] improved their public auditing scheme to support efficient user revocation using proxy re-signatures, which allows the cloud to convert the signatures that were previously computed by a revoked user into signatures of an existing user in the group. However, the cloud in [15] needs to know all the re-signing keys between any two users of the group in advance since the revoked users may not corporate to generate the re-signing keys for re-signatures. This may lead to two serious security problems. First, a malicious cloud may arbitrarily convert signatures of one user into signatures of another using the re-signing keys. Second, collusion between the revoked users and the cloud may disclose the secret keys of all the remaining users in the group. Apart from the two problems above, the computation cost of auditing in [15] is linear to the group size, which limits the scalability of the scheme.

For the sake of secure user revocation and efficient integrity checking, Yuan *et al.* [16] proposed a public integrity checking scheme utilizing polynomial-based authentication tags, which is collusion-resistant and of a constant computational cost on the user side. However, this scheme suffers from two serious secure vulnerabilities. A malicious cloud server is able to discard all the shared data and generate a valid proof of data possession by reserving some intermediate results or a previous valid proof, which we refer to as replace attack and replay attack, respectively. The details can be found in the appendix. As far as we know, it is still an open challenge to design a public integrity auditing scheme for shared data with efficient auditing and secure user revocation.

To solve this challenging problem, we propose a novel public auditing scheme for the integrity of shared data with efficient auditing and secure user revocation. By leveraging the concept of Shamir Secret Sharing, our scheme splits the re-signing process into a number of parts and deploy them to different proxies. Thus the re-signing keys do not need to be computed in advance and stored in the cloud as the previous works [15] does, and this new design prevents the cloud from arbitrarily converting signatures between any two users in the group. In particular, the decentralized re-signing process makes the collusion attacks practically infeasible. Besides, the utilization of our improved polynomial-based authentication tags, which is free from the replace attack and the replay attack mentioned above, makes the auditing of our proposed scheme secure and efficient. In addition, we devise two algorithms to securely outsource the signatures generation and integrity proof verification to untrusted cloud servers, respectively. This allows more efficiency for the client side, namely the group users and the auditor. As we all know, exponentiations and bilinear pairings in ECC group are computation costly, which dominates the computation overhead of the signatures generation and integrity proof verification processes. To speedup the two processes, we investigate on offloading these costly computations to an untrusted cloud server in a privacy-preserving way.

Our research contribution can be summarized as follows:

1) Leveraging the concept of Shamir secret sharing, we propose a public integrity auditing for shared data in the cloud with efficient auditing and collusion-resistant user revocation.

2) We improve the security of polynomial-based authentication tags, the utilization of which makes the auditing of our scheme efficient and secure.

3) We devise algorithms to securely outsource the signatures and verification process to an untrusted cloud server, which allows more efficiency for the client side and makes the scheme more affordable for mobile environments.

4) We evaluate our scheme in numerical analysis and experiments, the results of which demonstrate that our scheme is of provably security and highly efficiency.

The rest of this paper is organized in the following way: Section 2 defines the system model, security model and our design goals. Section 3 introduces several cryptographic primitives. Section 4 presents the design details of our proposed scheme and its security analysis.

Section 6 presents the numerical analysis and experimental results. Section 7 discusses related works. Section 8 concludes the paper.

2. Problem Statement

In this section, we first give a description of the system model and framework for shared data integrity auditing. Then, we define the security model and discuss the security challenges to design a public integrity auditing for shared data with secure user revocation and computation outsourcing. Finally, we give the design goals of the proposed scheme.

2.1. System Model and Framework

As illustrated in Fig. 1, we consider a cloud storage auditing system with four entities: the cloud, the third party auditor (TPA), the group and the proxies. The cloud provides data hosting and sharing service to the group. The TPA is able to publicly audit the integrity of the shared data in the cloud for the group. The group is an entity consisting of users, who create data and share with each other. Users in the group trust each other and are able to manage the group cooperatively. With the services provided by the cloud, users of the group can easily modify the shared data and share data within the group. The shared data is further divided into a series of blocks and each block is attached a signature computed by its modifier.

2.2. Security Model

Generally, there are three sources of threats for the integrity of shared data in the cloud. The first is the cloud servers, which may unawarely corrupt shared data due to human errors and hardware/software failures, or intentionally hide data lose from the users for reputation. What's worse, a malicious cloud may try to convert signatures of an important user into signatures of another less important user since the blocks signed by the former will be accessed more frequently than the latter, which will cut down the operation cost. The second is the external adversaries, who may try to prevent the group users from using the shared data correctly by destroying them. The third is the revoked users, who may still try to generate valid signatures of the group or collude with the cloud to disclose the secret keys of other users in the group through the re-signature keys that are used in re-signing process.

To ensure the integrity of the shared data, users need to compute signatures for the blocks that are modified and outsource the blocks together with their corresponding signatures to the cloud. When a user leaves the group for some reasons, such as misbehavior, the blocks previous signed by this revoked user need to be re-signed such that the integrity of shared data can still be

audited and only be audited with the public keys of the existing users in the group. During the re-signing process, a re-signing key is needed. However, we can not always expect that the revoked users will cooperate in the re-signing key generation process, which involves the secret keys of the revoked user and an existing user of the group. Thus, a direct and trivial solution, which is adopted by the previous work [15], is to compute the re-signing keys for any two users of the group in advance and store these re-signing keys in the cloud. This gives the chance for the revoked users and the cloud to collude to reveal the secret keys of the other users in the group.

2.3. Design Goals

To provide an efficient public integrity auditing for group data with efficient auditing and collusion-resistant user revocation, the following goals are kept in our mind throughout the design: (1) Correctness: The TPA is able to verify the integrity of shared data with the public keys of the existing users in the group even if user revocation happens. (2) Efficiency: The scheme should be efficient for the client side. In other words, the signatures generation and user revocation processes are efficient for users side and the integrity proof verification process is efficient for the auditor side. Besides, the storage overhead of our scheme is small. (3) Collusion-resistance: The user revocation is collusion-resistant and the integrity auditing process is secure against replace attack and replay attack.

3. Preliminaries

In this section, we introduce some cryptographic techniques that will be used in this paper, including bilinear maps, threshold secret sharing and proxy re-signatures.

3.1. Bilinear Map

Let G_1 and G_2 be multiplicative cyclic groups with the same prime order p, and g is the generator of group G_1. A bilinear map is a map $e : G_1 \times G_1 \rightarrow G_2$ such that for all $u, v \in G_1$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$. Of course, the map e must be efficiently computable and non-degenerate: $e(g, g) \neq 1$.

3.2. Shamir Threshold Scheme

Threshold scheme, first proposed by Shamir *et al.* [17], is able to distribute a secret among a group of participants, each of whom is allocated a share of the secret. The secret can be easily reconstructed with sufficient shares. Besides, it is impossible to recover the secret value

without sufficient shares. Suppose that there are n participants $P_i (i = 1, \dots, n)$ in the group and the secret is $s \in \mathbb{Z}_p$, the procedure of a (κ, n) threshold scheme is as follows:

Firstly, the dealer random choose $\kappa - 1$ random elements $a_i \in \mathbb{Z}_p$ for i = 1, … , $\kappa - 1$, and construct an interpolation polynomial:

$$L(x) = s + a_1 x + \cdots + a_{\kappa-1} x^{\kappa-1}$$

Secondly, the dealer computes $y_j = L(x_j) \in \mathbb{Z}_p, 1 \le j \le n$ with different x_j and sends (x_j, y_j) to P_i. Note that $s = L(0)$.

Thirdly, given a set of at least κ different points $\{(x_{l_j}, y_{l_j})\}_{1 \le j \le \kappa}$ in the $L(x)$, the dealer is able to reconstruct the secret value as follows:

$$L(x) = \sum_{j=1}^{\kappa} y_{l_j} \lambda_{l_j}(x)$$

$$\lambda_{l_j}(x) = \prod_{1 \le f \le \kappa, f \ne j} \left(\frac{x - x_{l_f}}{x_{l_j} - x_{l_f}}\right)$$

where . Then the secret is recovered $s = L(0) = \sum_{j=1}^{\kappa} y_{l_j} \lambda_{l_j}(0)$. Using the Shamir threshold scheme, the secret key of users can be split and distributed among the groups, which can be recovered for computing the re-signing keys when user revocation.

3.3. Proxy re-signatures

Blaze *et al.* [18] first proposed the concept of proxy re-signatures, which allow a semi-trusted proxy to convert signatures from one user to signatures from another. Meanwhile the proxy itself does not learn any signing keys and can not arbitrarily compute signatures on behalf of both of the users. In [19], Chow *et al.* investigated the security of bidirectional schemes and showed how to design a generic unidirectional proxy re-signature scheme using homomorphic compartment signature. Instead of the above two schemes, our scheme is based on the proxy re-signatures proposed by Ateniese *et al.* [20]. However, direct adoption of the proxy re-signatures of Ateniese will lead to serious security problems, namely collusion attacks.

3.4. Algorithm for outsourcing multi-exponentiations

The algorithm *GExp*, proposed by Wang *et al.* [21], allows a client to compute product of a series of exponentiations with the help of an untrusted cloud server. It takes $a_{i,j} \in \mathbb{Z}_p$ and $u_{i,j}$

$\in G$ $(1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s)$ as inputs, and outputs $(\prod_{j=1}^{s} u_{1,j}^{a_{1,j}}, \ldots, \prod_{j=1}^{s} u_{r,j}^{a_{r,j}})$. The algorithm can be described as follows:

Step 1: The client invokes BPV+ or SMBL [21] to generate four random pairs $\{(\alpha_i, \mu_i)\}_{1 \leqslant i \leqslant 4}$, where $\mu_i = g^{\alpha_i}$. Pick a random value χ with respect to the security parameter and a series of random number $b_{i,j} \in \mathbb{Z}_p, (1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s)$. Then the client computes the following values:

$$c_{i,j} = a_{i,j} - b_{i,j}\chi$$
$$w_{i,j} = u_{i,j} / \mu_1$$
$$h_{i,j} = u_{i,j} / \mu_3$$
$$\theta_i = (\alpha_1 \sum_{j=1}^{s} b_{i,j} - \alpha_2)\chi + (\alpha_3 \sum_{j=1}^{s} c_{i,j} - \alpha_4)$$

Step 2: The client invokes BPV+ or SMBL to obtain $\{(t_k, g^{t_k})\}_{1 \leqslant k \leqslant r+2}$ and queries the cloud server U in random orders as:

$$U(\theta_i / t_i, g^{t_i}) \rightarrow B_i, \text{ for } 1 \leqslant i \leqslant r$$
$$U(\theta / t_{r+1}, g^{t_{r+1}}) \rightarrow A, \text{ where } \theta = t_{r+2} - \sum_{i=1}^{r} \theta_i$$
$$U(b_{i,j}, w_{i,j}) \rightarrow C_{i,j}, \text{ for } 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s$$
$$U(c_{i,j}, h_{i,j}) \rightarrow D_{i,j}, \text{ for } 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant s$$

When receives a query $U(a, b)$, the cloud computes and returns b^a to the client.

Step 3: The client checks whether $A \cdot \prod_{i=1}^{r} B_i \stackrel{?}{=} g^{t_{r+2}}$. If it holds, then the client computes the results as follows, for $1 \leqslant i \leqslant r$,

$$\prod_{j=1}^{s} u_{i,j}^{a_{i,j}} = (\mu_2 \prod_{j=1}^{s} C_{i,j})^{\chi} B_i \mu_4 \prod_{j=1}^{s} D_{i,j}$$

Although the algorithm allows the client to compute the product of a series of exponentiations via secure computation outsourcing, it still needs the client to composite the final result. This incurs quite a number of multiplications and additions for the client side.

3.5. Algorithm for outsourcing single bilinear pairing

The algorithm of Chen *et al.* [22] allows a client to compute $e(A, B)$ with the help of two untrusted cloud servers in a privacy-preserving way. Let $U_i(A_1, A_2) \rightarrow e(A_1, A_2)$ denote that cloud

server U_i takes as inputs (A_1, A_2) and ouputs $e(A_1, A_2)$. Their algorithm can be described as follows:

(1) The client invokes a subroutine named *Rand* [22] to obtain three six-tuple (X_i, Y_i, x_iX_i, y_iY_i, λ_i), where $\lambda_i = e(x_iX_i, y_iY)$ and $i = 1, 2, 3$.

(2) The client queries U_1 in random orders as

$$U_1(A + x_1X_1, B + y_1Y_1) \rightarrow e(A + x_1X_1, B + y_1Y_1) = \alpha_1$$
$$U_1(x_1X_1 + y_1X_1, Y_1) \rightarrow e(X_1, Y_1)^{x_1+y_1} = \alpha_2$$
$$U_1(x_2X_2, y_2Y_2) \rightarrow e(x_2X_2, y_2Y_2) = \beta_1$$
$$U_1(x_3X_3, y_3Y_3) \rightarrow e(x_3X_3, y_3Y_3) = \gamma_1$$

(3) The client queries U_2 in random orders as

$$U_2(A + X_1, y_1Y_1) \rightarrow e(A + X_1, y_1Y_1) = \alpha_3$$
$$U_2(x_1X_1, B + Y_1) \rightarrow e(x_1X_1, B + Y_1) = \alpha_4$$
$$U_2(x_2X_2, y_2Y_2) \rightarrow e(x_2X_2, y_2Y_2) = \beta_2$$
$$U_2(x_3X_3, y_3Y_3) \rightarrow e(x_3X_3, y_3Y_3) = \gamma_2$$

(4) Finally, the client checks the responses of U_1 and U_2 by $\beta_1 \stackrel{?}{=} \beta_2$ and $\gamma_1 \stackrel{?}{=} \gamma_2$. If they hold, the client computes $e(A, B) = \alpha_1\alpha_2\alpha_3^{-1}\alpha_4^{-1}\lambda_1^{-1}$

The above algorithm uses an offline subroutine *Rand* to speedup the online computations. However, it does not support batch outsourcing of multiplications of a series bilinear pairings, which are the cases in integrity proof verification process.

4. Our Scheme

In this section, we first give an overview of our scheme and then we present the design details. Last but not least, we enable our scheme to support data dynamics.

4.1. Overview

To achieve the design goals, we proposed a new public integrity verification scheme for shared data utilizing the concept of Shamir threshold, which requires the users to distribute their secret values among the group and allows the group to compute a re-signing key for a revoked user after revocation. Thus our scheme does not need to compute all the re-signing keys between any two users of the group in advance and store the re-signing keys in the cloud as [15]. Besides, the proposed scheme is able to limit the capacity of the cloud and make the collusion between the cloud and revoked users practically infeasible. Moreover, we have improved the

polynomial-based authentication tags [16] with resistance against replace attack and replay attack mentioned above. By adoption of our improved polynomial-based authentication tags, our scheme is able to reduce the communication cost and the computation overhead of TPA during auditing process.

Specifically, we assume that there are d users $\{u_k\}_{1\leqslant k\leqslant d}$ in the group. In the **GroupSetup** procedure, every user u_k in the group generates his/her secret value and distributes it among the group with a threshold κ. When user u_k creates or modify a data block, he/she computes a signature on this block in **SignGen** procedure. After u_a is revoked from the group, the group is able to generate a re-signing key $rk_{a\rightarrow b}$, which is used for converting signatures previously computed by this revoked user into signatures from an existing user u_b in the **ReSignature** procedure. In the **Chanllenge** procedure, the TPA generates a challenge message (CM) and sends it to the cloud. Given the challenge message, the cloud is able to generate a proof of possession of shared data in the **ProofGen** procedure. Finally, the TPA is able to verify the integrity of shared data in the cloud by checking the correctness of the proof in the **ProofVerify** procedure.

4.2. Scheme Description

We now describe the design details of our scheme. Let G_1, G_2 be two multiplicative cyclic groups of prime order p and g be the generator of G_1. $e : G_1 \times G_1 \in G_2$ is a bilinear map as introduced in preliminaries and $H : \{0, 1\}^* \rightarrow G_1$ is a secure hash function. The shared data is split into n blocks, each has s segments: $\{m_{ij}\}$, $1 \leq i \leq n$, $1 \leq j \leq s$. The proposed scheme is described as follows:

SysInit. In this procedure, system parameters are produced. The founder of this group randomly selects $\alpha \in \mathbb{Z}_p$ and computes $\{g^{\alpha^j}\}_{1\leq j\leq s}$. Thus, the public system parameters are $\{p, g, n, s, \{g^{\alpha^j}\}_{1\leq j\leq s}\}$ and α is shared with the TPA.

GroupSetup. For every user u_i in the group, he/she generates a random $\varepsilon_i \in \mathbb{Z}_p$ as his/her secret value and computes his/her private key $sk_i = \varepsilon_i$ and public key $pk_i = g^{\varepsilon_i}$. Then user u_i distributes his/her secret value among the group as follows:

1) The user u_i randomly selects $a_i \in \mathbb{Z}_p, i \in [1, \kappa - 1]$ and constructs an interpolation polynomial $L(x) = 1/\varepsilon_i + a_1 x + \cdots + a_{\kappa-1} x^{\kappa-1}$, obviously $1/\varepsilon_i = L(0)$.

2) The user u_i first generates d different random values $x_k \in \mathbb{Z}_p, k = 1, \ldots, d$ and computes the secret value shares $y_k = L(x_k)$. Then the user u_i sends (x_k, y_k) to user u_k to distribute the secret value of user u_i among the group.

SignGen. When u_i creates or modifies a block m_k of the shared data, the user computes a signature on the block as:

$$\sigma_k = (H(id_k) \prod_{j=1}^{s} (g^{\alpha^j})^{m_{kj}})^{\varepsilon_i} \in G_1$$

where id_k is the block identifier for m_k and the detailed design of id_k is described in the next part. Then the user updates the corresponding block and its signature in the cloud.

ReSignature. When user u_a is revoked from the group, the signatures previously computed by this revoked user need to be transformed into signatures of an existing user in the group as the following steps:

1) The cloud first randomly selects an existing user u_b to be responsible for the blocks that were previously signed by the revoked user u_a. Meanwhile the cloud randomly selects an element $R \in \mathbb{Z}_p$ and sends it to u_b. Receiving the random R from the cloud, the user u_b computes ε_b/R and sends it to the group. Note that there is no particular restriction for the choosing of user u_b, as long as u_b is an existing user in the group.

2) Given the ε_b/R and the secret value share (x_k, y_k) of u_a, user u_k of the group computes $\xi_k = y_k \cdot \varepsilon_b/R$ as the re-signing key share and $P_k = g^{\xi_k}$ as the public key for this re-signing key share. Then user u_k sends (x_k, ξ_k) to one of the proxies and sends P_k to the cloud for verifying the re-signature share generated by the selected proxy.

3) Given the block m_l and its signatures σ_l previously computed by the revoked user u_a, the proxy first verifies the signature $e(\sigma_l, g) = e(H(id_l) \prod_{j=1}^{s} g^{\alpha^j m_{lj}}, pk_a)$. If the equation holds, the proxy computes the re-signature share $\sigma_l^{(k)} = \sigma_l^{\xi_k}$ with the received re-signing key share (x_k, ξ_k) and sends the result $(x_k, \sigma_l^{(k)})$ to the cloud as a re-signature share.

4) Given at least κ re-signature shares $(x_k, \sigma_l^{(k)})$ from the proxies and the random R, the cloud is able to compose the re-signature. For simplicity, we denote the indexes set of the received re-signature shares as φ. First, the cloud verifies the correctness of the received re-signature shares by checking $e(\sigma_l^{(k)}, g) \stackrel{?}{=} e(\sigma_l, P_k), k \in \varphi$. If all the equations hold, the cloud composes the re-signature as:

$$\sigma_l' = \prod_{k \in \varphi} (\sigma_l^{(k)})^{\lambda_k(0) \cdot R} \quad (1)$$

Challenge. To audit the integrity of shared data, the TPA first randomly select c blocks out of all the blocks of the shared data, denote the indexes of the selected blocks as L. Then the TPA generates two random number $x, r \in \mathbb{Z}_p$ and computes $X = g^x$, $R = g^r$. After that, the TPA computes $\{X^{\alpha^j}\}_{1 \le j \le s}$. Finally, the TPA outputs the challenge message $CM = \{L, R, \{X^{\alpha^j}\}_{1 \le j \le s}\}$ and sends it to the cloud.

ProofGen. On receiving the challenge message $CM = \{L, R, \{X^{\alpha^j}\}_{1 \le j \le s}\}$, the cloud generates a proof of possession of shared data as follows:

1) The set L of selected blocks is divided into subset $L_1, \ldots, L_d$, where L_i is the subset of selected blocks that are signed by user u_i.

2) For each subset L_i, the cloud computes

$$\mu_{ij} = \sum_{l \in L_i} m_{lj}, 1 \le j \le s \quad (2)$$

and

$$\pi_i = \prod_{l \in L_i} e(\sigma_l, R) = e(\prod_{l \in L_i} H(id_l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g)^{\varepsilon_i r}$$

3) The cloud compose the proof as

$$\omega_i = \prod_{j=1}^{s} X^{\alpha^j \mu_{ij}}, 1 \le i \le d \quad (3)$$

and

$$\pi = \prod_{i=1}^{d} \pi_i$$

Then the cloud outputs a proof $Prf = \{\{\omega_i\}_{1\le i\le d}, \pi\}$ as the response to the challenge message from the TPA.

ProofVerify. Given the auditing proof $Prf = \{\{\omega_i\}_{1\le i\le d}, \pi\}$ together with the challenge message $CM = \{L, R, \{X^{\alpha^j}\}_{1\le j\le s}\}$, the TPA can verify the integrity of shared data by checking the correctness of the following equation:

$$\prod_{i=1}^{d} e(\eta_i^x, pk_i^r) \cdot e(\omega_i, pk_i^r) \stackrel{?}{=} \pi^x \quad (4)$$

where $\eta_i = \prod_{l\in L_i} H(id_l), 1 \le i \le d$. Note that, the above equation can be further rewrited as $(\prod_{i=1}^{d} e(\eta_i^x \omega_i, pk_i))^r \stackrel{?}{=} \pi^x$. If the equation holds, the TPA outputs TRUE, otherwise it returns FALSE.

4.3. Support Data Dynamics

To support data dynamic operations, the identifiers of data blocks need to be carefully designed. This is because that the idnetifiers of blocks are included in the computation of signatures. If we directly use the block indexes in the computation of signatures, the indexes of blocks after a inserted block or a deleted block will be all changed. This requires to re-compute the signatures of all these blocks even if the content of them are not changed. To solve this problem, we design a data structure called dynamic data table (DDT), which allows a user to modify a single block without changing the identifiers of the other blocks and their signatures, as shown in Table 1.

The DTT is stored in the cloud and the TPA. More concretely, a block identifier is unique in the dynamic data table and is described as $id_k = \{B_k \| V_k \| E_k \| H_k \| S_i\}$, where B_k is the logical index of this block, V_k is the update counter of this block with −1 representing that this block is deleted, E_k is the record for block insertion, S_i is the signer id of this block and H_k is computed as $H_k = Hash(m_k \| B_k \| V_k \| E_k \| S_i)$. Every time a data block is updated, the responding V_k will be increased by 1, as illustrated in second sheet of Table 1. When V_k is set to −1, as shown in the forth sheet of Table 1, it means that the data block is deleted. When inserting a data block after B_k, the inserted block has the same logical index as its former block, but the value of E is increased by 1, that is $E_k + 1$, as demonstrated in the third sheet of Table 1. Besides,

S_i is needed to distinguish the signer of every blocks in the shared data. Mention that the DDT can be split into several parts and accesses to it can be balanced among these parts.

4.4. Analysis of our scheme

In this part, we analyse the correctness and security of the proposed scheme.

4.4.1. Correctness

We first prove the correctness of our scheme, which is concluded in the following theorems:

Theorem 1. *In our scheme, the group together with the cloud and the proxies is able to convert the signatures from one revoked user into signatures from an existing users after the revocation.*

Proof. To prove the correctness of this theorem is equivalent to prove Equation (1) is correct. Based on the properties of Shamir Secret Sharing, the correctness of Equation (1) is presented as the following:

$$\sigma_l' = \prod_{k\in\varphi} (\sigma_l^{(k)})^{\lambda_k(0)\cdot R} = \prod_{k\in\varphi} \sigma_l^{\xi_k \lambda_k(0)\cdot R}$$

$$= \prod_{k\in\varphi} \sigma_l^{y_k \cdot \frac{\varepsilon_b}{R} \cdot \lambda_k(0)\cdot R} = \sigma_l^{\varepsilon_b \sum_{k\in\varphi} y_k \lambda_k(0)}$$

From the Preliminaries section above, we note that $\sum_{k\in\varphi} y_k \lambda_k(0) = L(0) = 1/\varepsilon_a$. Therefore, we can get $\sigma_l' = \sigma_l^{\varepsilon_b \cdot 1/\varepsilon_a} = (H(id_k)\prod_{j=1}^{s} (g^{\alpha^j})^{m_{kj}})^{\varepsilon_b}$, which is a valid signature of user u_b. Thus, the correctness is proved.

Theorem 2. *In the proposed public auditing scheme, the cloud passes the auditing iff all the selected data blocks and their corresponding authenticators are correctly stored in the cloud.*

Proof. We first prove that if the selected data blocks and their corresponding signatures are correctly stored in the cloud, then the cloud will pass the auditing. The proof lies in the following equations:

$$\prod_{i=1}^{d} e(\eta_i^x, pk_i^r) \cdot e(\omega_i, pk_i^r)$$

$$= \prod_{i=1}^{d} e((\prod_{l \in L_i} H(id_l))^x, g^{\varepsilon_i r}) \cdot e(\prod_{j=1}^{s} (g^z)^{\alpha^j \sum_{l \in L_i} m_{lj}}, g^{\varepsilon_i r})$$

$$= \prod_{i=1}^{d} e(\prod_{l \in L_i} H(id_l), g)^{x\varepsilon_i r} \cdot e(\prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g)^{x\varepsilon_i r}$$

$$= \prod_{i=1}^{d} e(\prod_{l \in L_i} H(id_l) \prod_{j=1}^{s} g^{\alpha^j \sum_{l \in L_i} m_{lj}}, g)^{x\varepsilon_i r}$$

$$= \prod_{i=1}^{d} \pi_i^x = (\prod_{i=1}^{d} \pi_i)^x = \pi^x$$

On the other hand, if the selected blocks in the challenge is corrupted, the cloud is not able to generate a valid proof. Thus, the cloud will fail to pass the auditing process launched by the TPA. □

Theorem 3. *In the proposed scheme, the TPA is able to verify the integrity of outsourced data in the cloud with a very high probability.*

Proof. Based on Theorem 1 and Theorem 2 is equivalent to prove that the TPA is able to detect the data corruption in the cloud with a very high probability through the auditing process. Assume that φ blocks out of n blocks of the shared data are damaged. Let χ be the number of damaged blocks that are selected by the TPA during the auditing process and P_χ be the probability that at least one damaged block is selected by the TPA. According to the "sampling" strategy of the auditing process, we have

$$P(\chi = 0) = 1 - P_\chi = \frac{n-\varphi}{n} \cdot \frac{n-1-\varphi}{n-1} \cdot \ldots \cdot \frac{n-c+1-\varphi}{n-c+1}$$

Thus, $1 - \left(\frac{n-\varphi}{n}\right)^c \le P_\chi \le 1 - \left(\frac{n-c+1-\varphi}{n-c+1}\right)^c$. As shown in Table 2, the TPA is able to detect the error in the shared data with a extremely high probability by asking proof for a constant amount of randomly selected data blocks. Besides, the error detection probability is independent from the total number of data blocks. For example, if 1% of the shared data blocks are corrupted, the TPA is able to detect the corruption with a probability of 95% by challenging about 300 blocks, or 450 blocks for 99% error detection probability. □

4.4.2. Security Analysis

We now give a brief analysis of the security of our scheme in the following theorems. Before that, we'll introduce some assumptions first.

Definition 1 (Discrete Logarithm (DL) Problem)**.** *The DL problem is that, for* $a \in \mathbb{Z}_p$*, take g,* $g^a \in G$ *as inputs and output a.*

If it is computationally infeasible to solve the DL problem in a group G_1, we say that the DL assumption holds in the group G_1.

Definition 2 (Computational Diffie-Hellman (CDH) Problem)**.** *The CDH problem is that, for* $a, b \in \mathbb{Z}_p$*, take g,* g^a*,* $g^b \in G_1$ *as inputs and output* g^{ab}*.*

If it is computationally infeasible to solve the CDH problem in a group G_1, we say that the CDH assumption holds in the group G_1.

Theorem 4. *For the cloud, it is practically infeasible to extract a re-signing key if the security of Shamir's secret sharing holds.*

Proof. We prove this theorem based on the security of Shamir's secret sharing scheme [17]. Since the secret value $1/\varepsilon_i$ of the users are split into d shares using interpolation polynomial and distributed among the group, an adversary needs to compromise at least κ users in the group to obtain sufficient secret value shares to generate a re-signing key, which is practically infeasible considering the cost. Suppose that there are 100 users in the group, for a threshold value of 50, it means that the adversary needs to compromise at least 50 users, which is costly and practically infeasible. Therefore, our scheme hides the re-signing keys from the cloud, which makes it practically infeasible for the collusion attacks from the cloud and the revoked users.

Theorem 5. *For the cloud, It is computational infeasible to produce a forgery of an auditing proof in our proposed scheme.*

Proof. First, we prove that a probabilistic polynomial time adversary is computationally infeasible to generate a forgery signatures through forging a re-signing key to pass the auditing. Assume that an adversary is able to forge a re-signing key to generate a valid re-signature in a security game, named *Game 1*, then we can find a solution to solve the Discrete Logarithm problem in G_1 with a non-negligible probability. The Game 1 is defined as follows:

Game 1: The group together with the cloud can generate a correct re-signing key $rk_{a\to b}$, which is able to convert the signatures of a revoked user u_a into ones of an existing user u_b in the

group and the results of this converting should pass the verification with Equation (2). However, a malicious adversary may generate a forgery of re-signing key $rk'_{a\to b}$ for the transformation. If the results of transformation using the forged re-signing key still pass the auditing of the TPA, then the adversary wins this game, otherwise, it fails. For simplicity, we denote σ_k as the signature of user u_a and verify the equation $e(H(id_k)^x, pk_i^r) \cdot e(\prod_{j=1}^{s} X^{\alpha^j m_{kj}}, pk_i^r) = e(\sigma_k, R)^x$ that represents Equation (2).

We first assume that the adversary wins the game. Then we can get the following results:

$$e(\eta^x, pk_b^r) \cdot e(\omega, pk_b^r) = e(\sigma_k^{rk'_{a\to b}}, R)^x$$

and

$$e(\eta^x, pk_b^r) \cdot e(\omega, pk_b^r) = e(\sigma_k^{rk_{a\to b}}, R)^x$$

Based on the properties of Bilinear Map, we can learn that

$$\sigma_k^{rk'_{a\to b}} = \sigma_k^{rk_{a\to b}}$$

Obviously, it means that given $\sigma_k, \sigma_k^{rk_{a\to b}} \in G_1$, we can output $rk'_{a\to b}$, which contradicts the assumption that DL problem is computationally infeasible to solve in G_1.

Then, we prove that the adversary can not pass the auditing through replace attack or replay attack. To launch the replace attack, the adversary cloud must be able to combine computations of Eq. 2 and Eq. 3. However, the challenged blocks set $L = \{L_1, \ldots, L_d\}$ and the base $X = g^x$ are both randomly generated in every challenge. Thus, it is computational infeasible for the cloud to pre-compute some intermediate results for incoming challenges. As to launch the replay attack, the cloud must able to compute η^x. Assume the cloud is able to compute η^x, then we can find a way to solve the CDH problem, which is computationally infeasible to solve in G_1. Specifically, since G_1 is a cyclic group and $\eta = \prod_{l\in L} H(id_l) \in G_1$, then there exists $\theta \in \mathbb{Z}_p$ that $\eta = g^\theta$. According the assumption, the cloud is able to compute $g^{\theta x}$, which indicates that the cloud can output $g^{\theta x}$ by taking $g, g^\theta, g^x \in G_1$ as inputs. This contradicts the CDH assumption in G_1.

5. Extension with Secure Computations Outsourcing

In this section, we extend our scheme described in the previous section with secure computations outsourcing, which allows more efficiency for the client side in our scheme and makes it more affordable for the mobile computing environment.

As we know, exponentiations and bilinear pairings are very costly computations, which are the main computations for our scheme and other cloud storage integrity auditing schemes [15, 16]. To reduce the computation overhead for the client side, we extend our scheme to allow secure computations outsourcing. Specifically, based on the protocols proposed by Wang *et al.* [21], we extend the **SignGen** to allow the group users to generate a signature on a block of the shared data with the help of the untrusted cloud servers. Besides, based on [23], we devise a verification outsourcing algorithm to allow the auditor to verify the correctness of the auditing proof with the help of the untrusted cloud servers.

5.1. Signatures Outsourcing

As described above, the **SignGen** procedure of our scheme requires the group user to compute the product of a substantial number of modular exponentiations, which imposes heavy computation burdens on the group users. To improve the efficiency of signatures generation, we intend to offload the multi-exponentiations of the computations below to two untrusted cloud servers:

$$\sigma_k = \left(H(id_k) \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{m_{kj}} \right)^{\varepsilon_i}$$

Unlike the scheme of Wang *et al.* [21], our signatures outsourcing algorithm is in the two untrusted servers model, where collusion is not allowed. Note that, the bases $\{g^{\alpha^j}\}_{1 \leqslant j \leqslant s}$ are public system parameters. Thus we only needs to preserve the privacy of $\{m_{kj}\}_{1 \leqslant j \leqslant s}$ and the outputs.

5.1.1. Description of Signatures Outsourcing Algorithm

Step 1: The group user selects two random values χ_1, χ_2 such that $\chi_1, \chi_2 \geqslant 2^{\lambda}$, where λ is a security parameter. Then the group user pikcs two series of random values $b'_j, b_j \in_R \mathbb{Z}_p$, for $1 \leqslant j \leqslant s$. After that, the group user computes the following values:

$$c_j = m_{kj} - b_j \chi_1$$
$$c'_j = m_{kj} - b'_j \chi_2$$

Step 2: Let $B = \{b_j\}_{1 \leqslant j \leqslant s}$, $B' = \{b'_j\}_{1 \leqslant j \leqslant s}$, $C = \{c_j\}_{1 \leqslant j \leqslant s}$, $C' = \{c'_j\}_{1 \leqslant j \leqslant s}$ and $\mu = \{g^{\alpha^j}\}_{1 \leqslant j \leqslant s}$, the group user separately queries the two cloud servers U_1 and U_2 in random orders as follows.

$$U_1(B, \mu) \to \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{b_j} = \varphi_1$$

$$U_1(C, \mu) \to \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{c_j} = \phi_1$$

$$U_2(B', \mu) \to \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{b'_j} = \varphi_2$$

$$U_2(C', \mu) \to \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{c'_j} = \phi_2$$

Step 3: The group user verifies the results by checking $\phi_1 \varphi_1^{\chi_1} \stackrel{?}{=} \phi_2 \varphi_2^{\chi_2}$. If it holds, the group user computes the signature as:

$$\sigma_k = \left(H\left(id_k\right) \phi_1 \varphi_1^{\chi_1} \right)^{\varepsilon_i}$$

5.1.2. Discuss of Signatures Outsourcing Algorithm

The correctness of the signatures outsourcing algorithm lies in the above equation, which can be proofed as below:

$$\begin{aligned}
\sigma_k &= \left(H\left(id_k\right) \phi_1 \varphi_1^{\chi_1} \right)^{\varepsilon_i} \\
&= \left(H\left(id_k\right) \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{c_j} \left(\prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{b_j} \right)^{\chi_1} \right)^{\varepsilon_i} \\
&= \left(H\left(id_k\right) \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{m_{kj} - b_j \chi_1} \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{b_j \chi_1} \right)^{\varepsilon_i} \\
&= \left(H\left(id_k\right) \prod_{j=1}^{s} \left(g^{\alpha^j} \right)^{m_{kj}} \right)^{\varepsilon_i}
\end{aligned}$$

The inputs and outputs are blinded by χ_1, χ_2 and $\{b_j, b'_j\}_{1 \leqslant j \leqslant s}$. The checkability is 1. Intuitively, the above signatures outsourcing scheme has greatly reduced the number of exponentiations that the group user have to perform. Experimental results in the next section has shown the efficiency gain.

5.2. Verification Outsourcing

In the **ProofVerify** procedure, the auditor needs to perform 4*d* pairing operations, which dominates the computation cost of the auditor side. Utilizing the concept of outsourcing computation [24], we extend our scheme to support secure verification outsourcing. By outsourcing, the auditor is able to efficiently compute a series of bilinear pairings and obtain the product of them with the help of the untrusted cloud servers. Specifically, the verification is divided into two parts, namely, a trusted part which is performed by the auditor and is efficient compared with the original verification, and an untrusted part *U* which is invoked by the auditor and carried out by the untrusted cloud servers.

The left side of the verification equation (2) can be rewritten as $\prod_{i=1}^{d} e(\eta_i^{x}\omega_i, pk_i)^r$, where pk_i are public constants since they are public keys and $\eta_i^{x}\omega_i$ are secret variables due to that they contain random value x that is kept private by the auditor. Let $A_i = \eta_i^{x}\omega_i$ and $B_i = pk_i$, where $1 \leqslant i \leqslant d$. The main challenge to outsource the verification is to compute the following product of bilinear pairings with the help of the untrusted cloud servers in a privacy-preserving way, where A_i are secret variables and B_i are public constants.

$$\prod_{i=1}^{d} e(A_i, B_i)$$

5.2.1. Description of Verification Outsourcing Algorithm

Similar to [23], the auditor is equipped with a series of constants $\gamma_i = e(g, B_i)$, where $1 \leqslant i \leqslant d$. To speedup the online computations, we also adopt a subroutine named *Rand* (e.g. BPV+ and SMBL [21]), to generate random pairs (v, g^v), where $v \in \mathbb{Z}_p$. The algorithm consists of the following steps:

Precompute: The auditor computes $\gamma_i = e(g, B_i)$ for $1 \leqslant i \leqslant d$. Considering that B_i, $1 \leqslant i \leqslant d$ are public constants, this procedure can be computed offline by a trusted server in advance.

Request: The auditor first computes $A_i = \eta_i^{x}\omega_i$ and then invokes *Rand* to generate a series random pairs $\{(a_i, g^{a_i})\}_{1 \leqslant i \leqslant d}$ and $\{(b_i, g^{b_i})\}_{1 \leqslant i \leqslant d}$. After that, the auditor computes

$\hat{A}_i = A_i \cdot g^{a_i}$ and $\tilde{A}_i = A_i \cdot g^{b_i}$ for $1 \leqslant i \leqslant d$. Let $\hat{A} = \{\hat{A}_1, \ldots, \hat{A}_d\}$, $\tilde{A} = \{\tilde{A}_1, \ldots, \tilde{A}_d\}$ and $B = \{B_1, \ldots, B_d\}$, the auditor queries U_1 and U_2 as follows:

$$U_1(\hat{A}, B) \rightarrow \prod_{i=1}^{d} e(\hat{A}_i, B_i) = \alpha$$
$$U_2(\tilde{A}, B) \rightarrow \prod_{i=1}^{d} e(\tilde{A}_i, B_i) = \beta$$

Verify: To verify the responses of the cloud servers, the auditor retrieves the results as follows:

$$\lambda_1 = \alpha \prod_{i=1}^{d} \gamma_i^{-a_i} = \prod_{i=1}^{d} e(A_i, B_i)$$
$$\lambda_2 = \beta \prod_{i=1}^{d} \gamma_i^{-b_i} = \prod_{i=1}^{d} e(A_i, B_i)$$

and verifies the results by checking whether $\lambda_1 \stackrel{?}{=} \lambda_2$. If it holds, the auditor accepts $\prod_{i=1}^{d} e(A_i, B_i) = \lambda_1 = \lambda_2$. Otherwise, it indicates that at least one of the cloud servers has produced wrong response, and thus the auditor rejects the results.

Output: Finally, the auditor checks whether $\lambda_1^r \stackrel{?}{=} \pi^x$ and outputs the result as the verification report.

5.2.2. Discuss of Verification Outsourcing Algorithm

The secret variables $\{A_i\}_{1 \leqslant i \leqslant d}$ are random blinded by a series of random pairs $\{(a_i, g^{a_i})\}_{1 \leqslant i \leqslant d}$ and $\{(b_i, g^{b_i})\}_{1 \leqslant i \leqslant d}$, which preserves the privacy of the inputs from the untrusted cloud servers. Intuitively, the verification outsourcing algorithm is able to significantly improves the efficiency at the auditor side since it offloads all the bilinear pairing operations to the cloud servers. Although the outsourcing algorithm requires some more multiplications in G_1 and G_2, these operations are much more efficient than pairing operations of $e : G_1 \times G_1 \rightarrow G_2$. The efficiency improvement is demonstrated by the experimental results in the next section.

As for the correctness of the algorithm, if the two cloud servers perform honestly, we have

$$\lambda_1 = \alpha \prod_{i=1}^{d} \gamma_i^{-a_i} = \prod_{i=1}^{d} e\left(\hat{A}_i, B_i\right) \prod_{i=1}^{d} e\left(g, B_i\right)^{-a_i}$$
$$= \prod_{i=1}^{d} e\left(A_i \cdot g^{a_i}, B_i\right) \prod_{i=1}^{d} e\left(g, B_i\right)^{-a_i}$$
$$= \prod_{i=1}^{d} e\left(A_i, B_i\right)$$
$$= \prod_{i=1}^{d} e\left(A_i \cdot g^{b_i}, B_i\right) \prod_{i=1}^{d} e\left(g, B_i\right)^{-b_i}$$
$$= \prod_{i=1}^{d} e\left(\tilde{A}_i, B_i\right) \prod_{i=1}^{d} e\left(g, B_i\right)^{-b_i} = \beta \prod_{i=1}^{d} \gamma_i^{-b_i} = \lambda_2$$

6. Performance Analysis

In this section, we first numerically analyze the primary scheme in terms of communication cost, computation complexity and storage overhead, and then compare it with the two extensions. Finally, we implement a system prototype of our proposed scheme and evaluate it in experiments.

6.1. Numerical Analysis

In this part, we numerically analyze the proposed scheme. A brief comparison between our primary scheme and previous works is presented in Table 3. The detail analysis is described as follows.

6.1.1. Communication Cost

The communication cost of our proposed scheme mainly comes from the auditing process, which consists of two parts: challenge message and possession proof. For the challenge message $CM = \{L, R, \{X^{\alpha^j}\}_{1 \le j \le s}\}$, it is of a constant size $c|n| + (s + 1)|G_1|$ with respect to the number c of challenged blocks, where $|n|$ is the size of an element in set $[1, n]$ and $|G_1|$ is the size of an element in G_1. For the auditing proof $Prf = \{\{\omega_i\}_{1 \le i \le d}, \pi\}$, its size is $d|G_1| + |G_2|$, where $|G_2|$ is the size of an element in G_2. Therefore, the total communication cost of the auditing process is $c|n| + (s + d + 1)|G_1| + |G_2|$. From Table 3, we can see that the communication cost of our scheme is relatively smaller than Wang's [15] since s is usually much smaller than c.

6.1.2. Computation Complexity

We compare user revocation and proof verification with previous works. Since the secret sharing process is very efficient [17], the mainly computation cost of user revocation is to re-sign

a block, which is EXP_{G_1}, one exponentiation operation in G_1. According to Equation (2), the computation cost to verify the possession proof is $dEXP_{G_1} + cMUL_{G_1} + dPair + (d-1)MUL_{G_2} + 2EXP_{G_2}$, where MUL_{G_1} denotes one multiplication operation in G_1, *Pair* denotes one pairing operation on $e: G_1 \times G_1 \rightarrow G_2$, MUL_{G_2} denotes one multiplication operation in G_2 and EXP_{G_2} denotes one exponentiation in G_2. From Table 3, we can see that our user revocation is more efficient than Yuan's [16] and our proof verification is much more efficient than Wang's [15].

6.1.3. Storage Overhead

In our scheme, there are two kinds of data need to be stored, namely the secret value shares of the group users and the signatures of the shared data blocks. For the secret value shares, the total storage overhead is $d(d-1)|p|$ and each user in the group only has to store $(d-1)|p|$ extra data, where $|p|$ is the size of an element in $\mathbb{Z}_p$. For the signatures, the storage overhead is $|G_1||M|/(s|Seg|)$, where the $|M|$ is the size of the shared data, $|Seg|$ is the size of a segment in a block. From Table 3, we can see that the total signatures storage overhead is much smaller than Wang's [15] for data files of the same size.

6.1.4. Comparisons

We discuss the efficiency improvements of the extensions for the client side. A brief comparison of computation cost for the client side is shown in Table 4. It takes the group user $(s+1)EXP_{G_1} + sMUL_{G_1}$ to compute a signature in the primary scheme while it only takes $2sMUL_{\mathbb{Z}_p} + 2(d-1)SUB_{\mathbb{Z}_p} + 3EXP_{G_1} + 3MUL_{G_1}$ for the group user to generate a signature with the help of the untrusted servers, where $MUL_{\mathbb{Z}_p}$ is one multiplication in $\mathbb{Z}_p$ and $SUB_{\mathbb{Z}_p}$ is one substract in $\mathbb{Z}_p$. Note that the multiplications in $\mathbb{Z}_p$ are much more efficient than exponentiations in G_1. For the auditing proof verification, the auditor needs to perform *d* pairing operations in the primary scheme, while it requires no pairing operation for the auditor in the verification outsourcing extension. Although the extension scheme incurs some extra operations in G_2, they are much more efficiency than pairing operations. Experimental results in the next subsection has demonstrated the efficiency improvements of the extensions.

6.2. Experimental Results

Now we evaluate our proposed schemes in experiments. We implement a system prototype of our scheme using Pairing Based Cryptography (PBC) library [25] and deploy it in the Alibaba Cloud. We initiate the cloud server with the ECS instance "ecs.sn1.medium" in Linux (Ubuntu Server 16.04), which is equipped with 2 cores (Intel Xeon E5 2682 v4 2.5GHz CPU) and 4 GB memory. In the following experiments, the security parameter is set to 160 bits. All the experiment results in this paper represent the mean of 10 trials.

6.2.1. Evaluation of GroupSetup

We first evaluate the performance of **GroupSetup** with respect to the group size, namely the number of users in the group, and the results are presented in Fig. 2. Note that we have set the ratio of threshold value and user number to a constant number when the user number varies in this experiment. Then we evaluate the effect of threshold value on the performance of **GroupSetup** and the results can be found in Fig. 3. Since the secret value generation and distribution procedures of different users are parallel, we only consider the time cost of one user in the **GroupSetup**. From the two figures, we can draw the conclusion that the group setup time is linear to the threshold value and has a positive correlation with the group size. Nevertheless, it is worth to note that the **GroupSetup** procedure is one-time and does not affect the real-time performance of our scheme.

6.2.2. Evaluation of User Revocaiton

We argue that our scheme supports efficient and collusion-resistant user revocation. From the security analysis above, we know that the user revocation in our scheme is collusion-resistant. Now we demonstrate the high efficiency of user revocation in our scheme. As a comparison, we consider a straightforward way to revoke a user in the group, which needs an existing user in the group to retrieve the data blocks that were previously signed by the revoked user, compute new signatures on these blocks and upload the new signatures to the cloud. Compared to the straightforward way, our scheme does not introduce heavy communication cost to download data blocks and upload signatures. Moreover, our scheme does not cause much computation on the user side, which is usually resources-limitted. Obviously, the user revocation in our scheme is highly efficient. In this experiment, we measure the performance of user revocation on different threshold value. As shown in Fig. 4, the time to re-sign a data block is positive related to the threshold value and rarely affected by the group size. This is because that the larger the threshold

value is, the more users the re-signing key generation involves and the higher the security of user revocation is, but the longer the time of user revocation takes. To balance the security and efficiency, it needs to choose an appropriate threshold value according to the group size.

6.2.3. Evaluation of Auditing

In the experiment, we evaluate the performance of integrity auditing in our scheme on different challenged blocks and user numbers, respectively. It can be seen from Fig. 5, the more data blocks challenged, the longer it takes the TPA to conduct the integrity auditing, while the higher probability that the TPA is able to detect errors in the shared data (as shown in Table 2). Although the auditing time of our scheme also suffers a linear relationship with the group size as shown in Fig. 6, the computation overhead of our scheme is much smaller than [15] according to Table 3. From the evaluations above, we can draw the conclusion that our scheme supports efficient integrity auditing.

6.2.4. Evaluation of Signatures Outsourcing

In this experiment, we compare the signatures generation of the primary scheme and the signatures outsourcing algorithm in terms of computation overhead for the user side. Table 5 illustrates the time cost of the group user to generate a signature in **SignGen** and the signatures outsourcing algorithm, respectively. It can be seen that the signatures outsourcing algorithm considerable saves the computation cost of the signers and the savings increase with the number of segments in a block.

6.2.5. Evaluation of Verification Outsourcing

This experiment tests the verification outsourcing algorithm and compares it with the **ProofVerify** algorithm in the primary scheme. Fig. 7-9 show the verification time of the two algorithms on varied group size with different number of challenged data blocks. When the auditor challenges 300 data blocks, Fig. 7 shows that the verification time for the two algorithms are both increase with the group size. However, the outsourcing algorithm is much more efficient for the auditor side than **ProofVerify**. This happens similarly when challenging 349 and 460 data blocks, which indicates that the speedup is independent of the group size.

7. Related Work

Provable Data Possession (PDP), first proposed by Ateniese *et al.* [5], which allows a verifier to verify the integrity of the data stored at untrusted servers without retrieving the entire file. Utilizing RSA-based homomorphic authenticators, PDP first achieves public auditing (also

referred to as public verifiability). However, it only supports static data. To support data dynamic operations during auditing, Ateniese *et al.* [26] improved their PDP based on symmetric keys. Unfortunately, this improved scheme only supports partially data dynamic operations. What is worse, this scheme can not support public verifiability and the number of verification requests is also limited.

Based on the PDP model, many schemes have been proposed to meet all kinds of properties: public auditing, efficient auditing, privacy-preserving, data dynamic operations etc. Among them, Shacham and Waters [27] proposed the first public verification scheme in the literature that use the BLS signature [28], which produces much shorter signature than RSA signature at the same security strength. Erway *et al.* [29] proposed the Dynamic Provable Data Possession (DPDP) that first supports full data dynamic operations. To achieve full daynamic, they employ authenticated skip list to eliminate the index information in the tag computation. However, the DPDP scheme is not public verifiable and the efficiency of it is still in question.

In a similar way, Wang *et al.* [7] utilized Merkle Hash Tree (MHT) to eliminate the index information and proposed a scheme that can support public auditing and full data dynamics. Based on the work of Wang, Liu *et al.* [30] proposed a scheme that supports authorized auditing and fine-grained update requests. Their enhanced scheme can extremely reduce communication overheads for small updates. However, users still have to retrieve the unchanged parts of the block modified to re-compute a new signature. At a recent work, Wang *et al.* [8, 9] first introduced the random masking technology to ensure that the auditor can't get any knowledge of the data content during the auditing. Besides, their schemes also supports batch auditing for multiple challenge requests.

In [31], Zhu *et al.* proposed a cooperative provable data possession scheme that allows multiple cloud providers in hybrid cloud to cooperatively prove data integrity to data owners and also extend it to support dynamic auditing. In [13], Wang *et al.* proposed a scheme to audit the integrity of shared data in the cloud for large groups and preserve the identity privacy of users from the auditor during the auditing. Then, they extended their scheme to support public auditing using ring signature in [12] and efficient user revocation using proxy re-signatures in [20]. In another work [32], Wang and Chow *et al.* introduced a security-mediator to generate signatures for data owners, which preserves the anonymity of data owners. However, the scheme can not support traceability for revealing the identities of users when necessary. In [33], Chow *et al.*

designed a secure cloud storage with dynamic user numbers, which supports anonymity of users and traceability simultaneously.

8. Conclusion

In this paper, we propose a novel public integrity auditing scheme for shared data in the cloud with efficient auditing and collusion-resistant user revocation. Utilizing the concept of Shamir secret sharing, our scheme allows the group together with the proxies and cloud to convert signatures from revoked users into ones from the existing users after their revocations. Meanwhile, collusion attack is practically infeasible during the re-signing process. Besides, the signatures outsourcing and verification outsourcing algorithms significantly save computations for the group users and the auditor, which makes our scheme more suitable for mobile computing environments. The numerical analysis and experimental results demonstrate that our scheme is secure and efficient, the total overhead of our auditing scheme is relatively small.

Acknowledgement

This work was supported in part by the NSF China Grants (Project No. 61379144, 61572026, and 61672195) and the Open Foundation of State Key Laboratory of Cryptology.

Yuchuan Luo received the B.S. degree and M.S. degree from National University of Defense Technology (NUDT) in 2009, 2015 respectively, both in Computer Science and Technology. He is currently working towards the Ph.D degree in Computer Science and Technology in NUDT. He is currently studying at the City University of Hong Kong for a year. His research interests include Cloud Computing Security, Crowdsensing Security and Privacy.

Ming Xu is currently a professor and director of the Network Engineering Department in the College of Computer, National University of Defense Technology, China. Prof. Xu is a senior member of CCF and member of IEEE and ACM. His major research interests include mobile computing, wireless network, cloud computing and network security.

Kai Huang received the B.S. degree in Network Engineering and the M.S. degree in Cryptography from Naval University of Engineering, China, in 2007 and 2009, respectively. Currently, he is working towards the Ph.D. degree in the College of Computer at National University of Defense Technology. His research interests are in the areas of Cloud Computing security.

Dongsheng Wang received the B.S. degree from Tsinghua University, Beijing, China and the M.S. degree from National University of Defense Technology, Changsha China, in 2009 and

2012, respectively. He is currently a Ph.D. candidate at National University of Defense Technology, Changsha, China. His research interests focus on network and cloud computing security.

Shaojing Fu received his Ph.D. degree in applied cryptography from National University of Defense Technology in 2012 and has studied at University of Tokyo for one year during his Ph.D. He is currently a lecturer in the College of Computer, National University of Defense Technology. His research interests include cryptography theory and application, cloud storage security.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation computer systems 25 (6) (2009) 599–616.

[2] C. Cachin, I. Keidar, A. Shraer, Trusting the cloud, Acm Sigact News 40 (2) (2009) 81–86.

[3] K. Ren, C. Wang, Q. Wang, et al., Security challenges for the public cloud, IEEE Internet Computing 16 (1) (2012) 69–73.

[4] A. Juels, B. S. Kaliski Jr, Pors: Proofs of retrievability for large files, in: Proceedings of the 14th ACM conference on Computer and communications security, ACM, 2007, pp. 584–597.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable data possession at untrusted stores, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS '07, ACM, New York, NY, USA, 2007, pp. 598–609.

[6] K. D. Bowers, A. Juels, A. Oprea, Proofs of retrievability: Theory and implementation, in: Proceedings of the 2009 ACM workshop on Cloud computing security, ACM, 2009, pp. 43–54.

[7] Q. Wang, C. Wang, J. Li, K. Ren, W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing, in: Proceedings of the 14th European Conference on Research in Computer Security, ESORICS'09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 355–370.

[8] C. Wang, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for data storage security in cloud computing, in: INFOCOM, 2010 Proceedings IEEE, 2010, pp. 1–9.

[9] C. Wang, S. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, Computers, IEEE Transactions on 62 (2) (2013) 362–375.

[10] J. Yuan, S. Yu, Proofs of retrievability with public verifiability and constant communication cost in cloud, in: Proceedings of the 2013 international workshop on Security in cloud computing, ACM, 2013, pp. 19–26.

[11] J. Yuan, S. Yu, Secure and constant cost public cloud storage auditing with deduplication, in: Communications and Network Security (CNS), 2013 IEEE Conference on, IEEE, 2013, pp. 145–153.

[12] B. Wang, B. Li, H. Li, Oruta: Privacy-preserving public auditing for shared data in the cloud, in: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, 2012, pp. 295–302.

[13] B. Wang, B. Li, H. Li, Knox: privacy-preserving auditing for shared data with large groups in the cloud, in: Applied cryptography and network security, Springer, 2012, pp. 507–525.

[14] B. Wang, H. Li, M. Li, Privacy-preserving public auditing for shared cloud data supporting group dynamics, in: Communications (ICC), 2013 IEEE International Conference on, 2013, pp. 1946–1950.

[15] B. Wang, B. Li, H. Li, Public auditing for shared data with efficient user revocation in the cloud, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 2904–2912.

[16] J. Yuan, S. Yu, Efficient public integrity checking for cloud data sharing with multi-user modification, in: INFOCOM, 2014 Proceedings IEEE, 2014, pp. 2121–2129.

[17] A. Shamir, How to share a secret, Communications of the ACM 22 (11) (1979) 612–613.

[18] M. Blaze, G. Bleumer, M. Strauss, Divertible protocols and atomic proxy cryptography, in: Advances in Cryptology-UROCRYPT'98, Springer, 1998, pp. 127–144.

[19] S. S. Chow, R. C. Phan, Proxy re-signatures in the standard model, in: International Conference on Information Security, 2008, pp. 260–276.

[20] G. Ateniese, S. Hohenberger, Proxy re-signatures: new definitions, algorithms, and applications, in: Proceedings of the 12th ACM conference on Computer and communications security, ACM, 2005, pp. 310–319.

[21] Y. Wang, Q. Wu, D. S. Wong, B. Qin, S. S. Chow, Z. Liu, X. Tan, Securely outsourcing exponentiations with single untrusted program for cloud storage, in: European Symposium on Research in Computer Security, Springer, 2014, pp. 326–343.

[22] X. Chen, W. Susilo, J. Li, D. S. Wong, J. Ma, S. Tang, Q. Tang, Efficient algorithms for secure outsourcing of bilinear pairings, Theoretical Computer Science 562 (C) (2015) 112–121.

[23] P. P. Tsang, S. S. M. Chow, S. W. Smith, Batch pairing delegation, Advances in Information and Computer Security IWSEC 2007 4752 (2006) (2007) 74–90.

[24] S. Hohenberger, A. Lysyanskaya, How to securely outsource cryptographic computations, in: Theory of Cryptography Conference, Springer, 2005, pp. 264–282.

[25] Pairing based cryptography (pbc) library, https://crypto.stanford.edu/pbc/ (2017).

[26] G. Ateniese, R. Di Pietro, L. V. Mancini, G. Tsudik, Scalable and efficient provable data possession, in: Proceedings of the 4th international conference on Security and privacy in communication netowrks, ACM, 2008, p. 9.

[27] H. Shacham, B. Waters, Compact proofs of retrievability., in: Asiacrypt, Vol. 5350, Springer, 2008, pp. 90–107.

[28] B. Dan, B. Lynn, H. Shacham, Short signatures from the weil pairing, Journal of Cryptology 17 (4) (2004) 297–319.

[29] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, in: Proceedings of the 16th ACM conference on Computer and communications security, ACM, 2009, pp. 213–222.

[30] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, R. Kotagiri, Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates, IEEE Transactions on Parallel and Distributed Systems 25 (9) (2014) 2234–2244.

[31] Y. Zhu, H. Hu, G.-J. Ahn, M. Yu, Cooperative provable data possession for integrity verification in multicloud storage, Parallel and Distributed Systems, IEEE Transactions on 23 (12) (2012) 2231–2244.

[32] B. Wang, S. S. Chow, M. Li, H. Li, Storing shared data on the cloud via security-mediator, in: Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on, IEEE, 2013, pp. 124–133.

[33] S. S. Chow, C.-K. Chu, X. Huang, J. Zhou, R. H. Deng, Dynamic secure cloud storage with provenance, in: Festschrift Jean-Jacques Quisquater, Springer-Verlag Berlin Heidelberg, 2011, pp. 442–464.

Table 1: An Example of Dynamic Data Table

a. Initial State					**b. Data Update Operation**				
No.	B	V	E	S	No.	B	V	E	S
1	1	0	0	s_1	1	1	0	0	s_1
2	2	0	0	s_2	2	2	0	0	s_2
3	3	0	0	s_3	3	3	1	0	s_3'
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	n	0	0	s_n	n	n	0	0	s_n
c. Data Insertion Operation					**b. Data Deletion Operation**				
No.	B	V	E	S	No.	B	V	E	S
1	1	0	0	s_1	1	1	0	0	s_1
2	2	0	0	s_2	Null	2	−1	0	s_2
3	2	0	1	s_2'	2	2	0	1	s_2'
4	3	1	0	s_3'	3	3	1	0	s_3'
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n + 1	n	0	0	s_n	n	n	0	0	s_n

Table 2: The error detection probability

φ/n	3%	2%	1%	0.5%
P_χ				
0.95	99	149	299	598
0.97	116	174	349	700
0.99	152	228	459	919

Table 3: A brief comparison among previous works and our scheme

	Communication Cost		Computation Complexity		Storage Overhead
	Challenge	Proof	Revocation	Verification	Signatures Storage
Wang's[15]	$c(\|n\| + \|q\|)$	$2d\|p\| + c\|id\|$	$rEXP_{G_1}$	$(c+d)EXP_{G_1} + (c+2d)MUL_{G_1} + (d+1)Pair + dMUL_{G_2}$	$\|G_1\|\|M\|/(\|Block\|)$
Yuan's[16]	$c\|n\| + (d+1)\|G_1\|$	$\|G_1\| + \|G_2\|$	$rEXP_{G_1} + rPair$	$1EXP_{G_1} + 2Pair$	$\|G_1\|\|M\|/(s\|Seg\|)$
Our Scheme	$c\|n\| + (s+1)\|G_1\|$	$d\|G_1\| + \|G_2\|$	$rEXP_{G_1}$	$dEXP_{G_1} + cMUL_{G_1} + dPair + (d-1)MUL_{G_2} + 2EXP_{G_2}$	$\|G_1\|\|M\|/(s\|Seg\|)$

Notes: $|q|$ is the size of one element in $\mathbb{Z}_q$, $|id|$ is the size of a block identifier in [15], r is the number of blocks signed by the revoked user, $|Block|$ is the size of one data block in [15], which is comparable to $|Seg|$, the segment size in [16] and our scheme. The item of user revocation in this table only involves the computation of re-signatures but the generation of re-signing keys.

Table 4: Comparison of primary scheme and extensions

	Comp.Cost (Client Side)
Signatures without Outsourcing	$(s+1)EXP_{G_1} + sMUL_{G_1}$
Signatures with Outsourcing	$2sMUL_{\mathbb{Z}_p} + 2(d-1)SUB_{\mathbb{Z}_p} + 3EXP_{G_1} + 3MUL_{G_1}$
Verification without Outsourcing	$dEXP_{G_1} + cMUL_{G_1} + dPair + (d-1)MUL_{G_2} + 2EXP_{G_2}$
Verification with Outsourcing	$dEXP_{G_1} + 3dMUL_{G_1} + (2d+1)EXP_{G_2} + 2dMUL_{G_2}$

Table 5: Comparison between outsourcing signatures computation and local signatures computations

Segments	Signatures with Outsourcing (ms)	Signatures without Outsourcing (ms)	speedup
50	36	74	2.06
100	67	145	2.18
150	97	218	2.23
200	128	290	2.26
250	159	362	2.27
300	185	435	2.35
350	215	507	2.36
400	245	579	2.37
450	275	652	2.37
500	305	723	2.37