A Survey on Efficient Public Integrity Checking for Cloud Data Sharing with Multi-User Modification

Prof. Madhu. B
Assistant Professor
Department of CSE
Dr. AIT-56

Prof. Veena Potdar
Associate Professor
Department of CSE
Dr. AIT-56

Koosappa M
Shivaputra Wadeyar
Siddaram Biradhar
Vinod V B
8th Semester CSE, DR.AIT

Abstract:- **In past years a body of data integrity checking techniques has been proposed for securing cloud data services. Most of these works assume that only the data owner can modify cloud-stored data. Recently a few attempts started considering more realistic scenarios by allowing multiple cloud users to modify data with integrity assurance. However, these attempts are still far from practical due to the tremendous computational cost on cloud users. Moreover, collusion between misbehaving cloud servers and revoked users is not considered. This paper proposes a novel data integrity checking scheme characterized by multi-user modification, collusion resistance and a constant computational cost of integrity checking for cloud users. This is a result of our novel design of polynomial-based authentication tags and proxy tag update techniques. Our scheme also supports public checking and efficient user revocation and is provably secure. Numerical analysis and extensive experimental results show the efficiency and scalability of our proposed scheme [1].**

INTRODUCTION

The continuous development of cloud techniques has boosted a number of cloud-based applications. In particular, more and more applications are using cloud as collaboration platforms, in which data are not only persisted in cloud for storage but also subject to frequent modifications from multiple users. Real-world examples are: Cloud-based synchronization platforms such as Drop box for Business [4] and Sugar sync [5], which enable multiple team members to work in sync, accessing and modifying the same file on cloud servers anywhere, anytime. For correct execution of this kind of cloud-based collaborative applications, one problem is to assure data integrity, i.e., each data modification operation is indeed performed by an authorized group member and the data remains intact thereafter. This problem is important given the fact that cloud platforms, even well-known cloud platforms, may experience hardware / software failures, human errors and malicious attacks [6], [7].

RELATED WORK

In order to ensure the integrity of data stored on remote / cloud servers, a series of solutions have been proposed based on various techniques. However, most of these existing solutions only consider the case of single writer – only the data owner who holds secret keys can modify the data. If these solutions are trivially extended to support multiple writers with data integrity assurance, the data owner has to stay online, collecting modified data from other users and regenerating authentication tags for them. Obviously, this kind of trivial extension will introduce a tremendous workload to the data owner, especially in scenarios with a large number of writers (users) and / or a high frequency of data modification operations. To allow multiple users to modify data, a public integrity auditing scheme using ring signature-based homomorphic authenticators has been proposed. User revocation is not considered and the auditing cost is linear to the group size and data size. In order to further enhance the previous works, which still suffers from a non-trivial computational cost, is linear to the group size and the number of checking tasks is thus limited in scalability. In addition, user revocation is based on the assumption that no collusion occurs between cloud servers and revoked user. As a matter of fact, collusion between invalid users and cloud servers will lead to the disclosure of secrets of all other valid users. To our best knowledge, there still lacks an efficient and publicly verifiable integrity checking scheme for cloud data sharing that supports multiple writers with secure user revocation[1].

CHALLENGES

To solve this open problem, the following major challenges exist:

1) Efficiency and scalability. The main difficulty is attributed to two facets - on one hand, each data block (assuming data integrity is verified per block) shall have its own authentication tag; on the other hand, each writer (user) needs to generate a separate authentication tag, which is necessary for integrity checking, for each data block that he / she has modified. Consequently, the communication / computational complexity for integrity checking can grow linearly with the number of writers and / or the number of data blocks being checked.

2) Efficient and secure user revocation. User revocation is a challenging issue in most security systems and usually involves disabling user secret keys. For efficient user revocation, one technique is to delegate the task to the cloud by disclosing partial secret to cloud servers. This method, however, leaves it a challenging issue to detect misbehaving cloud servers who collude with unauthorized users.

3) Public checking. In practical systems, data integrity checking can be performed not
only by data owners or writers but also by a third-party auditor or any general user who needs to access the data [1].

In cloud environments, if a data owner wants to share data with users, he will encrypt data and then upload it to cloud storage service. Through the encryption step, the cloud cannot know the information of the encrypted data. Besides, to avoid the unauthorized user accessing the encrypted data in the cloud, a data owner uses the encryption scheme for access control of encrypted data. In existing schemes, many encryption schemes can achieve and provide security, assure data confidentiality and prevent collusion attack scheme. One of the encryption schemes is attribute-based encryption scheme. The first concept of attribute-based encryption was proposed in 2005. And then many attribute-based encryption schemes were proposed. According to the access policy, two types of these schemes can be classified, the key-policy and cipher text-policy attribute-based encryption schemes. The key-policy attribute-based scheme is that the access policy is attached to the user's private key, and a set of descriptive attributes is in the encrypted data. If a set of attributes satisfies the access policy, the user will recover the message. If not, he cannot obtain it. And the cipher text policy attribute-based scheme is that the access policy is associated to the encrypted data and a set of descriptive attributes is in the user's private key. If a set attribute satisfies the access policy, the user can decipher the encrypted data. In this section, five attribute-based encryption schemes have been introduced. According to the type of access policy, there are monotonic access structures and non-monotonic access structures. Non-monotonic access structures can use the negative word to describe every attribute, but the monotonic access structures cannot [2].

Solutions for the above problem:

1. Attribute based Encryption Scheme Sahai and Waters proposed an attribute based encryption scheme in 2005. There are authority, data owner (also be called sender) and data user (also be called receiver) in this scheme, and authority's role is to generate keys for data owners and users to encrypt or decrypt data. In this scheme, the authority generates keys according to attributes; and these attributes of public key and master key, which are generated by the authority, should be predefined (means that it will list attributes which will be used in the future). The authority will redefine attributes and generate a public key and master key again. And data owner's role in this scheme is to encrypt data with a public key and a set of descriptive attributes. A data user's role is to decrypt encrypted data with his private key sent from the authority, and then he can obtain the needed data. For decrypting data, attributes in data user's private key will check by matching with the attributes in encrypted data. If the number of "matching" is at least a threshold value d, the data user's private key will be permitted to decrypt the encrypted data. For example, for a set of descriptive attributes in the encrypted data, {MIS, Teacher, Student}, the threshold value is 2. If a data user wants to decrypt the encrypted data, his number of attributes in private key will need two or the more than two attributes in the encrypted data, so that a data user has a private key with attributes,{MIS, Student} to decrypt and obtain the data[2].

2. Key-Policy Attribute-based Encryption Scheme In 2006, Goyal proposed a key-policy attribute-based (KP-ABE) scheme. This scheme uses a set of attributes to describe the encrypted data and builds an access policy in user's private key. If attributes of the encrypted data can satisfy the access structure in user's private key D, and user can obtain the message through decrypt algorithm. In addition, the Key Gen () algorithm is different from the attribute-based encryption which is introduced at subsection one in this section. The user's private key is according to the access structure to generate. In this algorithm, it adopts secret sharing and chooses a polynomial q x such that q x (0) = q parent(x) (index(x)), (Note that parent(x) is x's parent node, and index(x) is the number associated with node x that is given by x's parent node.) in a top-down manner which is to start from the root node r for each node x in the access structure. So q r (0) is equal to the master key y, and the master key y is distributed among the user's private key component D i which is corresponding to the leaf node (Note that the leaf node represents attribute)[2].

3. Cipher text-Policy Attribute-based Encryption Scheme In 2007, Bethencourt et al. proposed a ciphertext policy attribute-based scheme, and the access policy in the encrypted data (ciphertext). The access control method of this scheme is similar to the key policy attribute-based encryption. In key policy attribute-based encryption, the access policy is in user's private key, but the access policy is switched to the encrypted data in ciphertext policy attribute-based encryption. And a set of descriptive attributes are associated with the user's private key, and the

access policy is built in the encrypted data. The access structure of the encrypted data is corresponding to the user's private key with a set of descriptive attributes. If a set of attributes in user's private key satisfies the access structure of the encrypted data, the data user can decrypt the encrypted data; if it cannot, the data user cannot obtain the message. For example, the access structure in the encrypted data is {MIS V(Teacher W Student)}. If a set of attributes in user's private key is {MIS V Teacher}, the user can recover the data [2].

4. Attribute-based Encryption Scheme with Non-Monotonic Access Structures In 2007, Ostrovsky et al. proposed an attribute-based encryption with non-monotonic access structure. The access formula of access structure in user's private key can represent any type through attributes such as negative ones. It is different from the previous attribute-based encryption scheme. The previous schemes are like KP-ABE scheme, and the access structure in user's private key has monotonic access formula. No negative attributes exist in it. Apart from this, the access structure of this scheme is the same as the access structure of KP-ABE scheme. There is a Boolean formula such as AND, OR, and threshold gates in these access structures, but there is a Boolean formula, NOT in access structure of this scheme. However, other schemes do not include it. There is an example for this scheme. If a teacher in department of information management wants to share the data with students, he will set a set of attributes in the encrypted data. And there is an access structure, {MIS V Student} in students' private key. But the teacher doesn't want graduates to access this data, he adds NOT graduate to the access structure. So the access structure is {MIS V Student V NOT graduate}. It can let data not be accessed by graduates [2].

Our contribution in this work addresses all the above challenges and proposes an efficient public integrity checking scheme for cloud data sharing that supports multiple writers. Our proposed scheme is featured by public integrity checking and constant computational cost on the user side, thanks to our novel design on polynomial-based authentication tags which allows aggregation of tags of different data blocks. For system scalability, we further empower the cloud with the ability to aggregate authentication tags from multiple writers into one when sending the integrity proof information to the verifier. As a result, just a constant size of integrity proof information needs to be transmitted to the verifier no matter how many data blocks are being checked and how many writers are associated with the data blocks. Moreover, our novel design allows secure delegation of user revocation operations to the cloud even if cloud servers collude with malicious users. Last but not least, our proposed scheme allows aggregation of integrity checking operations for multiple tasks (files) through our batch integrity checking technique. We prove the security of our design based on the Computational Diffie-Hellman (CDH) problem, the Bilinear Diffie-Hellman (BDH) problem and the t-Strong Diffie-Hellman (t-SDH) problem. Thorough analysis and extensive experimental results on Amazon EC2 demonstrate the scalability and efficiency of our design. Our contributions are summarized as follows,

. Our proposed scheme is among the first that achieves public integrity checking for cloud data sharing supporting multiple writers. Different from existing solutions, our scheme is efficient, highly scalable, and collusion resistant.

• Our novel design of authentication tags can serve as an independent tool that can easily find application in other related scenarios such as verifiable keyword search.

• We have fully implemented our scheme and evaluated its performance through both numerical analysis and experimental results [2]. The security of our design is proved.

MODELS AND ASSUMPTIONS.

A. System Models

We consider a cloud system composed of three major entities as shown in Fig.1: the cloud server, group users and the third-party auditor (TPA). The cloud server is the party that provides data storage services to group users. Group users consist of a number of general users and a master user, who is the owner of the shared data and manages the membership of other group users. All group users can access and modify data. The TPA refers to any party that checks the integrity of data being stored on the cloud. As our proposed scheme allows public integrity checking, the TPA can actually be any cloud user as long as he/she has access to the public keys. In our design, data can be uploaded/created by either the master user or other group users. We assume data are stored in form of files which are further divided into a number of blocks. For integrity checking, each data block is attached with an authentication tag that is originally generated by the master user. When a user adds or modifies a block, he/she (the user) updates the corresponding authentication tag with his/her own secret key without contacting the master user.

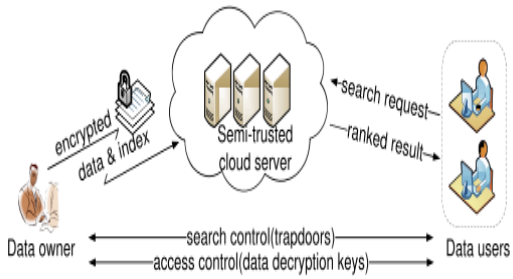

Fig. 1: Architecture of the search over encrypted cloud data

B. Security Models

In this work we consider the following factors that may impact data integrity:

1) Hardware/software failures and operational errors of system administrator; 2) External adversaries that corrupt data stored on the cloud;

3) Revoked users who no longer have data access privilege but try to illegally modify data. We assume the cloud server to be curious but honest –the cloud server will follow the protocol but try to disclose the private

information (e.g. users secret keys) as much as possible. Different from previous work, we allow collusion between the cloud server and revoked group users. Since valid users are always allowed to modify data, we assume that they never collude with the cloud servers for corrupting data. However, the cloud and the valid users may collude with revoked users for helping them retain the modification privilege or impersonate other valid users (for generating tags).

Therefore, our security goals are defined as follows: 1) If the data are corrupted, the cloud servers are not able to produce valid integrity proof information; 2) Collusion among the cloud and any group users other than the master user will not give users any chance to forge authentication on behalf of other users. In particular, revoked users shall not be able to impersonate valid users and generate legitimate tags by collusion attacks [3].

SECURITY ANALYSIS

A. Assumption Our security proof is based on the following hard problems:

Definition IV.1. Computational Diffie-Hellman (CDH) Problem Let a, b be two random numbers. Given (g, g a, g b), it is computationally intractable to compute the value of g ab, where G is a cyclic group of order q and g is a generator of G.

Definition IV.2. Bilinear Diffie-Hellman (BDH) Problem Let G, G 1 be two groups of prime order q, e : G×G → G 1 be an admissible bilinear map and g is the generator of G. Given {g, g a , g b , g c } for some a, b, c ∈ Z ∗q , the probability Pr[Adv(g, g a , g b , g c) = e(g, g) abc] is negligible for any probabilistic polynomial time adversary (Adv).

Definition IV.3. t- Strong Diffie-Hellman (t-SDH) Problem Let α R← Z ∗q. Given input as a (t + 1) – tuple (g, g α , · · · , g αt) ∈ G t+1 , where g is the generator of a cyclic group G of order q. For any probabilistic polynomial time adversary (Adv), the probability Pr[Adv(g, g α , · · , g αt) =(c, g1α+c)] is negligible for any value of a ∈ Z ∗q−α.

PROPOSED SYSTEM

We consider a cloud system composed of three major entities as shown in Fig.1: the cloud server, group users and the third-party auditor (TPA). The cloud server is the party that provides data storage services to group users. Group users consist of a number of general users and a master user, who is the owner of the shared data and manages the membership of other group users. All group users can access and modify data. The TPA refers to any party that checks the integrity of data being stored on the cloud. As our proposed scheme allows public integrity checking, the TPA can actually be any cloud user as long as he/she has access to the public keys. In our design, data can be uploaded/created by either the master user or other group users. We assume data are stored in form of files which are further divided into a number of blocks. For integrity checking, each data block is attached with an authentication tag that is originally generated by the master user. When a user adds or modifies a block, he/she (the user) updates the corresponding authentication tag with his/her own secret key without contacting the master user.

CONCLUSION

In this paper, we propose a novel data integrity checking scheme that supports multiple writers to share data. Our proposed scheme is featured by salient properties of public integrity checking and constant computational cost on the user side. We achieve this through our innovative design on polynomial-based authentication tags which allows aggregation of tags of different data blocks. For system scalability, we further empower the cloud with the ability to aggregate authentication tags from multiple writers into one when sending the integrity proof information to the verifier (who may be general cloud users). As a result, just a constant size of integrity proof information needs to be transmitted to the verifier no matter how many data blocks are being checked and how many writers are associated with the data blocks. Moreover, our novel design allows secure delegation of user revocation operations to the cloud even if cloud servers collude with malicious users. Last but not least, our proposed scheme allows aggregation of integrity checking operations for multiple tasks (files) through our batch integrity checking technique. As compared to existing schemes, our solution has obvious advantages in terms of efficiency, scalability and security. Extensive numerical analysis and real-world experiments validate the performance of our scheme. Rigorous security analysis shows that our solution is provably secure.

REFERENCES

1. Jiawei Yuan, Shucheng Yu., "Efficient Public Integrity Checking for Cloud Data Sharing with Multi-User Modification"
2. Cheng-Chi Lee, Pei-Shan Chung, and Min-Shiang Hwang, "A Survey on Attribute-based Encryption Schemes of Access Control in Cloud Environments" http://ijns.femto.com.tw/contents/ijns-v15-n4/ijns-2013-v15-n4-p231-240.pdf
3. Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou., "Privacy-Preserving Multi-keyword Ranked Search over Encrypted Cloud Data"
4. "Dropbox for business," https://www.dropbox.com/business.
5. "Amazon ec2 and amazon rds service disruption," http://aws.amazon.com/message/65648/.
6. 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, ser. CCS '07. Alexandria, Virginia, USA: ACM, 2007
7. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in Proceedings of the 14th ACM Conference on Computer and Communications Security, ser. CCS '07. Alexandria, Virginia, USA: ACM, 2007