Secured Data Sharing System using Key Aggregate Cryptosystem in Cloud

Sasiniveda G.
Assistant Professor, Sri Krishna
College of Engineering and Technology,
Kuniyamuthur,
Coimbatore-641008,Tamil Nadu

ABSTRACT

Data sharing is an important functionality in cloud storage. In this paper, it shows how to securely, efficiently, and flexibly share data with others in cloud storage. It describes new public-key cryptosystems that produce constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts is 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. It provides formal security analysis of our schemes in the standard model. It 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. 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.

Keywords

VirtualMachines, Cryptography, Encryption, Decryption, Cipertext, Plain text, Random oracles

1. INTRODUCTION

Cloud storage is gaining popularity recently. In enterprise settings, 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. 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, which means any unexpected privilege escalation will expose all data. In a shared-tenancy 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 coresident 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 owners 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 it will take Dropbox1 as an example for illustration. Assume that Alice puts all her private photos on Dropbox, and she does not want to expose her photos to everyone. Due to various data leakage possibility Alice cannot feel relieved by just relying on the privacy protection mechanisms provided by Dropbox, so she encrypts all the photos using her own keys before uploading. One day, Alice's friend, Bob, asks her to share the photos taken over all these years which Bob appeared in. Alice can then use the share function of Dropbox, but the problem now is how to delegate the decryption rights for these photos to Bob. A possible option Alice can choose is to securely send Bob the secret keys involved. Naturally, there are two extreme ways for her under the traditional encryption paradigm:

1) Alice encrypts all files with a single encryption key and gives Bob the corresponding secret key directly.

2) Alice encrypts files with distinct keys and sends Bob the corresponding secret keys.

Obviously, the first method is inadequate since all unchosen data may be also leaked to Bob. For the second method, there are practical concerns on efficiency. The number of such keys is as many as the number of the shared photos, say, a thousand. Transferring these secret keys inherently requires a secure channel, and storing these keys requires rather expensive secure storage. The costs and complexities involved generally increase with the number of the decryption keys to be shared. In short, it is very heavy and costly to do that. Encryption keys also come with two flavors—symmetric key or asymmetric (public) key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryptor her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in publickey encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can upload encrypted data on the cloud storage server without the knowledge of the company's master-secret key.

Therefore, the best solution for the above problem is that Alice encrypts files with distinct public-keys, but only sends Bob a single (constant-size) decryption key. Since the decryption key should be sent via a secure channel and kept secret, small key size is always desirable. For example, it cannot expect large storage for decryption keys in the Resource-constraint devices like smart phones, smart cards, or wireless sensor nodes. Especially, these secret keys are usually stored in the tamper-proof memory, which is relatively expensive. The present research efforts mainly focus on minimizing the communication requirements (such as bandwidth, rounds of communication) like aggregate signature [6]. However, not much has been done about the key itself.

2. EXISTING SYSTEM

In the current existing system, a user provides an untrusted server, say a proxy operated by a cloud service provider, with a transformation key TK that allows the latter to translate any ABE ciphertext CT satisfied by that user's attributes or access policy into a simple ciphertext CT', and it only incurs a small overhead for the user to recover the plaintext from the transformed ciphertext CT'. The security property of the ABE scheme with outsourced decryption guarantees that an adversary (including the malicious cloud server) be not able to learn anything about the encrypted message; however, the scheme provides no guarantee on the correctness of the transformation done by the cloud server. In the cloud computing setting, cloud service providers may have strong financial incentives to return incorrect answers, if such answers require less work and are unlikely to be detected by users.

One of the main efficiency drawbacks of the most existing ABE schemes is that decryption is expensive for resource-limited devices due to pairing operations, and the number of pairing operations required to decrypt a ciphertext grows with the complexity of the access policy.

The above observation motivates us to study ABE with verifiable outsourced decryption in this thesis work. Here emphasized that an ABE scheme with secure outsourced decryption does not necessarily guarantee verifiability (i.e., correctness of the transformation done by the cloud server).

It considered the verifiability of the cloud's transformation and provided a method to check the correctness of the transformation. However, not formally define verifiability. But it is not feasible to construct ABE schemes with verifiable outsourced decryption following the model defined in the existing. Moreover, the method proposed in existing relies on random oracles (RO). Unfortunately, the RO model is heuristic, and a proof of security in the RO model does not directly imply anything about the security of an ABE scheme in the real world. It is well known that there exist cryptographic schemes which are secure in the RO model but are inherently insecure when the RO is instantiated with any real hash function.

In this thesis work, firstly modify the original model of ABE with outsourced decryption in the existing to allow for verifiability of the transformations. After describing the formal definition of verifiability, it propose a new ABE model and based on this new model construct a concrete ABE scheme with verifiable outsourced decryption. Our scheme does not rely on random oracles.

In this paper it only focuses on CP-ABE with verifiable outsourced decryption. The same approach applies to KP-ABE with verifiable outsourced decryption.To assess the performance of our ABE scheme with verifiable outsourced decryption, it implement the CP-ABE scheme with verifiable outsourced decryption and conduct experiments on both an ARM-based mobile device and an Intel-core personal computer to model a mobile user and a proxy, respectively.

3. ARCHITECTURE DESIGN

The architecture of the key aggregate cryptosystem is shown below where the data, attributes and key are encrypted and stored in cloud and the aggregate key is developed and sent to the receiver's mail and it is used to download the desired pictures. The corresponding expression is then represented via alert dialog box.

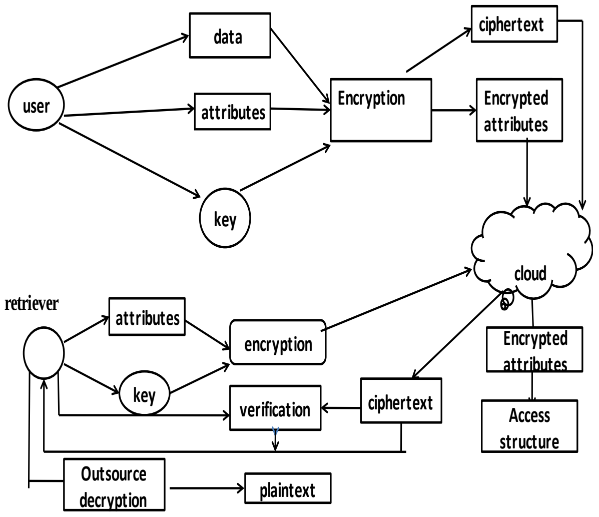

Fig 1:-Architecture of Key Aggregate System

3.1 Setup Phase

The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK

3.2 Encrypt Phase

Encrypt (PK,M, A). The encryption algorithm takes as input the public parameters PK, a message M, and an access structure A over the universe of attributes. The algorithm will encrypt M and produce a ciphertext CT such that only a user that possesses a set of attributes that satisfies the access structure will be able to decrypt the message. It will assume that the ciphertext implicitly contains A.

3.3 Key Gen Phase
Key Generation(MK,S). The key generation algorithm takes as input the master key MK and a set of attributes S that describe the key. It outputs a private key SK

3.4 Decrypt Phase
Decrypt (PK, CT, SK). The decryption algorithm takes as input the public parameters PK, a ciphertext CT, which contains an access policy A, and a private key SK, which is a private key for a set S of attributes. If the set S of attributes satisfies the access structure A then the algorithm will decrypt the ciphertext and return a message M.

The diagram explains the working of the login process. If the user is an existing user, then he logs in with the user name and password. Else the user registers himself/ herself as an registered user and then the database checks if the logging in user is a valid user or not

Table 1. System Specifications

Hardware requirements	Specifications
Hard Drive	20 GB
Processor	1GHz x86-64 processor (64-bit) 1GHz IA-32 processor (32-bit)
Memory (RAM)	1 GB (Required) 2 GB (Recommended)

3.5 Software Requirements
- Operating System: Microsoft Windows 7 x64 (Any edition) (Compatible with Microsoft Windows XP, Microsoft Windows Vista x64)
- Front End: Java 7.1
- Back End: SQL Server 2005

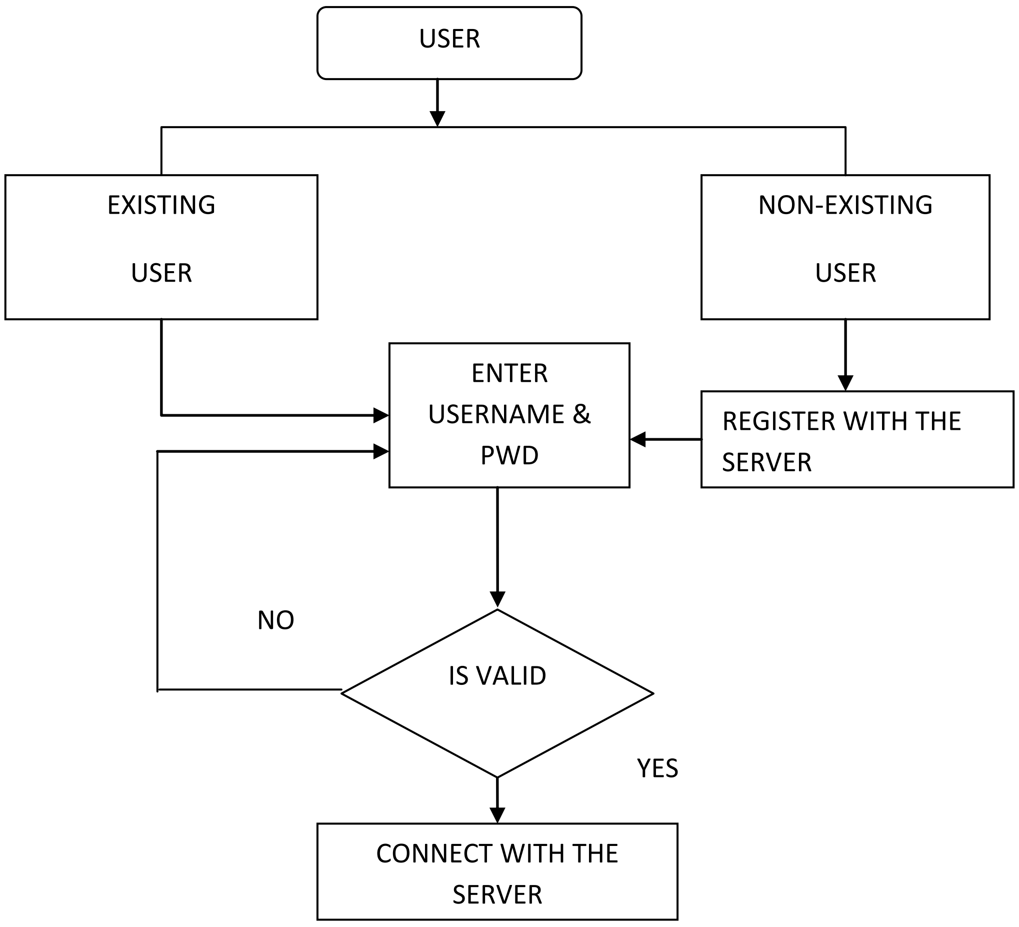

Fig 2: System Flow Diagram

4. CONCLUSION
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, it considers 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 delegatee 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.

5. FUTURE WORK
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 it have to reserve enough ciphertext classes for the future extension. Otherwise, it need to expand the public-key as it described.

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 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 Network Security (ACNS), vol. 7341, pp. 526-543, 2012.

[2] 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.

[3] 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

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

[5] 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.

[6] 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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.