International Journal of Database Theory and Application
Vol.7, No.6 (2014), pp.139-150
http://dx.doi.org/10.14257/ijdta.2014.7.6.13

Bacthing Auditing of Data in Multicloud Storage

Zhihua Xia, Xinhui Wang, Xingming Sun,Yafeng Zhu, Peng Ji and Jin Wang

Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, 210044, China
School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China

Abstract

Cloud storage enables users to outsource their data to cloud servers and enjoy the on-demand services. However, this new paradigm also introduces integrity threats toward user' outsourced data. This paper develops an efficient auditing mechanism, which support batch auditing for multiple data files in multi-cloud environment. By constructing a sequence-enforced Merkle Hash Tree, the proposed protocol can resist the replace attack. By using the bilinear map, the proposed protocol achieves stateless and transparent verification. By putting the computation of intermediate values of the verification on cloud servers, our method can greatly reduce the computing burden of the auditor. The performance analysis proves the good efficiency of the proposed protocol.

Keywords: *Multicloud, cooperative, batch auditing, PDP, BLS signature, homomorphic authenticators*

1. Introduction

Cloud storage service is an important service of cloud computing, which has become a new profit growth point by relieving individuals' or enterprises' burden for storage management and maintenance. By remotely storing data into the cloud, users can access their data via networks at anytime and from anywhere. However, many users are still hesitant to use this novel paradigm due to the security and integrity threats toward their outsourced data. This is because data loss could occur in any infrastructure, whatever high degree of reliable measures the cloud service providers (CSPs) would take [1]. Moreover, the CSPs could be dishonest. They may hide data loss accidents to maintain the reputation, or even discard the data that has not been or rarely accessed to save the storage space and claim the data is still correctly stored in the cloud. Therefore, it is highly essential for the cloud users to check the integrity and availability of their cloud data.

In order to address the issue above, various Provable Data Possession (PDP) protocols have been proposed [2-4]. PDP is a probabilistic proof technique for checking the availability and integrity of outsourced data with randomly sampling a few file blocks. Ateniese *et al.* [2] first defined PDP model with public verification. They utilized RSA-based homomorphic linear authenticators (HLA) and suggested randomly sampling a few blocks of the file for verification. In order to support dynamic operations, Ateniese *et al.*[3] developed a partially dynamic version of scalable PDP model based on symmetric key cryptography. After that, Erway *et al.* [4] presented a skip list-based dynamics PDP model with fully data dynamic operation. Wang *et al.* also proposed a dynamics PDP scheme based on combining Boneh–Lynn–Shacham signature (BLS)-based HLA with Merkle Hash Tree (MHT) structure [5, 6]. The scheme supports both the public stateless verification and fully dynamic data update. Their subsequent work [7] proposed a scheme supporting privacy-preserving public auditing, which was also extended to enable batch auditing.

To handle integrity auditing issue for multicloud storage, Zhu *et al.* [10, 11] first proposed a cooperative PDP (CPDP) scheme based on the techniques of fragment structure, hash index

ISSN: 2005-4270 IJDTA

hierarchy, and homomorphic verifiable response. In their scheme, one of CSPs takes the responsibility as the organizer to cooperate the communication between CSPs and verifier. This scheme supports the efficient verification of data stored on multicloud. However, it is not a stateless verification protocol because the verifier needs to store and update verification information. In addition, it is proved by Wang and Zhang [12] that, the malicious CSPs or organizer in this scheme even be able to pass validation without storing users' data. After that, Etemad *et al.* [13] proposed a distributed and replicated PDP scheme which could enable CSPs to hide the internal structure from the client. This scheme is only designed to audit data files one by one. With the wide adoption of cloud computing, the auditor may concurrently receive multiple auditing delegations from different users. If the multiple tasks can be handled in a batch manner, tprotocolhe efficiency will be greatly improved. Yang *et al.* [14] proposed a privacy-preserving auditing protocol which could deal with the verification tasks from different users in a batch manner in multicloud scenario. However, all of the PDP protocols for multicloud scenario above [10-14] cannot be regarded as the full stateless and transparent verification, because these protocols need to store the position information of the data blocks on the organizer or TPA.

In this paper, we develop an efficient auditing mechanism, which support batch auditing for multiple data files in multi-cloud environment. By utilizing the bilinear map, the proposed protocol can aggregate the verification task from different users to reduce the computing overhead of the auditor. By constructing a sequence-enforced Merkle Hash Tree, the proposed protocol can resist the replace attack.

The rest of the paper is organized as follows. Section 2 presents the system model, threat model, design goals, and the preliminaries. Section 3 describes the proposed protocol in detail. We provide security analysis and performance evaluation in Section 4. Section 5 concludes the proposed protocol.

2. Problem Statement

2.1 The System Model

As illustrated in Figure 1, we consider a multicloud storage service model which is adopted by some previous works [10, 11]. The model involves three different entities cloud users, multicloud and third party auditor.

The cloud users have a number of data files to be stored in multiple clouds. They have the authority to access and manipulate the stored data.

The multi-cloud consists of multiple Cloud Server Providers (CSPs). They provide data storage service and have enough storage space and significant computation resources. In this paper, to reduce the communication burden of verifier, one of CSPs is designated as an organizer for auditing purpose. For example, the Zoho cloud in Figure 1 is considered as an organizer. In our scheme, the organizer takes the responsibility to distribute the auditing challenge and aggregate the proof from multiple clouds. In this paper, we suppose that the CSPs cannot communicate with each other apart from the organizer for the auditing issues, and also, the verifier can only contact with the organizer.

The Third Party Auditor (TPA) has a more powerful computation and communication ability than regular cloud users. In cloud storage system, none of cloud service providers or users could be guaranteed to provide unbiased auditing result. Thus, third party auditing is a natural choice. Moreover, by resorting to TPA, users can be relieved from the burden of checking the integrity of outsource data.

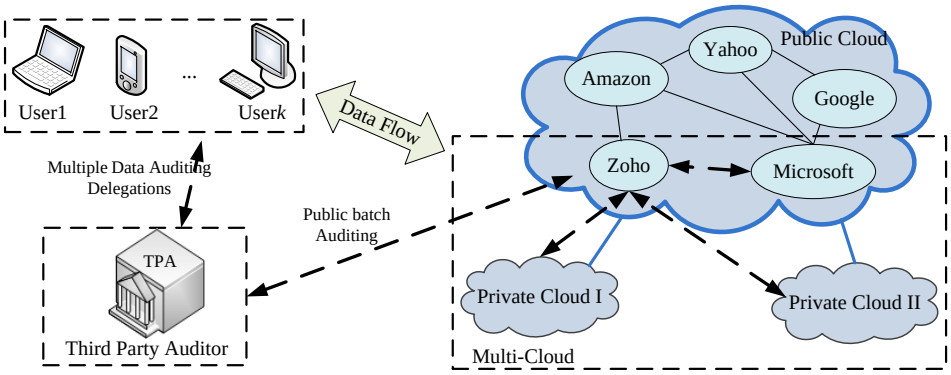


Figure 1. System Model of the Data Storage Auditing

2.2 Notations and Preliminaries

Bilinear map. Let G_1, G_2 be Gap Diffie-Hellman (GDH) groups of prime order p, and let G_T be another multiplicative cyclic group of prime order p. Let g_1, g_2 be generators of G_1 and G_2 respectively. For $u \in G_1, v \in G_2$, and all $a, b \in Z_p$, a bilinear map is a map $e: G_1 \times G_2 \to G_T$ with three properties [6]: 1) Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$; 2) Computable: There exists an efficient computable algorithm for computing $e(u, v)$; 3) Non-degenerate: $e(g_1, g_2) \neq 1$.

The key idea of the batch auditing is to use the bilinear map which could aggregate the signatures. For any $u_1, u_2 \in G_1, v \in G_2$, we have

$$e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v). \tag{1}$$

Sequence-enforced Merkle Hash tree (SMHT). A Merkle Hash Tree (MHT) [15] is a tree structure designed to authenticate a set of data efficiently. It is usually a binary tree with the hashes of authentic data values as its leaves. Each internal node in a MHT stores a hash value of the combination of its child nodes' value. Let $H(\cdot)$ and $h(\cdot)$ be a secure map-to-point hash function. Here, the function $H(\cdot)$ is used to compute the hash values of the data blocks. In this paper, we treat $\{T_i = H(m_i)\}_{i \in [1,n]}$ as the authentic data values, which is arranged to be a left-to-right sequence. The function $h(\cdot)$ can take more than one argument as $h(x_1, x_2) = h(x_1 \| x_2)$, where the operator || denotes concatenation. In this paper, we use a sequence-enforced Merkle Hash Tree (SMHT), which is an improved version of MHT structure as illustrated in Fig. 2. In our protocol, each node of SMHT carries two auxiliary values: a hash value and an index. The computation of hash value in a SMHT is similar to a common MHT. The index of a SMHT's node indicates the total leaf nodes in its subtree. Specifically, the index of leaf node is set to be 1 to inflect itself. Accordingly, if the left child a and right child b of the node r carried auxiliary values (h_a, n_a) and (h_b, n_b) respectively, the index of the node r is $n_r = n_a + n_b$, and the hash of it is computed as $h_r = h(h_a \| h_b \| n_r)$. Figure 2 depicts an example of a simple SMHT which have four data blocks. With the SMHT, we can verify both the value and the position of chosen data block. Figure 3 shows the process of verifying the 3th data block m_3 in Figure 2.

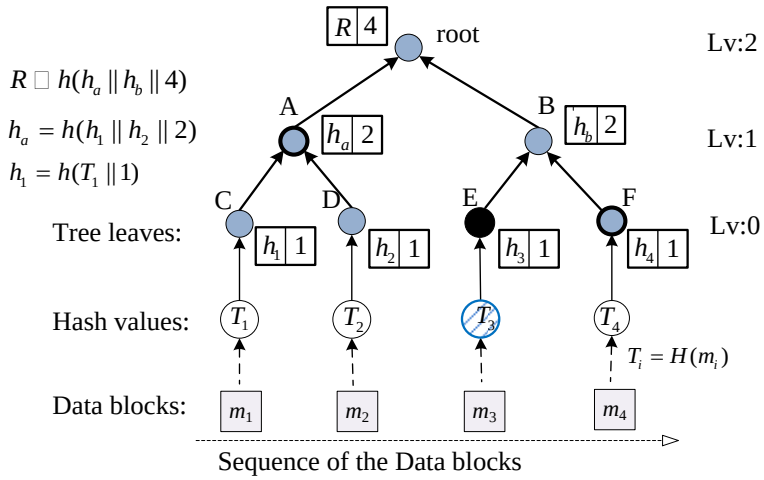


Figure 2. Sequence-enforced Merkle Hash Tree Authentication of Data Blocks

To verify the value and position of data block m_3, we use root $(R, 4)$ and $\Omega = \{((h_4, 1), 0), (h_a, 2), 1\}$.

Step 1: Compute relative index of node 'B' as $1+1=2$ and $h_b = h(h(T_3 \| 1) \| h_4 \| 2)$;

Step 2: Compute relative index of root as $2+2=4$ and $R' = h(h_a \| h_b \| 4)$;

Step 3: Verify if $R' = R$ and also if $LEFT(T_3) = 2$.

Note: the symbol Ω denotes auxiliary authentication information (AAI) used in verification process. The elements of Ω are the brothers of the tree nodes on the path from the verified data block to the tree root. And each element in Ω includes an indicator to indicate the whether corresponding node is the left or right child of its parent. Here, we use the '1' to indicate the left sibling node and the '0' to indicate the right one. The function $LEFT(\cdot)$ signifies the sum of the relative indexes of nodes which is the left sibling node in Ω.

Figure 3. Verify the Value and Position of T_3 in Fig. 2

3. The Batch Auditing CPDP Protocol

For the sake of clarity, Table1 lists some signals and descriptions used later in this paper. As illustrated in Figure 4, our batch auditing protocol for multicloud storage consists of two phases: setup phase and batch audit phase. During the setup phase, the users preprocess the data files to be stored in the multicloud by the algorithms *KeyGen*() and *TagGen*(). The batch auditing phase is run among CSPs, the organizer and TPA to complete the multiple auditing requests for distinct data files simultaneously.

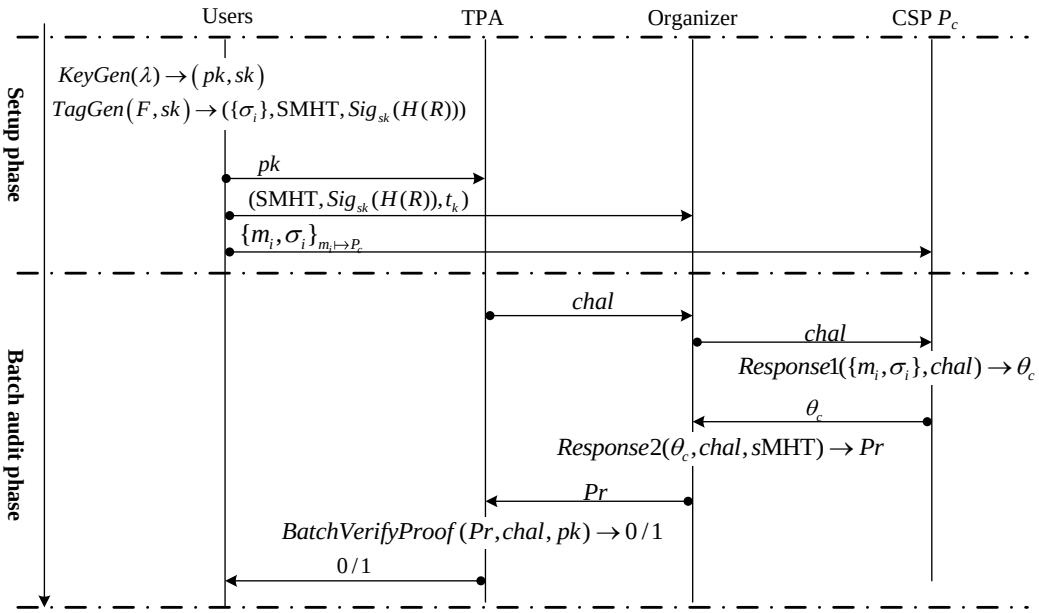


Figure 4 The Framework of our Batch Auditing Protocol

The data fragment technique is a data structure that keeps a set of block-tag. With the fragment structure, an outsourced file F is split into n blocks $\{m_1, m_2, ..., m_n\}$, and each block is further split into s sectors $\{m_{i,1}, m_{i,2}, ..., m_{i,s}\}$. Each data block has its physical meanings and can be updated dynamically by the users. We only need to generate one data tag for each data block that consists of s sectors.

In the proposed protocol, we need to use a bilinear map group system, two hash functions and a signature function. Let $S = (p, g, G_1, G_2, G_T, e)$ be a bilinear map group system with generator g, where G_1, G_2 and G_T are multiplicative cyclic groups of prime order p, and $e: G_1 \times G_2 \to G_T$ is a bilinear map. Let $H(\cdot): \{0,1\}^* \to G_1$ be a secure map-to-point hash function, $h(\cdot): G_T \to Z_p$ be another hash function which maps element of G_T to Z_p, and $Sig(\cdot)$ be the signature function. Let $\{U_k\}$ be the set of users, and $\{P_c\}$ be the set of CSPs. The number of CSPs is denoted as C.

3.1 Setup Phase

Each user runs setup phase of the protocol as follow:

Step1: $KeyGen(1^\lambda)$. The k^{th} user U_k generates a signing key pair (ssk_k, spk_k). Select a random $\alpha_k \leftarrow Z_p$, and compute $\beta_k \leftarrow g^{\alpha_k}$. Select s random elements $\{u_{k,1}, u_{k,2}, ..., u_{k,s}\} \subset G_1$. To sum up, the secret key is $sk_k = (\alpha_k, ssk_k)$ and the public parameters are $pk_k = (spk_k, \beta_k, g, \{u_{k,j}\}_{1 \le j \le s})$.

Step2: $TagGen(sk_k, F_k, P)$. The user U_k splits his file F_k into $n_k \times s$ sectors $F_k = \{m_{k,i,j}\}_{\forall i \in [1,n_k], l \in [1,s]} \subset Z_p^{n_k \times s}$, where the i^{th} block of user U_k block $m_{k,i}$ consists of s sectors $\{m_{k,i,1}, m_{k,i,2}, ..., m_{k,i,s}\}$. The user computes $t_k = name_k \| n_k \| Sig_{ssk_k}(name_k \| n_k)$ as the file tag for F_k, where $name_k$ is the file name. Next, the user U_k constructs SMHT with a root R_k, where the leave nodes are an ordered hash values of data blocks $\{T_{k,i} = H(m_{k,i})\}_{i \in [1,n_k]}$ as described in part 2.3. The user also signs R_k with the private key α_k:

$$Sig_{\alpha_k}(H(R_k)) = (H(R_k))^{\alpha_k} . \quad (2)$$

For each data block $m_{k,i}(\forall i \in [1,n])$, the user U_k computes a data tag $\sigma_{k,i}$ as

$$\sigma_{k,i} = (T_{k,i} \cdot \prod_{j=1}^{s} u_{k,j}^{m_{k,i,j}})^{\alpha_k} . \quad (3)$$

After all the parameters have been prepared, the user U_k distributes the data block and tag pairs $(m_{k,i}, \sigma_{k,i})$ to the corresponding cloud service providers. In addition, the user U_k sends the parameters $\{\text{SMHT}, Sig_{\alpha_k}(H(R_k)), t_k\}$ to the organizer, and sends the public parameters pk_k to TPA. After data transmission, the user asks TPA to conduct the confirmation auditing to make sure that their data is correctly stored on all the servers. Once confirmed, the user can choose to delete the local copy of the data blocks apart from the secret key. By now, TPA could run the sampling auditing periodically to check the data integrity for users.

3.2 Batch Audit Phase

Suppose that TPA process K auditing sessions of K distinct data files simultaneously. The data files are stored on C CSPs $(\{P_c\}c \in C)$. The audit phase is executed as follows:

Step1: $GenProof(\{m_i, \sigma_i\}, chal)$. It is an interactive 5-move protocol among CSPs, an organizer (O), and an Auditor (TPA). This process is described in the following.

1) **Retrieve** ($O \rightarrow$TPA): After receiving the verification request, the organizer sends the file tags $\{t_k\}_{k\in[1,K]}$ to the TPA. The TPA recovers $\{name_k\}_{k\in[1,K]}$ and $\{n_k\}_{k\in[1,K]}$ by using public keys $\{sspk_k\}_{k\in[1,K]}$, and verifies all the signatures. The verification quit by emitting *FALSE* if the file name verification fails.
2) **Challenge1** ($O \leftarrow$TPA): If the file name is successful verified, the TPA generates challenge index-coefficient message $Q_k = \{(i, v_i)\}_{i\in I_k}$ for $\forall k \in [1,K]$, where $I_k \subseteq [1,n_k]$ specifies the sampled blocks that will be verified, and $v_i \in Z_p$ is random element. TPA chooses a random element $\gamma \leftarrow Z_p$ and computes the set of challenge stamps $\{H_k = \beta_k^{\gamma}\}_{k\in[1,K]}$. To sum up, TPA generates the challenge as follow, and sent it to the organizer.

$$chal = (\{Q_k\}_{k\in[1,K]}, \{H_k = \beta_k^{\gamma}\}_{k\in[1,K]}). \quad (5)$$

3) **Challenge2** ($P \leftarrow O$): Upon receiving the challenge $chal$ from the TPA, the organizer forwards $chal$ to each P_c.
4) **Response1** ($P \rightarrow O$): For each Q_k, each P_c picks out $Q_{c,k} = \{(i, v_i)\}_{m_{k,i}\mapsto P_c} \subseteq Q_k$. Here, the symbol $m_{k,i} \mapsto P_c$ means that the data block $m_{k,i}$ is stored on P_c. Then, P_c calculates the linear combination of specified data blocks for each file as

$$\mu_{c,k,j} = \sum_{(i,v_i)\in Q_{c,k}} v_i m_{k,i,j} \in Z_p , \quad (6)$$

and generates data proof DP_c and tag proof TP_c as

$$\begin{cases} DP_c = \prod_{j=1}^{s}\prod_{k=1}^{K} e(u_{k,j}, H_k)^{\mu_{c,k,j}} \\ TP_c = \prod_{k=1}^{K}\left(\prod_{(i,v_i)\in Q_{c,k}} (\sigma_{k,i,c})^{v_i} \right) \in G_1 \end{cases}. \quad (7)$$

Finally, P_c returns $\theta_c = (DP_c, TP_c)$ to the organizer.

5) **Response2** ($O \rightarrow$TPA): Upon receiving all the proofs from all the CSPs, the organizer aggregates all of the proof θ_c into a response $\theta = (DP, TP)$, which are calculated as

$$DP = \prod DP_c, \qquad TP = \prod TP_c. \quad (8)$$

In addition, the organizer also provides the TPA with AAI $\{\Omega_{k,i}\}_{i\in I_k, k\in[1,K]}$, which are the siblings of the nodes on the path from the leaves $\{T_{k,i}\}_{i\in I_k, k\in[1,K]}$ to the root R_k of the SMHT. Finally, the organizer responds TPA with proof.

$$Pr = \left(\{\Omega_{k,i}\}_{i\in I_k, k\in[1,K]}, \{Sig_{\alpha_k}(H(R_k))\}_{k\in[1,K]}, \theta \right) \quad (9)$$

Step2: *BatchVerifyProof*(*pk, chal, Pr*). After receiving proof *Pr* from the organizer, TPA starts to verify to proof. First, for each $k \in [1, K]$ and $i \in I_k$, TPA first verifies the position of $\{T_{k,i}\}_{for\ \forall i\in I_k, k\in[1,k]}$ by checking if the equation $LEFT(T_{k,i}) = i - 1$ holds or not. Second, TPA constructs a verification root R_k' by using $\{T_{k,i}, \Omega_{k,i}\}_{i\in I_k}$ for $\forall k \in [1, K]$, and verifies values of R_k' by checking Eq. (10). Third, if the both authentication above succeeds, TPA verifies data integrity by checking Eq. (11).

$$e(\prod_{k=1}^{K} sig_{\alpha_k}(H(R_k)), g) \stackrel{?}{=} \prod_{k=1}^{K} e(H(R_k'), \beta_k) \quad (10)$$

$$e(TP^{\gamma}, g) \stackrel{?}{=} DP \cdot \prod_{k=1}^{K} e(H(R_k) \cdot \prod_{i\in I} T_{k,i}^{\ v_i}, H_k) \quad (11)$$

If the data blocks are not damaged by the cloud server, the equation (11) will be proved to be true.

The verification Eq. (11) only holds when all the responses are valid. However it will fail when there is even one single invalid response in the batch auditing. The ratio of invalid responses to the valid could be quite small, and yet a standard batch auditor will reject the entire collection. Thus, in order to further sort out these invalid responses in the batch auditing, the paper can utilize a recursive divide-and-conquer approach (binary search), as suggested by Ferrara *et al.* [16]. Specifically, if the batch auditing fails, we can simply divide the collection of responses into two halves, and repeat the auditing on halves via Eq. (11).

4. Evaluation

4.1 Security Analysis

The Computational Diffie-Hellman problem is that, given $g, g^x, g^y \in G$ for unknown $x, y \in G$, to output $g^{xy} \in G$. It is assumed that the (t,ϵ)-CDH assumption holds in G if no t-time algorithm has the non-negligible probability ϵ in solving the CDH problem. Note that the new protocol is designed based on the computational Diffie-Hellman (CDH) problem, which

is widely believed an unsolved problem in polynomial time.

Privacy-preserving property: Before computing and returning Response1, CSPs will authenticate the challenge requests with the certificate issued by user on TPA's public key. Thus, only TPA can send the authentication request. Moreover, after receive the final response from the organizer, TPA need to first validate the leaves of the SMHT which is stored in the organizer. This guarantees only the organizer can compute the final response. Besides, to protect the data privacy, we use Eq. (6) to process the linear combination of specified data blocks, which is the same as the random mask technique.

Transparent verification property: This paper introduces an organizer, who is responsible for interacting with TPA. So, TPA cannot learn the details of data storage. In this paper, it is considered that the organizer is also not trusted. In order to prevent the organizers from getting the location of the data block, user generates a pair of additional data-tag which will be sent to each CSP at setup phase of protocol. At the audit phase, whichever data blocks are in the challenge request from the TPA, every CSP will send a response to the organizer, which can conceal the details of data storage from the organizer. Thus, the proposed batch auditing protocol conceals the storage details of multiple CSPs from both TPA and the organizer.

4.2 Performance Analysis

Computation Cost: The computation cost of simple algebraic operations and modular arithmetic operations is fast enough. Thus, we analyze the performance of the proposed protocol by considering exponent operation and pairing operation. We compare the computation overheads between batch and individual auditing, where K audit tasks need to be performed. The result is shown in Table 1, where $[E]$ denotes the computation cost of an exponent operation in G_1, G_2 or G_T, and $[B]$ is regarded as the computation of a bilinear map $e(\cdot,\cdot)$, which is the most complex operation.

Table 1. Comparison on Computation Overheads between Batch and Individual Auditing

Protocol	*KeyGen*	*TagGen*	*GenProof*(P)	Verification(TPA)
Individual auditing	$K[E]$	$K(n+1)(s+1)[E]$	$KCs[B]+K(t+s)[E]$	$4K[B]+K(t+s)[E]$
Batch auditing	$K[E]$	$K(n+1)(s+1)[E]$	$KCs[B]+K(t+s)[E]$	$(2K+2)[B]+K(t+s)[E]$

As shown in Eq. (9) and Eq. (10), batch auditing not only allows TPA to perform the multiple auditing tasks simultaneously, but also greatly reduces the computation cost on the TPA side. This is because aggregating K verification equations into one helps reduce the number of relatively expensive pairing operations from $4K$ (as required in the individual auditing) to $2K+2$, which saves a considerable amount of auditing time for TPA. Actually, the communication overhead between the cloud servers and TPA is also greatly reduced. Besides, by the cloud servers computing intermediate values of the verification for the auditor, our method can greatly reduce the computing overhead of the auditor.

Communication Cost: In this section, we give the comparison between our protocol and two existing works: the CPDP proposed by Zhu *et al.* [11] and the batch audit protocol proposed by literature [17]. Because the communication cost during the initialization is almost the same in these auditing protocols, we only compare the communication cost between the auditor and the server, which consists of the challenge and the proof. The result is described in Table 2, where we consider a batch auditing with K users and C cloud servers, and t is the number of challenged data blocks from TPA.

Table 2. Communication Cost Comparison of Batch Auditing for K Users and C Cloud Servers

Protocol	*Challenge*	*Proof*
Zhu's CPDP [11]	$O(KCt)$	$O(KCs)$
Batch Auditing [17]	$O(KCt)$	$O(C)$
Our Protocol	$O(KCt)$	$O(C)$

From the table, we can see that these three protocols have the same communication cost during the challenge phase. During the proof phase, like the literature [17], the communication cost of the proof is only linear to C in our protocol. But the literature [17] didn't consider stateless verification and transparent verification. Moreover, in Zhu's CPDP, the communication cost of the proof is not only linear to C and K, but also linear to s. That is because Zhu's CPDP uses the mask technique to protect the data privacy, which requires sending both the masked proof and the encrypted mask to auditor. In our protocol, the server is only required to send the encrypted proof to the auditor, which incurs less communication cost than Zhu's CPDP.

5. Conclusions

In this paper, we explore collaborative integrity auditing issue of remote data stored in the multiple clouds. A batch auditing PDP mechanism for multi-cloud environment is proposed. Our construction enables TPA to perform multiple auditing tasks for multiple data files in multiple clouds. Meanwhile, transparent verification, full stateless verification and secure are also important objectives of the protocol. Utilizing the *s*MHT construction, the proposed protocol achieves full stateless verification and dynamic data operation with integrity assurance. The paper uses BLS-based homomophic authenticator to equip the verification protocol, which enable the TPA to perform batch audits for multiple users based on the technique of bilinear aggregate signature. In addition, by letting the cloud servers computing intermediate values of the verification for the auditor, our method can greatly reduce the computing overhead of the TPA. At last, security analysis shows that the proposed batch auditing protocol is secure, and it also holds much better efficiency than the individual auditing with our performance evaluation.

Acknowledgements

This paper is a revised and expanded version of a paper entitled "Batch Auditing for Multiclient Data in Multicloud Storage" presented at CST 2014, Indonesia, 19-22 June, 2014. This work is supported by the NSFC (61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61373133, 61402234), GYHY201206033, 201301030, 2013DFG12860, BC2013012, Open Fund of Jiangsu Engineering Center of Network Monitoring (KJR1308) and PAPD fund.

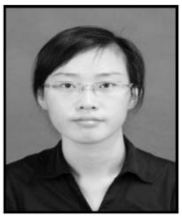
References

[1] G. R. Goodson, J. J. Wylie, G. R. Ganger and M. K. Reiter, "Efficient Byzantine-tolerant erasure-coded storage", in Dependable Systems and Networks, International Conference on, **(2004)**, pp. 135-144.
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner and Z. Peterson, "Provable data possession at untrusted stores," in Proceedings of the 14th ACM conference on Computer and communications security, **(2007)**, pp. 598-609.
[3] G. Ateniese, R. Di Pietro, L. V. Mancini and G. Tsudik, "Scalable and efficient provable data possession", in Proceedings of the 4th international conference on Security and privacy in communication netowrks, **(2008)**, p. 9.
[4] C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia, "Dynamic provable data possession", in Proceedings of the 16th ACM conference on Computer and communications security, **(2009)**, pp. 213-222.
[5] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing", Parallel and Distributed Systems, IEEE Transactions on, vol. 22, **(2011)**,pp.

847-859,.
[6] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing", in Advances in Cryptology—ASIACRYPT 2001, ed: Springer, **(2001)**, pp. 514-532.
[7] C. Wang, S. Chow, Q. Wang, K. Ren and W. Lou, "Privacy-preserving public auditing for secure cloud storage", **(2013)**.
[8] A. Juels and B. S. Kaliski Jr, "PORs: Proofs of retrievability for large files", in Proceedings of the 14th ACM conference on Computer and communications security, **(2007)**, pp. 584-597.
[9] M. A. AlZain, E. Pardede, B. Soh and J. A. Thom, "Cloud computing security: from single to multi-clouds", in System Science (HICSS), Hawaii International Conference on, **(2012)**, pp. 5490-5499.
[10] Y. Zhu, H. Hu, G.-J. Ahn, Y. Han and S. Chen, "Collaborative integrity verification in hybrid clouds", in Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), International Conference on, **(2011)**, pp. 191-200.
[11] Y. Zhu, H. Hu, G. Ahn and M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage", IEEE Transactions on Parallel and Distributed Systems, vol. 23, **(2012)**, pp. 2231-2244.
[12] H. Wang and Y. Zhang, "On the Knowledge Soundness of a Cooperative Provable Data Possession Scheme in Multicloud Storage", IEEE Trans. Parallel Distrib. Syst, **(2013)**,p. 99.
[13] M. Etemad and A. Küpçü, "Transparent, Distributed, and Replicated Dynamic Provable Data Possession", **(2013)**.
[14] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing", Parallel and Distributed Systems, IEEE Transactions on, vol. 24, **(2013)**,pp. 1717-1726.
[15] R. C. Merkle, "Protocols for Public Key Cryptosystems", in IEEE Symposium on Security and privacy, **(1980)**, pp. 122-134.
[16] A. L. Ferrara, M. Green, S. Hohenberger and M. Ø. Pedersen, "Practical short signature batch verification", in Topics in Cryptology–CT-RSA 2009, **(2009)**, pp. 309-324.
[17] K. Yang and X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing",**(2012)**.

Authors

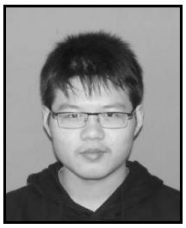
Zhihua Xia received his BS in Hunan City University, China, in 2006, PhD in computer science and technology from Hunan University, China, in 2011. He works as a lecturer in School of Computer & Software, Nanjing University of Information Science & Technology. His research interests include cloud computing security, and digital forensic.



Xinhui Wang received her BS in software engineering from Nanjing University of Information Science & Technology in 2012, China. She is currently pursuing her MS in computer science and technology at the College of Computer and Software, in Nanjing University of Information Science & Technology, China. Her research interest is cloud computing security.

Xingming Sun received his BS in mathematics from Hunan Normal University, China, in 1984, MS in computing science from Dalian University of Science and Technology, China, in 1988, and PhD in computing science from Fudan University, China, in 2001. He is currently a professor in School of Computer & Software, Nanjing University of Information Science & Technology, China. His research interests include network and information security, digital watermarking.

Yafeng Zhu is currently pursuing his BE in computer science and technology at the College of Computer and Software, in Nanjing University of Information Science & Technology, China. Her research interest is cloud computing security.



Peng Ji is currently pursuing his BE in computer science and technology at the College of Computer and Software, in Nanjing University of Information Science & Technology, China. Her research interest is cloud computing security

Jin Wang received the B.S. and M.S. degree from Nanjing University of Posts and Telecommunications, China in 2002 and 2005, respectively. He received Ph.D. degree from Kyung Hee University Korea in 2010. Now, he is a professor in Nanjing University of Information Science and Technology. His research interests mainly include routing algorithm design, performance evaluation and optimization for wireless ad hoc and sensor networks. He is a member of the IEEE and ACM.

