A Secure Light-Weight Public Auditing Scheme in Cloud Computing with Potentially Malicious Third Party Auditor

Yilun Wu†a, Xinye Lin††b, Xicheng Lu†††, Jinshu Su†, Nonmembers, and Peixin Chen††c, Student Member

SUMMARY Public auditing is a new technique to protect the integrity of outsourced data in the remote cloud. Users delegate the ability of auditing to a third party auditor (TPA), and assume that each result from the TPA is correct. However, the TPA is not always trustworthy in reality. In this paper, we consider a scenario in which the TPA may lower the reputation of the cloud server by cheating users, and propose a novel public auditing scheme to address this security issue. The analyses and the evaluation prove that our scheme is both secure and efficient.

key words: cloud security, public auditing, data integrity, storage service

1. Introduction

An explosive growth of cloud computing has been witnessed in the past few years. As the cloud services gaining more and more users and storing a huge amount of data, the integrity of the data becomes a major concern [1]. Actually, it has already been shown that various hardware/software failures and human errors that exist in the cloud can put the data integrity at high risk [2].

In order to better protect the data in the cloud, Wang et al. proposed two schemes [3], [4], referred to as public auditing, that enable users to audit the data integrity without maintaining local copies. In these schemes, most computation operations are performed by a third party auditor (TPA). After that, Wang et al. [5] proposed an scheme that guarantees a secure public auditing, in a privacy-preserving way. Subsequently, some schemes [6], [7] were proposed to support dynamic data operations. Considering the scenario of multiple users, two schemes [8], [9] were proposed to enable multiple users to audit the shared data. Nevertheless, all schemes mentioned assume that the TPA is trusted, which is not always true in reality. Xu [10] considered an untrusted TPA, and introduced two trusted servers to design a scheme that allows users to check the behaviors of the TPA. Based on [10], Huang et al. [11] proposed a scheme that defends against the collusion attack. Unfortunately, all the aforementioned schemes fail to consider the scenario in which the TPA damage the reputation of the cloud server by cheating users with the forged auditing results.

To address this security problem, we propose a novel public auditing scheme that assures a secure public auditing with a potentially malicious third party auditor. Unlike most existing solutions, in our scheme, the last step of the auditing will be performed by the user only, which implies that the TPA is no longer a verifier. Instead, the TPA will help the user execute most computation operations, and it returns computational results, referred to as auditing factors in this paper, back to the user. With the auditing factors, the user can efficiently verify the integrity of the data. Our scheme can still guarantee a secure public auditing when the TPA deceives the user by returning faked auditing factors.

Specifically, the main contribution of our scheme is threefold. First, we proposed a novel scheme that guarantees a secure public auditing with the potentially malicious TPA. In our solution, the TPA will undertake most computation operations as before except verifying the data integrity. Only the user can verify the data integrity. Second, our scheme is a light-weight solution that involves no additional entity. Third, we comprehensively prove the correctness and the security of the proposed scheme. In addition, the experimental evaluation proves that the computation workload of the user is very small.

2. Statement and Preliminaries

2.1 System Model

As illustrated in Fig. 1, there are three entities in our system: the user, the cloud server and the third party auditor.
(TPA). The user has some data to be outsourced to the remote cloud server, which provides the storage service. Before outsourcing the data, the user divides the data into many blocks, and calculates a signature for each block. After outsourcing these blocks and the corresponding signatures to the cloud server, the user can send an auditing request to the TPA. Upon receiving the request, the TPA generates a challenge message and sends it to the cloud server. The cloud server then generates a proof message accordingly, and return it to the TPA. After that, the TPA calculates the auditing factors and returns the factors to the user. Finally, the user can easily verify the data integrity with the auditing factors and the symmetric key. Unlike existing schemes, the user in our scheme is not only a data sender, but also a verifier. The last step of verification is performed solely by the user. To save the computation resources of the user, the TPA undertakes main computation operations, including challenge message generation and auditing factors calculation.

2.2 Security Model

We assume that the cloud server is untrusted. It means that the cloud server may try to forge an invalid proof message to pass the verification. Our scheme should guard against this malicious behavior. Meanwhile, the cloud server may be a victim. The TPA may forge the results to damage the reputation of the cloud server.

The TPA in our scheme is a potentially malicious entity. Since our solution has removed the verification ability from the TPA, the TPA only generates the challenge message and calculates the auditing factors. We consider that the TPA may return the faked auditing factors to the user for harming the reputation of the cloud server.

2.3 Design Goals

Our scheme should achieve: Functionality. Since the TPA is not a verifier anymore, the public auditing in our scheme means that the TPA will still take main computation operations. Security. Our scheme is not only able to prevent the cloud server from passing the verification with forged proof message, but also able to protect the reputation of the cloud server from being damaged by the TPA. Performance. The performance of our scheme should be acceptable.

2.4 Preliminaries

Bilinear Map: Let $G_1$ and $G_2$ be two finite cyclic groups of a large prime $p$, $g$ be a generator of $G_1$. A map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map if satisfying the following three properties.

1) Bilinearity: for all $u, v \in G_1$ and all $a, b \in \mathbb{Z}_p$, a map $\hat{e}$ is bilinear if $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$. 2) Non-degeneracy: $\hat{e}(g, g) \neq 1$. If $g$ is a generator of $G_1$, then $\hat{e}(g, g)$ is a generator of $G_2$. 3) Computability: for any $u, v \in G_1$, there is an efficient algorithm to compute $\hat{e}(u, v)$.

Diffie-Hellman Key Agreement: User Alice and user Bob agree on a group $G$ of a large prime $p$, and $g$ is a generator of $G$. Then, Alice selects a random secret number $a \in \mathbb{Z}_p$, and sends $g^a$ to Bob. Meanwhile, Bob selects a random secret number $b \in \mathbb{Z}_p$, and sends $g^b$ to Alice. Finally, both Alice and Bob can generate a key $K = (g^a)^b = (g^b)^a$.

DL problem assumption: Let $G$ be a finite cyclic group with a generator $g$ and a large prime number $p$. For any probabilistic polynomial time (PPT) adversary who has $p, g, g^a$, the probability to find $a$ is negligible in $G$.

DCDH problem assumption: Let $G$ be a finite cyclic group with a generator $g$ and a large prime $p$. For any PPT adversary who has $p, g, g^a, g^b$, where $g^a, g^b \in G$, the probability to compute $g^{a+b}$ is negligible in $G$.

3. Details of the Proposed Scheme

Our scheme contains three phases: System Setup, Auditing Factors Calculation and Data Integrity Verification.

3.1 System Setup

Figure 2 shows the main steps involved in this phase.

Initialization: The user generate $G_1$, $G_2$ and $\hat{e}$, where $G_1$ and $G_2$ are two finite cyclic groups of large prime $p$. $g$ is a generator of $G_1$. $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map.

$H : \{0, 1\}^n \rightarrow G_1$ is a hash function. $u$ is a random value in $G_1$. The public parameter $pk = \{G_1, G_2, \hat{e}, g, u, H\}$. Then, the user selects a random number $a \in \mathbb{Z}_p$ as her own secret key, and sets the public key $A$ as $g^a$. Similarly, the cloud server can generate his secret key $b$ and public key $g^b$.

Signature Generation: The user divide the file $F$ into $n$ blocks $m_1, m_2, \cdots, m_n$, where $m_i \in \mathbb{Z}_p$. For each block $m_i$, the user calculates the corresponding signature:

$$\sigma_i = (H(id_i) \cdot u^{|m_i|})^{a_i},$$

where $id_i$ is the identifier of $m_i$. The signatures can be denoted as a set $\Phi = \{\sigma_i\}$, where $1 \leq i \leq n$.

Outsourcing: The user sends $\{F, \Phi, \{id_i\}_{1 \leq i \leq n}\}$ to the cloud server, and deletes them from the local storage. For each received signature, the cloud server calculates $\sigma_i' = \sigma_i^b$, and generates a new signature set $\Phi' = \{\sigma_i' = \sigma_i^b\}_{1 \leq i \leq n}$.

Key Agreement: Simultaneously, the user and the
cloud server exchange their public keys to compute the symmetric key $K$ based on Diffie-Hellman key agreement protocol. The security of $K$ will be discussed in Sect. 4.1.

3.2 Auditing Factors Calculation

As shown in Fig. 3, in this phase, the TPA generates a challenge message, referred to as $chal$, and sends $chal$ to the cloud server. On receiving $chal$, the cloud server can calculate a response called $proof$, and sends $proof$ to the TPA. Finally, the TPA generates the auditing factors and sends them to the user.

$chal$ generation. To verify the data integrity, the TPA only needs to randomly select a certain number of blocks instead of all the blocks. This method is called probabilistic framework, which can significantly improve the efficiency with the guarantee of high detection probability of the polluted blocks [2]. The $chal$ message is generated as follows:

- Randomly select a $c$-element subset $I = \{s_1, s_2, \cdots, s_c\}$ from $[1, \cdots, n]$ to locate the $c$ blocks which will be verified. For each $i, j, s_i \neq s_j$ if $i \neq j$.
- $\forall i \in I$, select a random number $v_i \in \mathbb{Z}_p$.
- Denote $chal$ message as $\{i, v_i\}, i \in I$.

$proof$ generation. Upon receiving $chal$ from the TPA, the cloud server calculates $proof = \{\mu, \sigma\}$ as follows.

$$
\mu = \sum_{i=s_1}^{s_c} v_i m_i \in \mathbb{Z}_p, \mu_i = v_i m_i
$$

(2)

$$
\sigma = \prod_{i=s_1}^{s_c} (\sigma_i')^{v_i} \in G_1.
$$

3.3 Data Integrity Verification

In this phase, the user only needs to check Eq. (4) after receiving the auditing factors $E$ and $D$ from the TPA.

$$
E = \hat{\epsilon}(D, K)
$$

(4)

If $E$ is equal to $\hat{\epsilon}(D, K)$, the verification passes. Otherwise, the user will know that some blocks have been polluted. If the verification fails, the user can choose to believe the TPA or retry the verification.

3.4 Verification Retrial

Since the TPA cannot be fully trusted, the TPA may damage the cloud server’s reputation by returning the forged $E$ and $D$ back to the user. If the verification fails, the user can perform verification retrial to check whether the TPA tells lies. More precisely, the user requests $I$ from the TPA, generates a new challenge message, and sends the message to the cloud server. If the verification passed, it proves that the TPA should take the responsibility. Although this phase consumes some additional computation resources of the user, it will help the user confirm the accountable entity.

4. Analysis

4.1 Security Analysis

There are two theorems in this subsection. The first theorem proves that the cloud server cannot pass the verification by forging the $proof$ message. The second theorem proves that the TPA cannot retrieve $K$ according to the information learned, which implies that the TPA cannot forge the valid verification factors.

Theorem 1: It is impossible for the cloud server to forge a $proof$ message and pass the verification with non-negligible probability.

Proof: Previous work [9] shows the main method to prove Theorem 1. Here we only prove that, if the cloud server can forge a $proof$ message to pass the verification with non-negligible probability, then the Discrete Logarithm problem can be solved with non-negligible probability.

Denote $[\mu, \sigma]$ be the correct $proof$ message, when the TPA wants to check the block set $M = \{m_i\}_{1 \leq i \leq d}$. It is evident that $[\mu, \sigma]$ can pass the verification with Eq. (4), if no block is polluted. Assume that the cloud server calculates the forged $proof$ message $[\mu', \sigma']$ based on the incorrect block sets $M' = \{m_i'\}_{1 \leq i \leq d}$, where $\mu' = \mu_i = v_i \cdot m_i'$, and pass the verification in Eq. (4). Define $\Delta \mu_i = \mu_i - \mu_i'$. Due to $M \neq M'$, we can have $\sum_{i=1}^{d} \Delta \mu_i \neq 0$.

Since $[\mu, \sigma]$ is a correct $proof$, the following equation can be computed based on Eq. (3) and Eq. (4).

$$
E = \hat{\epsilon}(\prod_{i=1}^{d} H(id_i)^{v_i} \cdot u^\mu, K) = \hat{\epsilon}(D, K)
$$

(5)

As assumed before, the forged $proof$ can pass the verification, we can have

$$
E = \hat{\epsilon}(\prod_{i=1}^{d} H(id_i)^{v_i} \cdot u^\prime, K) = \hat{\epsilon}(D, K).
$$

(6)
According to Eq. (5) and Eq. (6), we can get Eq. (7).

\[ \prod_{i=1}^{d} u^{\Delta u_i} = \prod_{i=1}^{d} u^{\mu_i} \iff \prod_{i=1}^{d} u^{\lambda_i} = 1. \]  

(7)

For two elements \( x, y \in G_1 \), there exists a number \( a \in \mathbb{Z}_p \) that \( x = y^a \). Without loss of generality, given \( x, y \), choose \( s, r \in \mathbb{Z}_p \), let \( u = x^s \cdot y^r \). According to the properties of multiplicative cycle group, there exists a number \( t \in \mathbb{Z}_p \), that \( u = u^t = x^u \cdot y^v \). Let \( st = \xi \), and \( rt = \lambda \). Thus, \( u \) can be represented as \( x^\xi \cdot y^\lambda \). We have

\[ 1 = \prod_{i=1}^{d} u^{\Delta u_i} = \prod_{i=1}^{d} x^{\xi \Delta u_i} \cdot y^{\lambda \Delta u_i} = x^{\sum_{i=1}^{d} (\xi \Delta u_i)} \cdot y^{\sum_{i=1}^{d} (\lambda \Delta u_i)}. \]  

(8)

Then, we can calculate

\[ x = y^{\frac{-\sum_{i=1}^{d} (\lambda \Delta u_i)}{\sum_{i=1}^{d} (\xi \Delta u_i)}} \]

\[ a = -\frac{\sum_{i=1}^{d} (\Lambda \Delta u_i)}{\sum_{i=1}^{d} (\xi \Delta u_i)} = -\frac{\lambda}{\xi}. \]  

(9)

Equation (9) implies that \( a \) can be computed if the denominator equals to zero is that \( \xi = 0 \) with a probability of \( 1/p \). It means that one solution can be found to calculate \( a \) with probability of \( (1 - 1/p) \), which is a non-negligible probability. In conclusion, if the cloud server can forge a proof message and pass the verification, then one solution can be found to solve the Discrete Logarithm problem with non-negligible probability, which conflicts with the DL assumption.

Theorem 2: It is impossible for the TPA to get the symmetric key \( K \) with non-negligible probability in the proposed scheme.

Proof: According to Sect. 4.2, the TPA has \( D \) and \( E \). Assume the TPA knows the verification equation and can get the symmetric key \( K \) with non-negligible probability. It means that the TPA can feasibly retrieve \( K \) from \( E = \hat{e}(D, K) \), which equivalents to the following T-assumption.

T-assumption: Given \( \{U \in G_1, W \in G_2\}, V \in G_1 \) can be calculated in \( W = \hat{e}(U, V) \) with non-negligible probability.

Without loss of generality, denote \( \hat{U} = g^u, \hat{V} = g^v \) and \( \hat{W} = \hat{e}(g^u, g^v) \). Given \( g, g^u, g^v \in G_1, W \) can be calculated as \( \hat{e}(g, g^u) \). Due to T-assumption, \( V \) can be calculated in \( W = \hat{e}(U, V) \) with non-negligible probability. Then we can find that given \( g, g^u \) and \( g^v \), \( g^a \) can be calculated if T-assumption is correct. It is obvious that T-assumption is equivalent to a solution of DCDH problem, which contradicts the DCDH problem assumption. Thus the T-assumption is not correct. The proof of Theorem 2 is completed. \( \square \)

4.2 Theoretical Performance Analysis

We compare the computation costs on both side of the user and the TPA among our scheme, Wang et al’s scheme [5] and Huang et al’s scheme [11]. We assume that only one challenge message has been generated, and illustrate the corresponding results in Table 1. As shown in Table 1, the user only performs one pairing operation during the auditing in our scheme. Moreover, the TPA has minimal computation overhead in our scheme. In addition, our scheme considers a potential malicious TPA which tries to damage the cloud server’s reputation. Comparing with the scheme in [11], our scheme is more efficient with less entities. It is worth noting that our scheme does not introduce any entity.

4.3 Performance Evaluation

We conduct the experiment on a laptop running Ubuntu Linux with 2.5 GHz Intel Core i5 processor and 4 Gigabyte memory. The implementation of the experiment is based on PBC Library [13]. We adopt the type A elliptic curve of 160-bit group order to build the cryptographic environment.

Table 2 illustrates the computational costs of the user. Each result is an average value from 1000 runs. The experimental results show that the verification on the user side is very fast. The results in Table 2 also proves that the number of selected blocks has no relation to the user’s costs.

According to Table 1, the computation cost of the TPA depends on the number of selected blocks. We run the experiment for 10 times and show the average values in Table 3, which shows that the computation cost is linear to the number of selected blocks. Our scheme is very efficient because the TPA only consumes 1.871 second if 500 blocks are selected.

| Scheme         | User’s overhead | TPA’s overhead | Anti Malicious TPA | Additional Entities |
|----------------|-----------------|----------------|-------------------|---------------------|
| Wang [5]       | Small           | \((2+c)M+(1+c)E+2P\) | No                | No                  |
| Huang [11]     | \(2cM+3E+1TEnc\) | \((2+c)M+(3+c)E+2P+1Div+1TDec\) | Yes               | Yes                 |
| ours           | 1P              | \(cM+(1+c)E+1P\) | Yes               | No                  |

Notes: \( M \) denotes one multiplication operation, \( E \) denotes one exponentiation operation, \( P \) denotes one pairing operation, \( Div \) denotes one division operation, \( TEnc \) is the timed-release encryption [10], \( TDenc \) is the timed-release decryption [10], \( c \) is the number of sampled blocks.

| Selected blocks | Execution time (s) |
|-----------------|--------------------|
| 100             | 0.348              |
| 200             | 0.684              |
| 300             | 1.025              |
| 400             | 1.359              |
| 500             | 1.871              |

| Selected blocks | Execution time (ms) |
|-----------------|---------------------|
| 100             | 4.926               |
| 200             | 4.328               |
| 300             | 4.357               |
| 400             | 4.325               |
| 500             | 4.300               |

Table 1 Theoretical performance comparison with one Chal

Table 2 The execution time of the verification on the user’s side

Table 3 The execution time of the verification on the TPA’s side
5. Conclusion and Future Works

In this paper, we consider a more practical scenario that the TPA is untrusted. And we propose a novel public auditing scheme to solve the corresponding security issue. Both our theoretical and performance analysis prove that: 1) the scheme can audit the integrity of data correctly. If any mistake or malicious behavior happened on the data, our scheme will detect the polluted data with very high probability; 2) the security of our scheme satisfies the requirements of the security model. The cloud server is unable to forge the message. The TPA cannot forge the valid auditing factors without the symmetric key; 3) the performance analysis proves that our solution to be efficient and practical. In the future, we will focus on accountability, which implies that the capability of the system to identify whether an entity is responsible for specific events with undeniable evidence. It is significant because any entity, including the data user, may execute malicious behaviors.

References

[1] K. Ren, C. Wang, and Q. Wang, “Security Challenges for the Public Cloud,” IEEE Internet Comput., vol.16, no.1, pp.69–73, 2012.
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” Proc. 14th ACM CCS, Alexandria, U.S.A, pp.598–610, 2007.
[3] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, “Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing,” Proc. 14th ESORICS, Saint Malo, France, vol.5789, pp.355–370, 2009.
[4] C. Wang, Q. Wang, K. Ren, and W. Lou, “Ensuring Data Storage Security in Cloud Computing,” Proc. 17th ACM/IEEE IWQoS, Charleston, South Carolina, pp.1–9, 2009.
[5] C. Wang, Q. Wang, K. Ren, and W. Lou, “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,” Proc. 29th INFOCOM, San Diego, USA, pp.525–533, 2010.
[6] Y. Zhu, G.-J. Ahn, H. Hu, S.S. Yau, G.H. An, and C.-J. Hu, “Dynamic Audit Services for Outsourced Storages in Clouds,” Proc. IEEE Transactions on Services Computing, vol.6, no.2, pp.227–238, 2013.
[7] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S.S. Yau, “Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds,” Proc. 26th SAC, Taichung, Taiwan, pp.1550–1557, 2011.
[8] J. Yuan and S. Yu, “Efficient Public Integrity Checking for Cloud Data Sharing with Multi-User Modification,” Proc. 33rd INFOCOM, Toronto, Canada, pp.2121–2129, 2014.
[9] B. Wang, B. Li, and H. Li, “Public Auditing for Shared Data with Efficient User Revocation in the Cloud,” Proc. 32nd IEEE INFOCOM, Turin, Italy, pp.2904–2912, 2013.
[10] J. Xu, “Auditing the Auditor: Secure Delegation of Auditing Operation over Cloud Storage,” IACR Cryptology ePrint 201: 304, 2014.
[11] K. Huang, M. Xian, S. Fu, and J. Liu, “Securing the Cloud Storage Audit Service: Defending Against Frame and Collude Attacks of Third Party Auditor,” Communications, IET, vol.8, no.12, pp.2106–2113, 2014.
[12] W. Diffie and M. Hellman, “New Directions in Cryptography,” Proc. IEEE Trans. on Information Theory, vol.22, no.6, pp.644–654, 1976.
[13] “Pairing-based Cryptographic Library,” https://crypto.stanford.edu/pbc/