www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 4 April 2016, Page No. 16288-16292

An Efficient System for Scalable Data Sharing In Cloud Storage

Jenolin Rex M [1], Latha P S [2]

Assistant professor, Department of Computer science and Engineering,
Mahendra College of Engineering,Salem.
jenolinrexm@mahendracollege.com

Assistant professor, Department of Computer science and Engineering,
Mahendra College of Engineering,Salem.
latha.p.srinivasmurthy@gmail.com

ABSTRACT

Using the cloud storage, users store their data on the cloud without the burden of data storage and maintenance and services and high-quality applications from a shared pool of configurable computing resources. Cryptography is probably the most important aspect of communications security and is becoming increasingly important as a basic building block for computer security. As data sharing is an important functionality in cloud storage, In this paper we show that how to securely, efficiently and flexibly share data with others in cloud storage, Cloud-validation based Flexible Distributed, Migration, ciphertext with aggregate key encryption for data stored in cloud. This scheme provides secure data storage and retrieval. Along with the security the access policy is also hidden for hiding the user's identity. This scheme is so powerful since we use aggregate encryption and string matching algorithms in a single scheme. The scheme detects any change made to the original file and if found clear the error's. The algorithm used here are very simple so that large number of data can be stored in cloud without any problems. The security, authentication, confidentiality are comparable to the centralized approaches. A set of constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts is possible the best.

Index Terms - String matching algorithms, Cloud-validation based Flexible Distributed, Migration, and cipher text.

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. 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 co-resident with the target one . 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, or without compromising the data owners anonymity. 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 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. 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.

Here is an efficient approach to secure the data for scalable data sharing in cloud.

2. EXISTING SYSTEM

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 co-resident with the target one. 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, or without compromising the data owner's anonymity. 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, 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.

2.1 DRAWBACKS

- Unexpected privilege escalation will expose all
- It is not efficient
- Shared data will not be secure

3. PROPOSED SYSTEM

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, we 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. However, not much has been done about the key itself.

3.1 ADVANTAGES

- It is more secure.
- Decryption key should be sent via a secure channel and kept secret.
- It is an efficient public-key encryption scheme which supports flexible delegation.

4. SYTEM ARCHITECTURE DESIGN

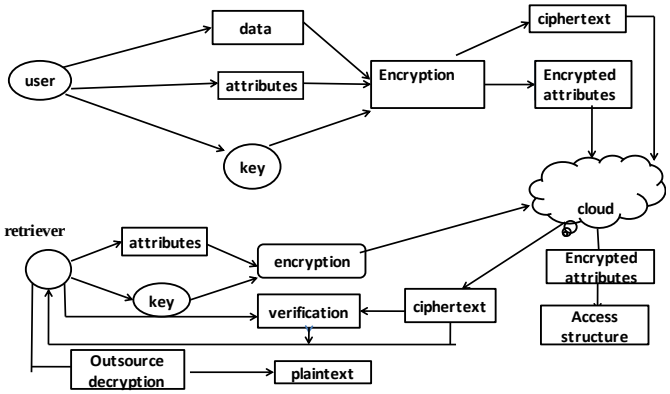
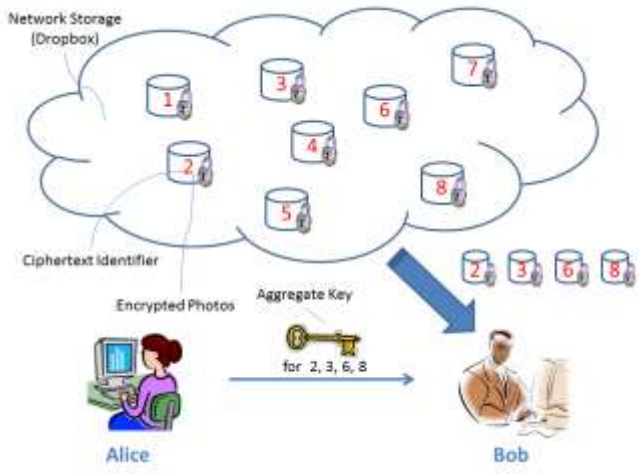


Fig 4.1: System Architecture Design

5. PROPOSED DATA SHARING SYSTEM IN CLOUD



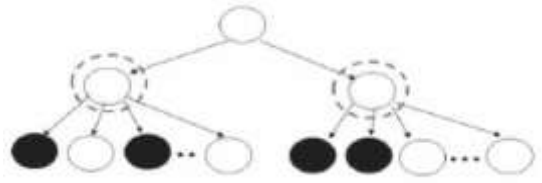


Fig 5.1: Proposed data sharing system &Key Assignment

6. TECHNIQUES

6.1 STRING MATCHING ALGORITHM

We show that how to securely, efficiently and flexibly share data with others in cloud storage, Cloud-validation based Flexible Distributed, Migration, ciphertext with aggregate key encryption for data stored in cloud. This scheme provides secure data storage and retrieval. Along with the security the access policy is also hidden for hiding the user's identity. This scheme is so powerful since we use aggregate encryption and string matching algorithms in a single scheme. The scheme detects any change made to the original file and if found clear the error's. The algorithm used here are very simple so that large number of data can be stored in cloud without any problems. The security, authentication, confidentiality are comparable to the centralized approaches. A set of constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts is possible the best.

6.2 AGGREGATE KEY ENCRYPTION ALGORITHM

A. Framework

The basis or outline of the key-aggregate encryption scheme consists of five polynomial-time algorithms, which are elucidated below: Setup ensures that the owner of the data can construct the public system stricture or parameter. KeyGen, as the name suggests generates a public/master secret (not to be confused with the delegated key explained later) key pair. By using this public and master-secret key cipher text class index he can convert plain text into cipher text via use of Encrypt. Using Extract, the master-secret can be utilized to generate an aggregate decryption key for a set of cipher text classes. These generated keys can be safely transported to the appointees by use of secure mechanisms with proper security measures adhered to. If and only if the cipher text's class index is enclosed in the single key, then every user with an aggregate key can decrypt the given cipher text provided through the use of Decrypt.

B. Algorithm

1. Setup(Security level parameter, number of cipher text classes): Setup ensures that the owner of the data can construct the public system stricture or parameter he create account on cloud. After entering the input, the total of cipher text classes n and a security level parameter 1, the public system parameter is given as output, which usually skipped from the input of other algorithms for the purpose of conciseness.

2. KeyGen: it is for generation of public or master key secret pair.

3. Encrypt(public key,index,message):run any person who want to convert plaintext into cipher text using public and master-secret key

4. Extract(master key, Set): Give input as master secret key and S indices of different cipertext class it produce output aggregate key. This is done by executing extract by the data owner himself. The output is displayed as the aggregate key represented by Ks, when the input is entered in the form the set S of indices relating to the various classes and mastersecret key msk.

5. Decrypt (Ks,S,i,C): When an appointee receives an aggregate key Ks as exhibited by the previous step, it can execute Decrypt. The decrypted original message m is displayed on entering Ks, S, i, and C, if and only if I belongs to the set S.

7. CONCLUSION

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 co-resident with the target one. 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, or without compromising the data owner's anonymity. 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, 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.

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, we 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. However, not much has been done about the key itself..

8. 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," in Applied Cryptography and Network Security – ACNS 2012, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.

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

[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, 2013.

[4] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in International Conference on Distributed Computing Systems - ICDCS 2013. IEEE, 2013.

[5] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, ser. LNCS, vol. 6805. Springer, 2012, pp. 442–464.

[6] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in Proceedings of Advances in Cryptology - EUROCRYPT '03, ser. LNCS, vol. 2656. Springer, 2003, pp. 416–432.

[7] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," ACM Transactions on Information and System Security (TISSEC), vol. 12, no. 3, 2009.

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

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

[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted data," in Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06). ACM, 2006, pp. 89–98.

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

[12] G. C. Chick and S. E. Tavares, "Flexible Access Control with Master Keys," in Proceedings of Advances in Cryptology

– CRYPTO ’89, ser. LNCS, vol. 435. Springer, 1989, pp. 316–322.

[13] W.-G. Tzeng, “A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy,” IEEE Transactions on Knowledge and Data Engineering (TKDE), vol. 14, no. 1, pp. 182–188, 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.