Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.36 – 48

RESEARCH ARTICLE

Multi Owner Data Sharing in Cloud

ANILA A.S
Department of Computer Science
Mahendra Institute of Engineering & Technology,
Mallasamudaram Tamilnadu, India
anilaachu09@gmail.com

MANOJ M J
Asst.Professor, Department of Computer Science
University College of Engineering
Thodupuzha, Kerala, India
manoj_mgu@yahoo.com

Abstract—With the character of low maintenance, cloud computing provides an economical and efficient solution for sharing group resource among cloud users. Unfortunately, sharing data in a multi-owner manner while preserving data and identity privacy from an untrusted cloud is still a challenging issue, due to the frequent change of the membership. In this paper, we propose a secure multiowner data sharing scheme, named Mona, for dynamic groups in the cloud. By leveraging group signature and dynamic broadcast encryption techniques, any cloud user can anonymously share data with others. Meanwhile, the storage overhead and encryption computation cost of our scheme are independent with the number of revoked users. In addition, we analyze the security of our scheme with rigorous proofs, and demonstrate the efficiency of our scheme in experiments.

Index Terms—Cloud computing, data sharing, privacy-preserving, access control, dynamic groups

INTRODUCTION

Cloud computing is recognized as an alternative to traditional information technology [1] due to its intrinsic resource-sharing and low-maintenance characteristics. In cloud computing, the cloud service providers (CSPs), such as Amazon, are able to deliver various services to cloud users with the help of powerful data centres. By migrating the local data management systems into cloud servers, users can enjoy high-quality services and save significant investments on their local infrastructures.

One of the most fundamental services offered by cloud providers is data storage. Let us consider a practical data application. A company allows its staffs in the same group or department to store and share files in the cloud. By utilizing the cloud, the staffs can be completely released from the troublesome local data storage and maintenance. However, it also poses a significant risk to the

confidentiality of those stored files. Specifically, the cloud servers managed by cloud providers are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential, such as business plans. To preserve data privacy, a basic solution is to encrypt data files, and then upload the encrypted data into the cloud [2]. Unfortunately, designing an efficient and secure data sharing scheme for groups in the cloud is not an easy task due to the following challenging issues.

Advances in networking technology and an increase in the need for computing resources have prompted
many organizations to outsource their storage and computing needs. This new economic and computing model is commonly referred to as cloud computing and includes various types of services such as: infrastructure as a service (IaaS), where a customer makes use of a service provider's computing, storage or networking infrastructure; platform as a service (PaaS), where a customer leverages the provider's resources to run custom applications; and software as a service (SaaS), where customers use software that is run on the providers infrastructure. Cloud infrastructures can be roughly categorized as either private or public.

In a private cloud, the infrastructure is managed and owned by the customer and located on-premise (i.e., in the customers region of control). In particular, this means that access to customer data is under its control and is only granted to parties it trusts. In a public cloud the infrastructure is owned and managed by a cloud service provider and is located o_-premise (i.e., in the service provider's region of control). This means that customer data is outside its control and could potentially be granted to untrusted parties. Storage services based on public clouds such as Microsoft's Azure storage service and Amazon's S3 provide customers with scalable and dynamic storage. By moving their data to the cloud customers can avoid the costs of building and maintaining a private storage infrastructure, opting instead to pay a service provider as a function of its needs. For most customers, this provides several benefits including availability (i.e., being able to access data from anywhere) and reliability (i.e., not having to worry about backups) at a relatively low cost.
While the benefits of using a public cloud infrastructure are clear, it introduces significant security and privacy risks. In fact, it seems that the biggest hurdle to the adoption of cloud storage (and cloud computing in general) is concern over the confidentiality and integrity of data. While, so far, consumers have been willing to trade privacy for the convenience of software services (e.g., for web-based email, calendars, pictures etc), this is not the case for enterprises and government.

The concept of provenance has been extensively studied for a long time, and widely used in the archival theory to denote the documented history of some data objects [9]. Given its provenance, a data object can report who created and who modified its contents. Therefore, once a dispute rises in document stored in a cloud, provenance is important for data forensics to provide digital evidences for post investigation. However, provenance is still an unexplored area in cloud computing [5], in which we need to deal with many challenging security issues. For example, in support of data forensics in cloud computing, the provenance information must be secured, i.e., they do not violate the information confidentiality and user privacy in cloud computing. Specifically, as the essential of bread and butter of data forensics in cloud computing, secure provenance should at least satisfy the following basic requirements:
• *Unforgeability*: a genuine provenance record in cloud computing can effectively attest the ownership and process history of data objects stored in a cloud, any adversary cannot forge a valid

provenance records, i.e., modifying an item in a existing record or directly introducing a new forged record without being detected.

• *Conditional privacy preservation*: To ensure information confidentiality and anonymous authentication in cloud computing, a genuine provenance record should also be conditional privacy preserving [12, 14, 15, 13]. That is, only a trusted authority has the ability to reveal the real identity recorded in the provenance, while anyone else cannot.

Secure provenance is vital to the success of data to solve the challenges presented above, we propose Multi owner data sharing in cloud. The main contributions of this paper include:

1. Propose a multi-owner data sharing scheme. It implies that any user in the group can securely share data with others by the untrusted cloud.
2. Provide privacy-preserving access control to users, which guarantees any member in a group to anonymously utilize the cloud resource. Moreover, the real identities of data owners can be revealed by the group manager when disputes occur.

PROPOSED SYSTEM

We consider a cloud computing architecture by combining with an example that a company uses a cloud to enable its staffs in the same group or department to share files. The system model consists of three different entities: the cloud, a group manager (i.e., the company manager), and a large number of group members (i.e., the staffs) as illustrated in Fig. 1.

Cloud is operated by CSPs and provides priced abundant storage services. However, the cloud is not fully trusted by users since the CSPs are very likely to be outside of the cloud users' trusted domain. Similar to [3], [7], we assume that the cloud server is honest but curious. That is, the cloud server will not maliciously delete or modify user data due to the protection of data auditing schemes [17], [18], but will try to learn the content of the stored data and the identities of cloud users.

To fully ensure that the data integrity and save the cloud users computation resources as well as online burden, it is critical importance to enable public auditing service for cloud storage, which provides a much more easier and affordable way for the users to ensure their storage correctness in the cloud. The result from the TPAS would also be beneficial for the cloud service providers to improve their cloud based service platform, even when serve for independent arbitration purpose.

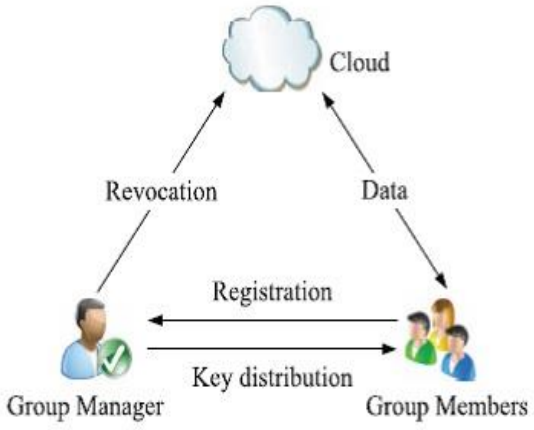

Fig. 1. System model.

PROBLEM STATEMENT

Definition of System Mode:We consider a representative cloud computing architecture, which consists of a trusted system manager (SM), a cloud service provider (SP) and a large number of users
$U = \{U1, U2, \cdots\}$, as shown in Figure 1.•

System Manager (SM):

SM is a trustable and powerful entity, and located at the top of the cloud computing system. The responsibility of SM is in charge of the management of the whole system, for example, registering the cloud service provider and users, and investigating the malicious or faulty users that had operated on some disputed data in cloud computing systems.

Service Provider (SP):

SP is also trustable and has significant resources and monitors live cloud computing systems. When a user accesses to a cloud computing system, for example, reading/writing a file, SP will first authenticate the user and make the system resources available after the user authentication is passed key generation algorithm.

KGen: it is an algorithm run by SM, which takes as input the public parameters *params*, *master key* and an identifier of either a user $Ui \in U$ or the SP, and outputs a corresponding private key *ski*. This algorithm can be either probabilistic or deterministic.

An anonymous authentication algorithm **AnonyAuth**: it is the first interaction run between a user *Ui* and the SP. First, SP sends a random number χ to *Ui*, and *Ui* takes as input a fresh nonce *Yi*, the private key *ski*, and the public parameters *params*, and outputs an anonymous signature σA of $Yi||\chi$. Then, the SP takes as input the purported signature σA, the fresh nonce *Yi*, and the public parameters *params* and test

The auditor is assumed to be honest-but-curious. It performs honestly during the whole auditing procedure but it is curious about the received data. But the sever could be dishonest and may launch the following attacks:

1. Replace Attack. The server may choose another valid and uncorrupted pair of data block and data tag (m_k , t_k) to replace the challenged pair of data block and data tag (m_i, t_i), when it already discarded m_i or t_i.
2. Forge Attack. The server may forge the data tag of data block and deceive the Auditor , if the owner's secret tag keys are reused for the different versions of data.
3. Replay Attack. The server may generate the proof from the previous proof or other information, without retrieving the

Algorithms for Auditing Protocol

Suppose a file F has m data components as $F = (F1, \ldots, Fm)$. Each data component has its physical meanings and can be updated dynamically by the data owners. For public data components, the data owner does not need to encrypted it, but for private data component, the data owner needs to encrypt it with its corresponding key.

inputs. in which FID is the identifier of the data and i represents the block number of m_i. It outputs the set of data tags $T = \{t_i\}_{i\in[1,n]}$.

· **Prove**$(M, T, C) \rightarrow P$. The prove algorithm takes as inputs the data M and the received challenge $C = (\{i, v_i\}_{i\in Q}, R)$. The proof consists of the tag proof

To generate the data proof, it first computes the sector linear combination of all the challenged data blocks MP_j for each $j \in [1, s]$ as

$$MP_j = \sum_{i\in Q} v_i \cdot m_{ij}.$$

Then, it generates the data proof DP as

$$DP = \prod_{j=1} e(u_j, R)^{MP_j}.$$

It outputs the proof $P = (TP, DP)$.

· **Verify**$(C, P, sk_h, pk_t, M_{info}) \rightarrow 0/1$. The verification algorithm takes as inputs the challenge C, the proof P, the secret hash key sk_h, the public tag key pk_t and the abstract information of the data component. It first computes the identifier hash values $h(sk_h, W_i)$ of all the challenged data blocks and computes the challenge hash H_{chal} as

$$H_{chal} = \prod_{i\in Q} h(sk_h, W_i)^{rv_i}.$$

It then verifies the proof from the server by the following verification equation:

$$DP \cdot e(H_{chal}, pk_t) = e(TP, g^r). \quad (2.1)$$

If the above verification Eq. 2.1 holds, it outputs 1. Otherwise, it outputs 0.

Secure Dynamic Auditing

In cloud storage systems, the data owners will dynamically update their data. As an auditing service, the auditing protocol should be designed to support the dynamic data, as well as the static archive data. However, the dynamic operations may make the auditing protocols insecure. Specifically, the server may conduct two following attacks: (1) Replay Attack The server may not update correctly the owner's data on the server and may use the previous version of the data to pass the auditing. (2) Forge Attack When the data owner updates the data to the current version, the server may get enough information from the dynamic operations to forge the data tag. If the server could forge the data tag, it can use any data and its forged data tag to pass the auditing.

Algorithms and Constructions for Dynamic Auditing

The dynamic auditing protocol consists of four phases: Owner Initialization, Confirmation Auditing, Sampling Auditing and Dynamic Auditing.
The first three phases are similar to the privacy-preserving auditing protocol as described in the above section. The only differences are the tag generation algorithm TagGen and the ITable generation during the owner initialization phase. Here, Fig. 2.3 only illustrates the dynamic auditing phase, which contains three steps: Data Update, Index Update and Update Confirmation.

Data Update

There are three types of data update operations that can be used by the owner: Mod- ification, Insertion and Deletion. For each update operation, there is a corresponding algorithm in the dynamic auditing to process the operation and facilitate the future auditing, defined as follows.

· **Modify**$(m^*, sk_t, sk_h) \rightarrow (Msg_{modify}, t^*)$. The modification algorithm takes as inputs the new version of data block m^*, the secret tag key sk_t and the secret hash key sk_h. It generates a new version number V^*, new timestamp T^* and calls the TagGen to generate a new data tag t^* for data block m^*. It outputs the new tag t^* and the update message $Msg_{modify} = (i, B_i, V^*, T^*)$. Then, it sends the new pair of data block and tag (m^*, t^*) to the server and sends the update message Msg_{modify} to the auditor.

· **Insert** $(m^*, sk_t, sk_h) \rightarrow (Msg_{insert}, t^*)$. The insertion algorithm takes as inputs the new data block m^*, the secret tag key sk_t and the secret hash key sk_h. It inserts a new data block m^* before the ith position. It generates an original number B^*, a new version number V^* and a new timestamp T^*. Then, it calls the TagGen to generate a new data tag t^* for the new data block m^*. It outputs the new tag t^* and the update message $Msg_{insert} = (i, B^*, V^*, T^*)$. Then, it inserts the new pair of data block and tag (m^*, t^*) on the server and sends the update message Msg_{insert} to the auditor.

· **Delete**$(m_i) \rightarrow Msg_{delete}$. It takes as input the data block m_i. It outputs the update message $Msg_{delete} = (i, B_i, V_i, T_i)$. It then deletes the pair of data block and its tag (m_i, t_i) from the server and sends the update message Msg_{delete} to the auditor.

Index Update

Receiving the three types of update messages, the auditor calls three corre- sponding algorithms to update the ITable. Each algorithm is designed as follows.

· **IModify**(Msg_{modify}). The algorithm used for index modification

Msg_{modify} as input. It replaces the version number V_i by the new one V^* and modifies T_i by the new timestamp T^*.

· **IInsert**(Msg_{insert}). It is an index insertion algorithm takes as input the update message Msg_{insert}. It inserts a new record (i, B^*, V^*, T^*) in ith position in the ITable. It then moves the original ith record and other records after the i-th position in the previous ITable backward in order, with the index number increased by one.

· **IDelete**(Msg_{delete}). The index deletion algorithm takes as input the update message Msg_{delete}. It deletes the ith record (i, B_i, V_i, T_i) in the ITable and all the records after the ith position in the original ITable moved forward in order, with the index number decreased by one.

Update Confirmation

When complete the auditor updates the ITable, it conducts a confirmation auditing for the updated data and sends the result to the owner. Then, the owner can choose to delete the local version of data according to the update confirmation auditing result.

Batch Auditing for Multi-Owner and Multi-Cloud

Let O_{chal} and S_{chal} denote the involved set of owners and cloud servers involved in the batch auditing respectively. The batch auditing also consists of three steps: Batch Challenge, Batch Proof and Batch Verification.information as input. It selects a set of owners O_{chal} and a set of cloud servers S_{chal}. For each data owner O_k ($k \in O_{chal}$), it chooses a set of data blocks as the challenged subset Q_{kl} from each server S_l ($l \in S_{chal}$).

Then, the auditor sends each C_l to each cloud server S_l ($l \in S_{chal}$) together with the challenge stamp $\{R_k\}_{k \in O_{chal}}$.

EVALUATION

Security Analysis

We first prove that the auditing protocols are provably secure under the security model. Then, we prove that the auditing protocols can also guarantee the data integrity. Finally, we prove that the auditing system is an interactive proof system.

Provably Secure Under the Security Model

Security proofs of the dynamic auditing protocol and batch auditing protocol are similar. Here, we only demonstrate the security proof for the dynamic auditing protocol, as concluded in the following theorems.

Theorem: The dynamic auditing protocol can resist the Replace Attack from the server.

Proof If any of the challenged data blocks m_l or its data tag t_l is corrupted or not up-to-date on the server, the server cannot pass the auditing because the verification equation cannot hold. The server may conduct the replace attack to try to pass the audit. It uses another pair of data block and data tag (m_k, t_k) to replace the chosen

One (m_l,t_l). Then, the data proof DP^* becomes

$$DP^* = \prod_{j=1} e(u_j, R)^{MP_j},$$

where each MP^* can be expressed as

$$MP^* = v_l \cdot m_{kj} + \sum_{i \in Q, i=l} v_i \cdot m_{ij}.$$

The tag proof TP^* can be calculated as

$$TP^* = t^{v_l \cdot k}_{i \in Q, i=l}$$

Privacy-Preserving Guarantee

The data privacy is an important requirement in the design of auditing protocol in cloud storage systems. The proposed auditing protocols are privacy-preserving as stated in the follow theorem.

Theorem: In the proposed auditing protocols, neither the server nor the auditor can obtain any information about the data and the secret tag key during the auditing procedure.

Proof because the data are encrypted by owners, it is obvious that the server cannot decrypt the data without the owners' secret key. The secret hash key and the secret tag key are kept secret to the server and the server cannot deduce them based on the received information during the auditing procedure. Therefore, the data and the secret tag key are confidential against the server in the auditing protocols.

On the auditor side, it can only get the product of all the challenged data tags from the tag proof TP. The data proof in the auditing protocol is in an encrypted way by the exponentiate on the challenge stamps R. It is a discrete logarithm problem to get the linear combinations of the chosen data sectors $\{MP_j\}_{j\in[1,s]}$ from the data proof DP, which is similar to obtain the secret tag key sk_t from g^{sk_t}. Hence, the auditor cannot get any information about the data and the secret tag key from the proof generated by the server in the auditing protocol. For the dynamical index update, the index update messages do not contain any information about the secret tag key and the content of the data, and

thus the auditor cannot obtain any information about the data content from the dynamic operations.

Performance Analysis

Storage auditing is a very resource demanding service in terms of computational resource, communication cost and memory space. In this section, we give the com- munication cost comparison and computation complexity comparison between the TSAS and two existing works: the Audit protocol proposed by Wang et al. [26, 27] and the IPDP proposed by Zhu et al. [31, 32].

Thus, in Wang's auditing scheme, the storage overhead on the server should be 3|M|-bit, three times of the data size.
Both the MONA and Zhu's IPDP apply the data fragment technique to further split each data block into s sectors. Since the data element is the sector in the TSAS and Zhu's IPDP, the size of each sector is corresponding to the security parameter. Then, for each data block that consists of s sectors only one data tag is generated, such that a |M|-bit data component only incurs $\frac{|M|}{s}$-bit storage overhead, which can greatly reduce the storage overhead.

Communication Cost

Because the communication cost during the initialization is almost the same in these three auditing protocols, we only compare the communication cost between the auditor and the server, which consists of the challenge and the proof.
Consider a batch auditing with K owners and C cloud servers. Suppose the number of challenged data block from each owner on different cloud servers is the same, denoted as t, and the data block are split into s sectors in Zhu's IPDP and MONA. We do the comparison under the same probability of detection. That is, in Wang's scheme, the number of data blocks from each owner on each cloud server should be st. The result is described in Table 2.5.
From the table, we can see that the communication cost in Wang's auditing scheme is not only linear to C, K, t, s, but also linear to the total number of data blocks n. As we know, in large scale cloud storage systems, the total number of data blocks could be very large. Therefore, Wang's auditing scheme may incur high communication cost.

MONA Zhu's IPDP have the same total communication cost during the chal- lenge phase. During the proof phase, the communication cost of the proof in MONA is only linear to C, but in Zhu's IPDP, the communication cost of the proof is not only linear to C and K, but also linear to s. That is because Zhu's IPDP uses the mask technique to protect the data privacy, which requires to send both the masked proof and the encrypted mask to the auditor. In MONA, the server is only required to send the encrypted proof to the auditor and thus incurs less communication cost than Zhu's IPDP.

Computation Complexity

The simulation of the computation on the owner, the server and the auditor is con- ducted on a Linux system with an Intel Core 2 Duo CPU at 3.16 GHz and 4.00 GB RAM. The code uses the Pairing-Based Cryptography (PBC) library version 0.5.12 to simulate TSAS and Zhu's IPDP (Under the same detection of probability, Wang's scheme requires much more data blocks than

MONA and Zhu's IPDP, such that the computation time is almost s times more than TSAS and Zhu's IPDP and thus it is not comparable). The elliptic curve used is a MNT d159-curve, where the base field size is 159-bit and the embedding degree is 6. The d159-curve has a 160-bit group order, which means p is a 160-bit length prime. All the simulation results are the mean of 20 trials.

Computation Cost of the Server

We compare the computation cost of the server versus the number of data blocks in Fig. 2.5a and the number of data owners in Fig. 2.5b. MONA moves the computing loads of the auditing from the auditor to the server, such that it can greatly reduce the computation cost of the auditor.

RELATED WORK

Juels et al. proposed a Proofs Of Retrievability (POR) scheme which enables a server (prover) to give a concise proof that a user (verifier) can retrieve a target file [12]. Their POR protocol encrypts the file F and randomly embeds a set of randomly-valued

check blocks called sentinels. The verifier challenges the prover by specifying the positions of a collection of sentinels and asking the prover to return the associated sentinel values. If the prover has modified or deleted a substantial portion of F, then it also has suppressed some sentinels with high probability and thus it cannot respond correctly to the verifier. The security of this protocol is proved by Dodis et al. in [7] without making any simplifying assumptions on the behavior of the adversary. However, this POR protocol is inappropriate for the proposed problem because it only allows a limited number of auditing times which is related to the number of sentinels.

To ensure the data integrity in remote servers, in [22, 23], the owner pre-computes some MACs of the data with different secret keys and sends all the MACs and keys to the auditor. When verifying data integrity, the auditor selects and sends a key k to the server. Then, the server computes the MAC with k and returns it to the auditor for comparison with the one stored on it. However, the number of times a particular data item can be verified is limited by the number of secret keys that fixed beforehand. Besides, the auditor needs to store several MACs for each file. Therefore, Shah's auditing protocols still cannot be applied to the problem

Filho et al. [9] proposed a cryptographic protocol based on RSA-based secure hash function, through which a prover can demonstrate possession of a set of data known to the verifier. But in their protocol the prover needs to exponentiate the entire data file which will cause high computation cost. To overcome the drawback of Filho's protocol, Sebe et al. [20] improved the protocol by first dividing data into blocks and fingerprinting each block and then using a RSA-based hash function on each block. Then, a Diffie-Hellman-based approach is used to verify the data integrity. Their protocol can reduce the computation time of verification by trading off the computation time required at the prover against the storage required at the verifier

Similarly, Yamamoto et al. [28] proposed a fast integrity checking scheme through batch verification of homomorphic hash functions on randomly selected blocks of data. However, in their schemes, the verifier needs to store a copy of the meta-data, such that they cannot be applied to the storage auditing in cloud storage system.

Ateniese et al. proposed a Sampling Provable Data Possession (SPDP) scheme [2], which combines the RSA cryptography with Homomorphic Verifiable Tags (HVT). It divides the data into several data blocks and encrypts each data block. For each auditing query, the auditor only challenge a subset of data blocks. By using such sampling method, the integrity of entire data can be guaranteed, when sufficient number of such sampling auditing queries are conducted. This sampling mechanism is applied in many remote integrity checking scheme, because it could significantly reduce the workloads of the server. Although the SPDP scheme can keep the data privacy, it cannot support the dynamic auditing and the batch auditing for multiple owners.

To support the dynamic auditing, Ateniese et al. developed a dynamic provable data possession protocol [3] based on cryptographic hash function and symmetric key encryption. Their idea is to pre-compute a certain number of metadata during the setup period, so that the number of updates and challenges is limited and fixed before- hand. In their protocol, each update operation requires recreating all the remaining metadata, which is problematic for large files. Moreover, their protocol cannot per- form block insertions anywhere (only append-type insertions are allowed). Erway et al. [8] also extended the PDP model to support dynamic updates on the stored data and proposed two dynamic provable data possession scheme by using a new version of authenticated dictionaries based on rank information. However, their schemes may cause heavy computation burden to the server since they relied on the PDP scheme proposed by the Ateniese.

In [27], the authors proposed a dynamic auditing protocol that can support the dynamic operations of the data on the cloud servers, but this method may leak the data content to the auditor because it requires the server to send the linear combinations of data blocks to the auditor. In [26], the authors extended their dynamic auditing scheme to be privacy-preserving and support the batch auditing for multiple owners. However, due to the large number of data tags, their auditing protocols will incur a heavy storage overhead on the server. In [31], Zhu et al. proposed a cooperative provable data possession scheme that can support the batch auditing for multiple clouds and also extend it to support the dynamic auditing in [32]. However, it is impossible for their scheme to support the batch auditing for multiple owners. That is because parameters for generating the data tags used by each owner are different and thus they cannot combine the data tags from multiple owners to conduct the batch auditing. Another drawback is that their scheme requires an additional trusted organizer to send a commitment to the auditor during the batch auditing for multiple clouds, because their scheme applies the mask technique to ensure the data privacy. However, such additional organizer is not practical in cloud storage systems. Furthermore, both Wang's schemes and Zhu's schemes incur heavy computation cost of the auditor, which makes the auditing system inefficient.

CONCLUSION

In this paper, we design a secure data sharing scheme, a user is able to share data with others in the group without veiling identity privacy to the cloud. Additionally, Mona supports efficient user revocation and new user joining. More specially, efficient user revocation can be achieved through a public revocation list without updating the private keys of the remaining users, and new users can directly decrypt files stored in the cloud before their participation. Moreover, the storage overhead and the encryption computation cost are constant. Extensive analyses show that our proposed scheme satisfies the desired security requirements and guarantees efficiency as well

REFERENCES

[1] A. Adya, W. Bolosky, M. Castro, G. Cermak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *OSDI*, pages 1–14, December 2002.

[2] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, and J. Stern. Scalable secure storage when half the system is faulty. In *ICALP*, 2000.

[3] T. Anderson. Specification of FCrypt: Encryption for AFS remote procedure calls,. http://www.transarc.ibm.com/_ota/fcrypt-paper.txt.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, pages 1–15, 1996.

[5] M. Blaze. A cryptographic file system for UNIX. In *CCS*, 1993.

[6] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46, 1999.

[7] P. D. McDaniel and A. Prakash, "Methods and limitations of security policy reconciliation," in *Proc. of SP'02*, 2002.

[8] T. Yu and M. Winslett, "A unified scheme for resource protection in automated trust negotiation," in *Proc. of SP'03*, 2003.

[9] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," in *Proc. of CCS'05*, 2005.

[10] J. Anderson, "Computer Security Technology Planning Study," Air Force Electronic Systems Division, Report ESD-TR-73-51, 1972, http://seclab.cs.ucdavis.edu/projects/history/

[11] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu,"Scalable secure file sharing on untrusted storage," in *Proc. of FAST'03*,
2003.

[12] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing remote untrusted storage," in *Proc. of NDSS'03*, 2003.

[13] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proc. of NDSS'05*, 2005.

[14] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc. of VLDB'07*, 2007.

[15] Lu, R., Lin, X., Zhu, H., and Shen, X. Spark: a new vanet-based smart parking scheme for large parking lots. In *The 28th Conference on Computer Communications (INFOCOM 2009)* (Rio de Janeiro, Brazil, April 2009).

[16] Lynch, C. A. When documents deceive: Trust and provenance as new factors for information retrieval in a tangled web. *Journal of the American Society for Information Sciene and Technology 52*, 1 (2001), 12–17.

[17] Mei, L., Chan, W., and Tse, T. A tale of clouds: Paradigm comparisons and some thoughts on research issues. In *Proceedings of Asia-Paci_c Services Computing Conference, APSCC'08* (Yilan, Dec. 2008), pp. 464–469.

[18] Shoup, V. Oaep reconsidered. *Journal of Cryptology 15*, 4 (2002), 223–249.

[19] Sterling, T., and Stark, D. A high-performance computing forecast: Partly cloudy. *Computing in Science & Engineering 11*, 4 (July-Aug. 2009), 42–49.

[20] Voas, J., and Zhang, J. Cloud computing: New wine or just a new bottle? *IT Professional 11*, 2 (March-April 2009), 15–17.

[21] Waters, B. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008.

