Experimental Study on Key –Aggregate Searchable Encryption (KASE) For Group Data Sharing Via Cloud Storage By File Verification

Ms. Priyanka S.Upalanchi[1], Mrs. S.S.Joshi[2]
[1, 2] Dept of Computer Science & Engineering
[1, 2] N. B. Navale Sinhgad College Of Engineering , Solapur 413255

Abstract- *Data sharing is an important functionality in cloud storage. In this article, we depict the way of securely, efficiently, and flexibly share data with others in cloud storage. We describe new public-key cryptosystems which produce constant-size cipher texts such that efficient delegation of decryption rights for any set of cipher texts 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 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.

Keywords- Wrapper maintenance, multilevel , extraction. public key, cryptosystem, encryption, cloud storage, SHA algorithm, cipher text.

I. INTRODUCTION

Cloud storage has emerged as a promising solving a problem for providing ubiquitous, convenient, and on-demand accesses to large amounts of data shared over the Internet. nowadays, millions of users are sharing personal data, such as photos and videos, with their friends through a dedicated website or other application which enables users to communicate with each and every other by posting information, messages, images based on cloud storage on a daily basis. Business users are also being attracted by cloud storage due to its too many benefits, including lower cost, greater agility, and better resource utilization. However, while enjoying the quality of being useful, easy, of sharing data via cloud storage, users are also increasingly worried about inadvertent data leaks in the cloud.

Such data leaks, caused by a malicious a misbehaving cloud operator, can usually lead to behave badly break or fail to observe of personal privacy or business secrets (e.g., the recent high probe incident of a famous person photos being leaked in iCloud). To address users relate to protect potential data leaks in cloud storage, a prevalent way of dealing with a situation is for the data owner to encrypt all the data before upload to cloud.

II. METHODOLOGY

This project follows the steps given below:

Step1: Setup and create the account on the server for sharing of data. This account is generated by data owner.
Step2: KeyGen algorithm is used for the generation of public key. The data owner generates a public secrete key to encrypt the data over cloud. It also creates an aggregate key to access the block of ciphers of limited size.
Step3: Encrypts the data provided by the data owner by using the secrete key. This encrypted data is then share among the cloud.
Step4: The aggregate key is used for extracting the particular block of the ciphers from the cipher file. But other encrypted data remains secure.
Step5: Decrypt: The encrypted data is then decrypted by using the same secrete key which is use for encryption.

1) AES Algorithm:

The encryption process uses a set of specially derived keys called round keys. These are applied, along with other operations, on an array of data that holds exactly one block of data the data to be encrypted. This array we call the state array. You take the following AES steps of encryption for a 128-bit block:

1. Derive the set of round keys from the cipher key.
2. Initialize the state array with the block data (plaintext).
3. Add the initial round key to the starting state array.
4. Perform nine rounds of state manipulation.
5. Perform the tenth and final round of state manipulation.
6. Copy the final state array out as the encrypted data (ciphertext).

The reason that the rounds have been listed as "nine followed by a final tenth round" is because the tenth round involves a slightly different manipulation from the others. These algorithm are used to file content are convert plaint text to cipher text.

2) HMAC Algorithm:

By using HMAC algorithm.we are generated signature at both sides i.e Admin side & User side for verifying of our file or Message is Safe or not in between the transfer file. It will generate Signature both side.so we can check integrity of message.

Similar to Message Digest protecting the integrity of a message

Generated by an algorithm that creates a small fixed-sized block

◦ depending on both message and some key
◦ like encryption though need not be reversible appended to message as a signature provides assurance that message is unaltered and comes from sender

1) Append zeros to the left end of K to create a b-bit string K^+ (for example, if K is of length 160 bits and $b = 512$, then K will be appended with 44 zero bytes 0x00).
2) XOR (bitwise exclusive OR) K^+ with *ipad* to produce the b-bit block S_i.
3) Append M to S_i.
4) Apply H to the stream generated in Step 3.
5) XOR K^+ with *opad* to produce the b-bit block S_o.
6) Append the hash result from Step 4 to S_o.
7) Apply H to the stream generated in Step 6 and output the result.

III. EXPERIMENTAL ANALASIS OF PROJECT:

S.N	Comparative Parameter	Existing	Proposed
1	Time required to decrypt file.	Time required :: 12500 ms	Time required : 892 ms
2	Key	Separate key is provided for each file	Only one key is provided for multiple files
3	Encryption Algorithm	RSA,AES	AES,with Key
4	Email Alert	No	Yes
5	Key Encryption	Key not encrypted	Key is encrypted
6	File Safety	Not Known	Known
7	Modification History	Not Available	Available,we can check whether file is modified

IV. EXPERIMENTAL RESULTS FOR FILE DESCRYPTION

As per growing industries, sensitive data is also growing rapidly and security of sensitive data is main issue and there are many solutions available to store such data. The main problem with current cloud provider is that a powerful attacker can breaks data confidentiality by acquiring cryptographic keys, by means of coercion or backdoors in cryptographic software

Exisiting System:

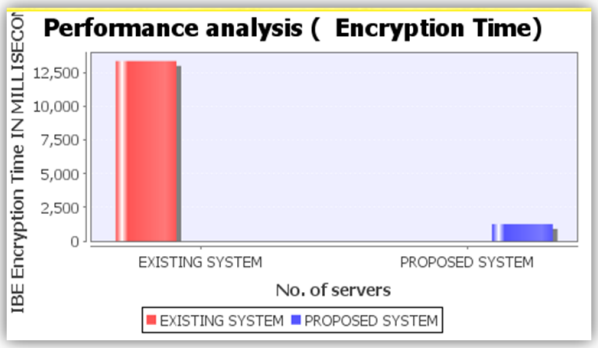
- In this approach we are using AES algorithm and encrypting files by using different public keys.
- For decryption of a file only single secret key is provided to intended user via mail.
- As we are mailing user the keys that ensures more security.

Proposed System:

- In this approach we are using same algorithm but with signature and using HMAC function which provides more security.
- Here encryption of file is done by using signature and encryption key.
- For decryption of files user get aggregate keys through mail by using that keys he can decrypt files.

Below graph shows the difference between existing and proposed system with respect to response time . Here

response time of proposed system is 1000 ms which less than existing system that is 2500 ms.



Performance analysis based on different parameters:

- As we can see in performance analysis graph, propose system is taking very less time as compared to existing system. As we know, proposed system is also more secure as compared to existing system.

- Here it is not only concern about time and security of data but in proposed system the system performance is also efficient as compared to existing system.

Encryption time based on different key size:

- The three different key sizes used are 128 bit, 192 bit and 256 bits. As the key size vary from 128 bits to 192 bits to 256 bits, the time required to encrypt a file also increases.

Decryption time based on different key size:

- Performance of AES algorithm in terms of decryption time based on different key sizes is as the key size vary from 128 bits to 192 bits to 256 bits, decryption time also increases.

Throughput:

- The throughput of an encryption scheme defines the speed of encryption. The throughput is calculated as the total plaintext in Kilo bytes encrypted/ encryption time (KB/sec).As the throughput increases, power consumption decreases.
- Throughput for AES decreases as key size increases because of more usage of computational power and encryption characteristics.

CPU process time:

- The CPU process time is the time that a CPU is dedicated only to the particular process for calculations. It reflects the load of the CPU. More the CPU time used in the encryption process, the higher is the CPU load.
- AES need more time to encrypt large size files. Hence the CPU process time that is the CPU load is higher.
- Same as encryption the time required to decrypt files in AES algorithm is also high hence the CPU process time is high in decryption too.

Memory Utilization:

- The Memory Utilization defines how much memory is being consumed while doing the encryption or decryption.
- AES consume more memory because of its characteristics. And as the file size increases memory size is drastically increased in AES means for extra-large files, we need a system with good memory and more CPU.
- In case of decryption, the memory requirement is more, the system with good memory can give you good output in terms of memory utilization.

V. CONCLUSION

Considering the practical problem of privacy preserving data sharing system based on public cloud storage which requires a data owner to distribute a large number of keys to users to enable them to access his/her documents, we for the first time propose the concept of key-aggregate searchable encryption (KASE) and construct a concrete KASE scheme. Both analysis and evaluation results confirm that our work can provide an effective solution to building practical data sharing system based on public cloud storage. In a KASE scheme, the owner only needs to distribute a single key to a user when sharing lots of documents with the user and the user only needs to submit a single trapdoor when he queries over all documents shared by the same owner. However, if a user wants to query over documents shared by multiple owners, he must generate multiple trapdoors to the cloud. How to reduce the number of trapdoors under multi-owners setting is a future work. Moreover, federated clouds have attracted a lot of attention nowadays, but our KASE cannot be applied in this case directly. It is also a future work to provide the solution for KASE in the case of federated clouds.

REFERENCES

[1] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving Secure,

Scalable, and Fine-Grained Data Access Control in Cloud Computing",Proc. IEEE INFOCOM, pp. 534-542, 2010.

[2] R. Lu, X. Lin, X. Liang, and X. Shen, "Secure Provenance:The Essential of Bread and Butter of Data Forensics in Cloud Computing", Proc. ACM Symp. Information, Computer and Comm. Security, pp. 282-292, 2010

[3] X. Song, D.Wagner, A. Perrig. "Practical techniques for searches on encrypted data", IEEE Symposium on Security and Privacy,IEEE Press, pp. 44C55, 2000.

[4] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky. "Searchable symmetric encryption: improved definitions and efficient constructions", In: Proceedings of the 13th ACM conference on Computer and Communications Security, ACM Press, pp. 79-88, 2006.

[5] S. Kamara, C. Papamanthou, T. Roeder. "Dynamic searchable symmetric encryption", Proceedings of the 2012 ACM conference on Computer and communications security (CCS), ACM, pp. 965-976, 2012.

[6] D. Boneh, C. G, R. Ostrovsky, G. Persiano. "Public Key Encryption with Keyword Search", EUROCRYPT 2004, pp.506C522,2004..

[7] Z. Liu, Z. Wang, X. Cheng, et al. "Multi-user Searchable Encryption with Coarser-Grained Access Control in Hybrid Cloud", Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), IEEE, pp. 249-255, 2013.

[8] T. C. Landgrebe, D. M. Tax, P. Paclik, and R. P. Duin, The interaction between classi cation and reject perfor mance for distancebased reject-option classi ers, Pattern Recognition Letters, vol. 27, no. 8, pp. 908917, 2006.

[9] J. Demsar, Statistical comparisons of classi ers over multi ple data sets, Journal of Machine Learning Research, vol. 7, pp. 130, 2006.