International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 14
October2016

Data Sharing in the Cloud Using The Scalable Data with The Secret key Generation and to Store Data in The Cloud

Mr.GKV Narasimha, Mr C, CHANDRANNA

ABSTRACT:

Data sharing is an important functionality in cloud storage. In this paper, we show how to securely, efficiently, and flexibly share data with others in cloud storage. We describe new public-key cryptosystems that produce constant-size ciphertext is such that efficient delegation of decryption rights for any set of ciphertext is are possible. The novelty is that one can aggregate any set of secret keys and make them as compact as a single key, but encompassing the power of all the keys being aggregated. In other words, the secret key holder can release a constant-size aggregate key for flexible choices of cipher text set in cloud storage, but the other encrypted files outside the set remain confidential. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. We provide formal security analysis of our schemes in the standard model. We also describe other application of our schemes. In particular, our schemes give the first public-key patient-controlled encryption for flexible hierarchy, which was yet to be known.

Index Terms:
Cloud storage, data sharing, key-aggregate encryption, patient-controlled encryption

1 INTRODUCTION

CLOUD storage is gaining popularity recently. In enterprise settings, we see the rise in demand for data outsourcing, which assists in the strategic management of corporate data. It is also used as a core technology behind many online services for personal applications. Nowadays, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25 GB (or a few dollars for more than 1 TB). Together with the current wireless technology, users can access almost all of their files and emails by a mobile phone in any corner of the world Considering data privacy, a traditional way to ensure it is to rely on the server to enforce the access control after authentication (e.g., [1]), which means any unexpected privilege escalation will expose all data. In a sharedtenancy cloud computing environment, things become even worse.

Data from different clients can be hosted on separate virtual machines (VMs) but reside on a single physical machine. Data in a target VM could be stolen by instantiating another VM co resident with the target one [2]. Regarding availability of files, there are a series of cryptographic schemes which go as far as allowing a third-party auditor to check

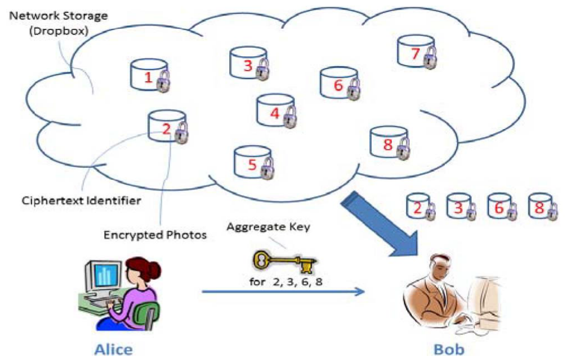
The availability of files on behalf of the data owner without leaking anything about the data [3], or without compromising the data owner's anonymity [4]. Likewise, cloud users probably will not hold the strong belief that the cloud server is doing a good job in terms of confidentiality. A cryptographic solution, for example, [5], with proven security relied on number-theoretic assumptions is more desirable, whenever the user is not perfectly happy with trusting the security of the VM or the honesty of the technical staff. These users are motivated to encrypt their data with their own keys before uploading them to the server.

Data sharing is an important functionality in cloud storage. For example, bloggers can let their friends view a subset of their private pictures; an enterprise may grant her employees access to a portion of sensitive data. The challenging problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, but it loses the value of cloud storage. Users should be able to delegate the access rights of the sharing data to others so that they can access these data from the server directly. However, finding an efficient and secure way to

share partial data in cloud storage is not trivial. Below we will take Dropbox1 as an example for illustration.

1.1 Our Contributions

In modern cryptography, a fundamental problem we often study is about leveraging the secrecy of a small piece of knowledge into the ability to perform cryptographic functions (e.g., encryption, authentication) multiple times. In this paper, we study how to make a decryption key more powerful in the sense that it allows decryption of multiple cipher texts, without increasing its size. Specifically, our problem statement is



cipher text called class. That means the cipher texts are further categorized into different classes. The key owner holds a master-secret called master-secret key, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of cipher text classes. With our solution, Alice can simply send Bob a single aggregate key via a secure e-mail. Bob can download the encrypted photos from Alice's Drop box space and then use this aggregate key to decrypt these encrypted photos. The scenario is depicted in Fig. 1. The sizes of cipher text, public-key, master-secret key, and aggregate key in our KAC schemes are all of constant size. The public system parameter has size linear in the number of cipher text classes, but only a small part of it is needed each time and it can be fetched on demand from large (but no confidential) cloud storage.

Previous results may achieve a similar property featuring a constant-size decryption key, but the classes need to conform to some predefined hierarchical relationship. Our work is flexible in the sense that this constraint is eliminated, that is, no special relation is required between the classes. The detail and other related works can be found in Section 3. We propose several concrete KAC schemes with different security levels and extensions in this paper. All constructions can be proven secure in the standard model. To the best of our knowledge, our aggregation mechanism2 in KAC has not been investigated.

2 KEY-AGGREGATE ENCRYPTION

We first give the framework and definition for key aggregate encryption. Then we describe how to use KAC in a scenario of its application in cloud storage.

2. It is obvious that we are not proposing an algorithm to compress the decryption key. On one hand, cryptographic keys come from a high entropy source and are hardly compressible. On the other hand, decryption keys for all possible combinations of cipher text classes are all in constant size—

information theoretically speaking such compression scheme cannot exist.

2.1 Framework

A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows. The data owner establishes the public system parameter via Setup and generates a public/master-secret3 key pair via Key Gen. Messages can be encrypted via Encrypt by anyone who also decides what cipher text class is associated with the plaintext message to be encrypted. The data owner can use the master-secret to generate a aggregate decryption key for a set of cipher text classes via Extract. The generated keys can be passed to delegates securely (via secure e-mails or secure devices) Finally, any user with an aggregate key can decrypt any cipher text provided that the cipher text's class is contained in the aggregate key via Decrypt. . Setupð1_; no: executed by the data owner to setup an account on an untrusted server. On input a security evel parameter 1_ and the number of ciphertext classes n (i.e., class index should be an integer bounded by 1 and n), it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.

KeyGen: executed by the data owner to randomly generate a public/master-secret key pair ðpk; mask. . Encryptðpk; imp: executed by anyone who wants to encrypt data. On input a public-key pk, an index I denoting the ciphertext class, and a message m, it outputs a ciphertext C. Extractðmsk; SÞ: executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate. On input the master-secret key msk and a set S of indices corresponding to different classes, it outputs the aggregate key for set S denoted by KS.

DecryptðKS; S; i; CÞ: executed by a delegate who received an aggregate key KS generated by Extract. On input KS, the set S, an index i denoting the ciphertext class the ciphertext C belongs to, and C, it outputs the decrypted result m if i 2 S.

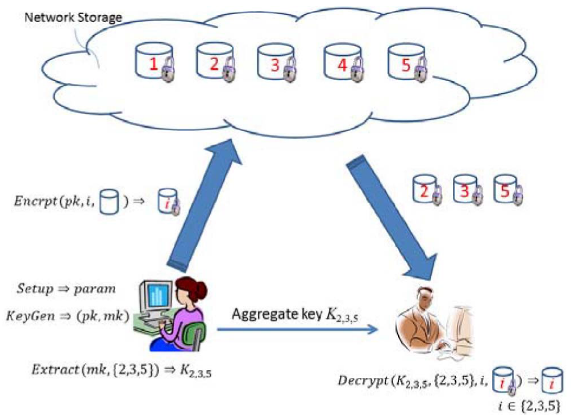

Network Storage
1
2
3
4
5
Encrpt(pk,i,) ⇒
2
3
5
Setup ⇒ param
KeyGen ⇒ (pk, mk)
Aggregate key $K_{2,3,5}$
Extract(mk, {2,3,5}) ⇒ $K_{2,3,5}$
Decrypt($K_{2,3,5}$, {2,3,5}, i,) ⇒
i ∈ {2,3,5}

Fig. 2. Using KAC for data sharing in cloud storage.

2.2 Sharing Encrypted Data

A canonical application of KAC is data sharing. The key aggregation property is especially useful when we expect the delegation to be efficient and flexible. The schemes enable a content provider to share her data in a confidential and selective way, with a fixed and small ciphertext expansion, by distributing to each authorized user a single and small aggregate key. Here, we describe the main idea of data sharing in cloud storage using KAC, illustrated in Fig. 2. Suppose Alice wants to share her data m1;m2; . . .;m_ on the server. She first performs Setupð1_; nÞ to get param and execute KeyGen to get the public/master-secret key pair ðpk; mskÞ. The system parameter param and public-key pk can be made public and master-secret key msk should be kept secret by Alice. Anyone (including Alice herself) can then encrypt each mi by Ci ¼ Encryptðpk; i;miÞ. The encrypted data are uploaded to the server.

With param and pk, people who cooperate with Alice can update Alice's data on the server. Once Alice is willing to share a set S of her data with a friend Bob, she can compute the aggregate key KS for Bob by performing Extractðmsk; SÞ. Since KS is just a constant-size key, it is easy to be sent to Bob via a secure e-mail. After obtaining the aggregate key, Bob can download the data he is authorized to access. That is, for each i 2 S, Bob downloads Ci (and some needed values in param) from the server. With the aggregate key KS, Bob can decrypt each Ci by DecryptðKS; S; i; CiÞ for each i 2 S.

3 CONCRETE CONSTRUCTIONS OF KAC

Let GG and GGT be two cyclic groups of prime order p and ^e : GG _ GG ! GGT be a map with the following properties: Bilinear: 8g1; g2 2 GG, a; b 2 ZZ, ^eðga1 ; gb 2Þ ¼ ^eðg1; g2Þab. . Nondegenerate: for some g 2 GG, ^e.g.; gÞ 6¼ 1. GG is a bilinear group if all the operations involved above are efficiently computable. Many classes of elliptic curves feature bilinear groups.

3.1 A Basic Construction

The design of our basic scheme is inspired from the collusion-resistant broadcast encryption scheme proposed by Boneh et al. [31]. Although their scheme supports constant-size secret keys, every key only has the power for decrypting ciphertexts associated to a particular index. We, thus, need to devise a new Extract algorithm and the corresponding Decrypt algorithm. Setupð1_; nÞ: Randomly pick a bilinear group GG of prime order p where 2_ _ p _ 2_þ1, a generator g 2 GG and _ 2R ZZp. Compute gi ¼ g_i 2 GG for i ¼ 1; . . . ; n; n þ 2; . . . ; 2n. Output the system parameter as param ¼ hg; g1; . . . ; gn; gnþ2; . . . ; g2ni (_ can be safely deleted after Setup). Note that each ciphertext class is represented by an index in the integer set f1; 2; . . . ; ng, where n is the maximum number of ciphertext classes. . KeyGen(): Pick _ 2R ZZp, output the public and master-secret key pair: ðpk ¼ v ¼ g_; msk ¼ _Þ. . Encryptðpk; i;mÞ: For a message 2 GGT and an index i 2 f1; 2; . . . ; ng, randomly pick t 2R ZZp and compute the ciphertext as C ¼ hgt; ðvgiÞt;m _ ^eðg1; gnÞti. . Extractðmsk ¼ _; SÞ: For the set S of indices j's, the aggregate key is computed as KS ¼Q j2S g_ nþ1_j. Since S does not include 0, gnþ1_j ¼ g_nþ1_j can always be retrieved from param. . DecryptðKS; S; i;C ¼ hc1; c2; c3iÞ: If i 62 S, output ?. Otherwise, return Q the message: m ¼ c3 _ e^ðKS _ j2S;j6¼i gnþ 1 _ j þ i; c1Þ=^eð Q j2S gnþ1_j; c2Þ. For the data owner, with the knowledge of _, the term ^eðg1; gnÞt can be easily recovered by ^eðc1; gnÞ_ ¼ ^eðgt; gnÞ_ ¼ ^eðg1; gnÞt. For correctness, we can see that c3 _ ^e KS _ Y j2S;j6¼I gnþ1_jþi; c1 !_ ^e Y j2S gnþ1_j; c2 ! ¼ c3 _ ^eð Q j2S g_ nþ1_j _ Q j2S;j6¼i gnþ1_jþi; gtÞ ^eð Q j2S gnþ1_j; ðvgiÞtÞ ¼ c3 _ ^e Y j2S;j6¼I gnþ1_jþi; gt !_ ^e Y j2S gnþ1_j; gti ! ¼ c3 _ ^eð Q j2S gnþ1_jþi; gtÞ=^eðgnþ1; gtÞ ^eð Q j2S gnþ1_jþi; gtÞ ¼ m _ ^eðg1; gnÞt=^eðgnþ1; gtÞ ¼ m:

4 NEW PATIENT-CONTROLLED ENCRYPTION (PCE)

Motivated by the nationwide effort to computerize America's medical records, the concept of patient-controlled encryption has been studied [8]. In PCE, the health record is Decomposed into a hierarchical representation based on the use of different ontologies, and patients are the parties who generate and store secret keys. When there is a need for a healthcare personnel to access part of the record, a patient will release the secret key for the concerned part of the record. In the work of Benaloh et al. [8], three solutions have been provided, which are symmetric-key PCE for fixed hierarchy (the "folklore" tree-based method in Section 3.1), public-key PCE for fixed hierarchy (the IBE analog of the folklore method, as mentioned in Section 3.1), and RSAbased symmetric-key PCE for "flexible hierarchy" (which is the "set membership" access policy as we explained). Our work provides a candidate solution for the missing piece, public-key PCE for flexible hierarchy, which the existence of an efficient construction was an open question.

Any patient can either define her own hierarchy according to her need, or follow the set of categories suggested by the electronic medical record system she is using, such as “clinic visits,” “x-rays,” “allergies,” “medications,” and so on. When the patient wishes to give access rights to her doctor, she can choose any subset of these categories and issue a single key, from which keys for all these categories can be computed. Thus, we can essentially use any hierarchy we choose, which is especially useful when the hierarchy can be complex. Finally, one healthcare personnel deals with many patients and the patient record is possible stored in cloud storage due to its huge size (e.g., high-resolution medical imaging employing x-ray), compact key size and easy key management are of paramount importance.

5 CONCLUSION AND FUTURE WORK

How to protect users’ data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In this paper, we consider how to “compress” secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegate can always get an aggregate key of constant size.

Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges. A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of ciphertexts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension. Otherwise, we need to expand the public-key as we described in Section 4.2. Although the parameter can be downloaded with ciphertexts, it would be better if its size is independent of the maximum number of ciphertext classes. On the other hand, when one carries the delegated keys around in a mobile device without using special trusted hardware, the key is prompt to leakage, designing a leakage-resilient cryptosystem [22], [34] yet allows efficient and flexible key delegation is also an interesting direction.

6 REFERENCES

[1] S.S.M. Chow, Y.J. He, L.C.K. Hui, and S.-M. Yiu, "SPICE – Simple Privacy-Preserving Identity-Management for Cloud Environment," Proc. 10th Int'l Conf. Applied Cryptography and

[2] L. Hardesty, Secure Computers Aren't so secure. MIT press, http:// www.physorg.com/news176107396.html, 2009.

[3] C. Wang, S.S.M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy- Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, vol. 62, no. 2, pp. 362-375, Feb. 2013.

[4] B. Wang, S.S.M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," Proc. IEEE 33rd Int'l Conf. Distributed Computing Systems (ICDCS), 2013.

[5] S.S.M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R.H. Deng, "Dynamic Secure Cloud Storage with Provenance," Cryptography and Security, pp. 442-464, Springer, 2012.

[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," Proc. 22nd Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT '03), pp. 416-432, 2003.

[7] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Trans. Information and System Security, vol. 12, no. 3, pp. 18:1-18:43, 2009.

[8] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," Proc. ACM Workshop Cloud Computing Security (CCSW '09), pp. 103-114, 2009.

[9] F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles," Proc. Information Security and Cryptology (Inscrypt '07), vol. 4990, pp. 384-398, 2007.

[10] V. Goyal, O. Pandey, A. Sahai and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), pp. 89-98, 2006.

[11]S.G. Akl and P.D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Trans. Computer Systems, vol. 1, no. 3, pp. 239-248, 1983.

[12] G.C. Chick and S.E. Tavares, "Flexible Access Control with Master Keys," Proc.

Advances in Cryptology (CRYPTO '89), vol. 435, pp. 316-322, 1989.

[13] W.-G. Tzeng, "A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy," IEEE Trans. Knowledge and Data Eng., vol. 14, no. 1, pp. 182-188, Jan./Feb. 2002.

[14] G. Ateniese, A.D. Santis, A.L. Ferrara, and B. Masucci, "Provably- Secure Time-Bound Hierarchical Key Assignment Schemes,"J. Cryptology, vol. 25, no. 2, pp. 243-270, 2012.

[15] R.S. Sandhu, "Cryptographic Implementation of a Tree Hierarchy for Access Control," Information Processing Letters, vol. 27, no. 2, pp. 95-98, 1988.

Author's Profile

Mr.GKV Narasimha Reddy received Ph.D from Annamalai University and received M.tech degree from Sathyabama University. He is currently working as Associate professor, Department of CSE, in St.Johns College of engineering and technology, Kurnool, Andhra Pradesh, India. His interests include Computer Networks, Operating system, Data Base Management Systems.

Mr C CHANDRANNAreceived **B.Tech** Degree from **ST.JOHNS COLLEGE OF ENGINEERING AND TECHNOLOGY** in yerrakota,Yemmiganur he is currently pursuing **M.Tech** Degree in Computer Science and Engineering specialization in **ST.JOHNS COLLEGE OF ENGINEERING AND TECHNOLOGY** in Yerrakota,Yemmiganur **Kurnool Dist ,ANDHRAPRADESH ,INDIA.**