Secured Key Escort Services in Cloud Computing

Ramavatar Kumawat[†], Sandeep Singh[†], Manohar Prajapati[††]

[†]Suresh Gyan Vihar University, Jaipur
[††]Poornima college of Engineering, Jaipur

Abstract:
Cloud computing is an emerging computing paradigm in which resources of the computing infrastructure are provided as services over the Internet and this brings forth many new challenges for data security and access control when users outsource sensitive data for sharing on cloud servers, which are not within the same trusted domain as data owners. To keep sensitive user data confidential against untrusted servers, existing solutions usually apply cryptographic methods by disclosing data decryption keys only to authorized users. However, in doing so, these solutions inevitably introduce a heavy computation overhead on the data owner for key distribution and data management when finegrained data access control is desired, and thus do not scale well. The problem of simultaneously achieving fine-grainedness, scalability, and data confidentiality of access control actually still remains unresolved. This project addresses this issue by using access policies based on data attributes that allow the data owner to delegate most of the computation tasks involved in finegrained data access control to untrusted cloud servers without disclosing the underlying data contents. The proposed scheme also has salient properties of user access privilege confidentiality and user secret key accountability. The proposed scheme can enable the data owner to delegate most of computation overhead to powerful cloud servers. Confidentiality of user access privilege and user secret key accountability can be achieved. Formal security proofs show that our proposed scheme is secure under standard cryptographic models. This paper aims at fine-grained data access control in cloud computing and also extensive analysis shows that our proposed schemes is highly efficient and provably secure under existing security models.

1. INTRODUCTION

Cloud computing is a promising computing paradigm which recently has drawn extensive attention from both academia and industry. By combining a set of existing and new techniques from research areas such as Service-Oriented Architectures (SOA) and virtualization, cloud computing is regarded as such a computing paradigm in which resources in the computing infrastructure are provided as services over the Internet. Along with this new paradigm, various business models are developed, which can be described by terminology of "X as a service (XaaS)" where X could be software, hardware, data storage, and etc. Successful examples are Amazon's EC2 and S3, Google App Engine, and Microsoft Azure which provide users with scalable resources in the pay-as-youuse fashion at relatively low prices. For example, Amazon's S3data storage service just charges $0.12 to $0.15 per gigabyte month. As compared to building their own infrastructures, users are able to save their investments significantly by migrating businesses into the cloud. With the increasing development of cloud computing technologies, it is not hard to imagine that in the near future more and more businesses will be moved into the cloud. As promising as it is, cloud computing is also facing many hallenges that, if not well resolved, may impede its ast growth. Data security, as it exists in many other pplications, is among these challenges that would raise reat concerns from users when they store sensitive information on cloud servers. These concerns originate from the fact that cloud servers are usually operated by commercial providers which are very likely to be outside of the trusted domain of the users. Data confidential against cloud servers is hence frequently desired when users outsource data for storage in the cloud. For example, in healthcare application scenarios use and disclosure of protected health information (PHI) should meet the requirements of Health Insurance Portability and Accountability Act (HIPAA), and keeping user data confidential against the storage servers is not just an option, but a requirement.

As a significant research area for system protection, data access control has been evolving in the past thirty years and various techniques have been developed to effectively implement fine-grained access control, which allows flexibility in specifying in the same trusted domain, where the servers are fully entrusted as an omniscient reference monitor responsible for defining and enforcing access control policies. These existing works, as we will discuss in section V-C, resolve this issue either by introducing a per file access control list (ACL) for fine-grained access control, or by categorizing files into several filegroups for efficiency. As the system scales, however, the complexity of the ACL-based scheme would be proportional to the number of users in the system.

In this paper, we address this open issue and propose a secure and scalable fine-grained data access control scheme for cloud computing. Our proposed scheme is partially based on our observation that, in practical application scenarios each data file can be associated with a set of attributes which are meaningful in the context of interest. The access structure of each user can thus be defined as a unique logical expression over these attributes to reflect the

Manuscript received August 5, 2013
Manuscript revised August 20, 2013

scope of data files that the user is allowed to access. We achieve our design goals by exploiting a novel cryptographic primitive, namely key policy attribute-based encryption (KP-ABE), and uniquely combine it with the technique of proxy reencryption (PRE) and lazy re-encryption. Main contributions of this paper can be summarized as follows.

1) To the best of our knowledge, this paper is the first that simultaneously achieves fine-grainedness, scalability and data confidentiality for data access control in cloud computing;

2) Our proposed scheme enables the data owner to delegate most of computation intensive tasks to cloud servers without disclosing data contents or user access privilege information;

3) The proposed scheme is provably secure under the standard security model. In addition, our proposed scheme is able to support user accountability with minor extension.

2. MODELS AND ASSUMPTIONS

2.1 System Models

Similar to, we assume that the system is composed of the following parties: the Data Owner, many Data Consumers, many Cloud Servers, and a Third Party Auditor if necessary. To access data files shared by the data owner, Data Consumers, or users for brevity, download data files of their interest from Cloud Servers and then decrypt. Neither the data owner nor users will be always online. They come online just on the necessity basis. For simplicity, we assume that the only access privilege for users is data file reading. Extending our proposed scheme to support data file writing is trivial by asking the data writer to sign the new data file on each update as does. From now on, we will also call data files by files for brevity. Cloud Servers are always online and operated by the Cloud Service Provider (CSP). They re assumed to have abundant storage capacity and computation power. The Third Party Auditor is also an online party which is used for auditing every file access event. In addition, we also assume that the data owner can not only store data files but also run his own code on Cloud Servers to manage his data files.

2.2. Security Models

In this work, we just consider Honest but Curious Cloud Servers as does. That is to say, Cloud Servers will follow our proposed protocol in general, but try to find out as much secret information as possible based on their inputs. More specifically, we assume Cloud Servers are more interested in file contents and user access privilege information than other secret information. Cloud Servers might collude with a small number of malicious users for the purpose of harvesting file contents when it is highly beneficial. Communication channel between the data owner/users and Cloud Servers are assumed to be secured under existing security protocols such as SSL. Users would try to access files either within or outside the scope of their access privileges. To achieve this goal, unauthorized users may work independently or cooperatively. In addition, each party is preloaded with a public/private key pair and the public key can be easily obtained by other parties when necessary.

2.3. Design Goals

Our main design goal is to help the data owner achieve fine-grained access control on files stored by Cloud Servers. Specifically, we want to enable the data owner to enforce a unique access structure on each user, which precisely designates the set of files that the user is allowed to access. We also want to prevent Cloud Servers from being able to learn both the data file contents and user access privilege information. In addition, the proposed scheme should be able to achieve security goals like user accountability and support basic operations such as user grant/revocation as a general one-to-many communication system would require. All these design goals should be achieved efficiently in the sense that the system is scalable.

3. TECHNIQUE PRELIMINARIES

3.1 Key Policy Attribute-Based Encryption (KP-ABE)

KP-ABE is a public key cryptography primitive for one-to-many communications. In KP-ABE, data are associated with attributes for each of which a public key component is defined. The encryptor associates the set of attributes to the message by encrypting it with the corresponding public key components. Each user is assigned an access structure which is usually defined as an access tree over data attributes, i.e., interior nodes of the access tree are threshold gates and leaf nodes are associated with attributes. User secret key is defined to reflect the access structure so that the user is able to decrypt a ciphertext if and only if the data attributes satisfy his access structure. A KP-ABE scheme is composed of four algorithms which can be defined as follows:

Setup This algorithm takes as input a security parameter κ and the attribute universe $U = 1, 2, \ldots, N$ of cardinality N. It defines a bilinear group G_1 of prime order p with a generator g, a bilinear map $e: 1$
$G_1 G_2$ which has the properties of *bilinearity*, *computability*, and *non-degeneracy*. It returns the public key *PK* as well as a system master key *MK* as follows

$PK = (Y, T_1, T_2, \ldots, T_N)$

MK = (y, t1, t2, . . . , tN) where Ti G1 and ti Zp

are for attribute i, 1 i N, and Y G2 is another public key component. We have Ti = gti and Y = e(g,g)y, y Zp. While PK is publicly known to all the parties in the system, MK is kept as a secret by the authority party.

Encryption This algorithm takes a message M, the public key PK, and a set of attributes I as input. It outputs the ciphertext E with the following format: E = (I, ˜ E, Ei i I) where ˜E = MYs, Ei = Ts i , and s is randomly chosen from Zp.

Key Generation This algorithm takes as input an access tree T, the master key MK, and the public key PK. It outputs a user secret key SK as follows. First, it defines a random polynomial pi(x) for each node I of T in the top-down manner starting from the root node r. For each non-root node j, pj(0) = pparent(j)(idx(j)) where parent(j) represents j's parent and idx(j) is j's unique index given by its parent. For the root node r, pr(0) = y. Then it outputs SK as follows. SK = ski i L where L denotes the set of attributes attached to the leaf nodes of T and ski = g pi(0) ti.

Decryption This algorithm takes as input the ciphertext E encrypted under the attribute set I, the user's secret key SK for access tree T, and the public key PK. It first computes e(Ei, ski) = e(g, g)pi(0)s for leaf nodes. Then, it aggregates these pairing results in the bottom-up manner using the polynomial interpolation technique. Finally, it may recover the blind factor Y s = e(g, g) ys and output the message M if and only if I satisfies T. Please refer to for more details on KP-ABE algorithms are an enhanced KP-ABE scheme which supports user secret key accountability.

3.2 Proxy Re-Encryption (PRE)

Proxy Re-Encryption (PRE) is a cryptographic primitive in which a semi-trusted proxy is able to convert a ciphertext encrypted under Alice's public key into another ciphertext that can be opened by Bob's private key without seeing the underlying plaintext. More formally, a PRE scheme allows the proxy, given the proxy re-encryption key rka b, to translate ciphertexts under public key pka into ciphertexts under public key pkb and vise versa. Please refer to for more details on proxy re-encryption schemes.

4. OUR PROPOSED SCHEME

4.1 Main Idea

In order to achieve secure, scalable and fine-grained access control on outsourced data in the cloud, we utilize and uniquely combine the following three advanced cryptographic techniques: KP-ABE, PRE and lazy re-encryption. More specifically, we associate each data file with a set of attributes, and assign each user an expressive access structure which is defined over these attributes. To enforce this kind of access control, we utilize KP-ABE to escort data encryption keys of data files. Data confidentiality is also achieved since Cloud Servers are not able to learn the plaintext of any data file in our construction. For further reducing the computation overhead on Cloud Servers and thus saving the data owner's investment, we take advantage of the lazy re-encryption technique and allow Cloud Servers to "aggregate" computation tasks of multiple system operations. Scalability is thus achieved. In addition, our construction also protects user access privilege information against Cloud Servers. Accoutability of user secret key can also be achieved by using an enhanced scheme of KP-ABE.

4.2 Definition and Notation

For each data file the owner assigns a set of meaningful attributes which are necessary for access control. Different data files can have a subset of attributes in common. Each attribute is associated with a version number for the purpose of attribute update as we will discuss later. Cloud Servers keep an attribute history list AHL which records the version evolution history of each attribute and PRE keys used. In addition to these meaningful attributes, we also define one dummy attribute, denoted by symbol AttD for the purpose of key management. AttD is required to be included in every data file's attribute set and will never be updated. Leaf nodes of the access tree are associated with data file attributes. For the purpose of key management, we require the root node to be an AND gate (i.e., n-of-n threshold gate) with one child being the leaf node which is associated with the dummy attribute, and the other child node being any threshold gate. The dummy attribute will not be attached to any other node in the access tree. Fig.1 illustrates our definitions by an example. In addition, Cloud Servers also keep a user list UL which records IDs of all the valid users in the system.

4.3 Scheme Description

For clarity we will present our proposed scheme in two levels: System Level and Algorithm Level. At system level, we describe the implementation of high level operations, i.e., System Setup, New File Creation, New User Grant, and User Revocation, File Access, File Deletion, and the interaction between involved parties. At algorithm level, we focus on the implementation of low level algorithms that are invoked by system level operations.

1) System Level Operations: System level operations in our proposed scheme are designed as follows. System Setup In this operation, the data owner chooses a security parameter κ and calls the algorithm level interface ASetup(κ), which outputs the system public parameter PK and the system master key MK. The data owner then signs each component of PK and sends PK along with these signatures to Cloud Servers.New File Creation Before uploading a file to Cloud Servers, the data owner processes the data file as follows.

select a unique ID for this data file; randomly select a symmetric data encryption key DEK R, where is the key space, and encrypt the data file using DEK; define a set of attribute I for the data file and encrypt DEK with I using KP-ABE,i.e., (˜ , Ei i I)AEncrypt(I,DEK,PK).
Finally, each data file is stored on the cloud in the format.New User Grant When a new user wants to join the system, the data owner assigns an access structure and the corresponding secret key to this user as follows.
assign the new user a unique identity w and an access structure P; generate a secret key SK for w, i.e., SK AKeyGen(P,MK);
encrypt the tuple (P,SK,PK, δO,(P,SK,PK)) with user w's public key, denoting the ciphertext by C; send the tuple (T,C, δO,(T,C)) to Cloud Servers, where T denotes the tuple (w, j,skj j LP AttD). On receiving the tuple (T,C, δO,(T,C)), Cloud Servers processes as follows.
verify δO,(T,C) and proceed if correct; store T in the system user list UL; forward C to the user. On receiving C, the user first decrypts it with his private key. Then he verifies the signature δO,(P,SK,PK). If correct, he accepts (P, SK,PK) as his access structure, secret key, and the system public key.As described above, Cloud Servers store all the secret key components of SK except for the one corresponding to the dummy attribute AttD. Such a design allows Cloud Servers to update these secret key components during user revocation as we will describe soon. As there still exists one undisclosed secret key component (the one for AttD), Cloud Servers can not use these known ones to correctly decrypt ciphertexts. Actually, these disclosed secret key components, if given to any unauthorized user, do not give him any extra advantage in decryption as we will show in our security analysis.
User Revocation We start with the intuition of the user revocation operation as follows. Whenever there is a user to be revoked, the data owner first determines a minimal set of attributes without which the leaving user's access structure will never be satisfied. Next, he updates these attributes by redefining their corresponding system master key components in MK. Public key components of all these updated attributes in PK are redefined accordingly.
In the first stage, the data owner determines the minimal set of attributes, redefines MK and PK for involved attributes, and generates the corresponding PRE keys. He then sends the user's ID, the minimal attribute set, the PRE keys, the updated public key components, along with his signatures on these components to Cloud Servers, and can go off-line again. This property allows Cloud Servers to update user secret keys and data files in the "lazy" way as follows. Once a user revocation event occurs, loud Servers just record information submitted by the Data owner as is previously discussed.
File Access This is also the second stage of user revocation. In this operation, Cloud Servers respond user request on data file access, and update user secret keys and re-encrypt requested data files if necessary. If correct, the user further verifies if each secret key component returned by Cloud Servers is correctly computed. He verifies this by computing a bilinear pairing between sk j and T j and comparing the result with that between the old skj and Tj that he possesses. If verification succeeds, he replaces each skj of his secret key with sk j and Update Tj with T j . Finally, he decrypts data files by first calling ADecrypt(P, SK,E) to decrypt DEK's and then decrypting data files using DEK's.
File Deletion This operation can only be performed at the request of the data owner. To delete a file, the data owner sends the file's unique ID along with his signature on this ID to Cloud Servers. If verification of the owner's signature returns true, Cloud Servers delete the data file.

4.4 Summary

In our proposed scheme, we exploit the technique of hybrid encryption to protect data files, i.e., we encrypt data files using symmetric DEKs and encrypt DEKs with KPABE. Using KP-ABE, we are able to immediately enjoy fine-grained data access control and efficient operations such as file creation/deletion and new user grant. To resolve the challenging issue of user revocation, we combine the technique of proxy re-encryption with KP-ABE and delegate most of the burdensome computational task to Cloud Servers. We achieve this by letting Cloud Servers keep a partial copy of each user's secret key, i.e., secret key components of all but one (dummy) attributes. This enhancement releases the data owner from the possible huge computation overhead on user revocation. The data owner also does not need to always stay online since Cloud Servers will take over the burdensome task after having obtained the PRE keys. To further save computation overhead of Cloud Servers on user revocation, we use the technique of lazy re-encryption and enable Cloud Servers to "aggregate" multiple successive secret key update/file re-encryption operations into one, and thus statistically save the computation overhead.

5. ANALYSIS OF OUR PROPOSED SCHEME

5.1 Security Analysis

We first analyze security properties of our proposed scheme, starting with the following immediately available properties.

1) Fine-grainedness of Access Control: In our proposed scheme, the data owner is able to define and enforce expressive and flexible access structure for each user. Specifically, the access structure of each user is defined as a logic formula over data file attributes, and is able to represent any desired data file set.
2) User Access Privilege Confidentiality: Our proposed scheme just discloses the leaf node information of a user access tree to Cloud Servers. As interior nodes of an access tree can be any threshold gates and are unknown to Cloud Servers, it is hard for Cloud Servers to recover the access structure and thus derive user access privilege information.
3) User Secret Key Accountability: This property can be immediately achieved by using the enhanced construction of KP-ABE which can be used to disclose the identities of key abusers. Now we analyze data confidentiality of our proposed scheme by giving a cryptographic security proof.
4) Data Confidentiality: We analyze data confidentiality of our proposed scheme by comparing it with an intuitive scheme in which data files are encrypted using symmetric DEKs, and DEKs are direclty encrypted using standard KP-ABE. In this intuitive scheme just ciphertexts of data files are given to Cloud Servers. Assuming the symmetric key algorithm is secure, e.g., using standard symmtric key algorithm such as AES, security of this intuitive scheme is merely relied on the security of KP-ABE. Actually, the standard KP-ABE is provably secure under the attribute-based Selective-Set model given the Decisional Bilinear Diffie-Hellman (DBDH) problem is hard.

6. CONCLUSION

This paper aims at fine-grained data access control in cloud computing. One challenge in this context is to achieve finegrainedness, data confidentiality, and scalability simultaneously, which is not provided by current work. In this paper we propose a scheme to achieve this goal by exploiting KPABE and uniquely combining it with techniques of proxy re-encryption and lazy re-encryption. Moreover, our proposed scheme can enable the data owner to delegate most of computation overhead to powerful cloud servers. Confidentiality of user access privilege and user secret key accountability can be achieved. Formal security proofs show that our proposed scheme is secure under standard cryptographic models.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: A berkeley view of cloud computing,” University of California, Berkeley, Tech. Rep. USB-EECS-2009-
28, Feb 2009.
[2] Amazon Web Services (AWS), Online at http://aws. amazon.com.
[3] Microsoft Azure, http://www.microsoft.com/azure/.
[4] 104th United States Congress, “Health Insurance Portability and Accountability Act of 1996 (HIPPA),” Online at http://aspe.hhs.gov/admnsimp/pl104191.htm, 1996.
[5] H. Harney, A. Colgrove, and P. D. McDaniel, “Principles of policy in secure groups,” in Proc. of NDSS’01, 2001.
[6] P. D. McDaniel and A. Prakash, “Methods and limitations of security policy reconciliation,” in Proc. of SP’02, 2002.
[7] T. Yu and M. Winslett, “A unified scheme for resource protection in automated trust negotiation,” in Proc. of SP’03, 2003.
[8] J. Li, N. Li, and W. H. Winsborough, “Automated trust negotiation using cryptographic credentials,” in Proc. of CCS’05, 2005.
[9] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, “Scalable secure file sharing on untrusted storage,” in Proc. Of FAST’03, 2003.
[10] 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.
[11] 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.
[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proc. Of CCS’06, 2006.
[13] M. Atallah, K. Frikken, and M. Blanton, “Dynamic and efficient key management for access hierarchies,” in Proc. of CCS’05, 2005.