Secure Data Sharing in Cloud Storage for Improving the Efficiency of Data using Key Compressed Cryptosystem

Ramya. M
PG-Information Technology
Jayam college of Engg and Tech
Dharmapuri,India.

Krishnan. K
Assitant Professor
Jayam college of Engg and Tech
Dharmapuri,India.

***Abstract*—Incloud storage data sharing is a significant functionality. In that we are going to see how securely share the data with others in cloud storage. To securely share the data we depict one new public key cryptosystems that acquire fixed-size cipher texts such that delegation of decryption rights for any set of cipher texts is possible. In this one can congeries any set of private keys and make .them as compress as a single key. That single key encompassing the index of all the keys compressed in cloud storage, but the remaining encrypted files outside the cloud storage is confidential. This compressed key can be handily sent to others or stored by using smart card with very limited storage.**

Key words: Cloud Storage; data storage;public key crypto system

I. INTRODUCTION

Cloud storage is reaching popularity recently. Considering data privacy, a conventional way to ensure it is rely on the server to apply access control after Authentication. Data can be hosted on separate virtual machines from different clients but occupy on a single physical machine. Concerning availability of file, there are a many of the cryptographic schemes available which go as far as possible to allowing a third party auditor to check the accessibility of files on behalf of the data owner without leaking anything about the data.

A. EXISTING SCHEME

In cloud storage bloggers can let their friends view a subset of their private pictures. A portion of the sensitive data can be access by the employees with the permission of that enterprise. In this the problem is how effectively share encrypted data. Naturally users can download the encrypted data from the cloud storage, then decrypt them and share with others by sending the data. This should be loses the value of cloud storage. In this users can access the data directly from the server. Even so, detecting an efficient and secure way to share partial data in cloud storage is important.

There are two extreme ways for the traditional encryption paradigm:

For example,

1 .Using the single key Alice can encrypt all files and secret key can be directly given to Bob.

2. Alice encrypts files with discrete keys and sends Bob the equivalent secret keys.

Clearly, the first method is insufficient since all unpreferred data may be also leaked to Bob. For the second method, there are useful concerns on efficiency. The number of such keys equal tothe number of the shared photos. A secure channel can be used for transferring the secret key, and storing these keys requires ratherrestrictedprotected storage. The expenditure and complexities involved generally enlarge with the number of the decryption keys to be shared. There are two types of encryption keys are available—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; visibly, this is not always attractive. In asymmetric key encryption (public), the encryption key and decryption key are dissimilar in publickey encryption. The use of public-key encryption gives more elasticity for our applications. For example, in enterprise, eachworker can upload encrypted data on the cloud storage server lacking the knowledge ofcompany's master-secret key. Therefore, the most excellent solution for the above trouble is that Alice encrypts files with discrete public-keys, but only sends Bob a single (constant-size) decryption key.

Some disadvantages areIncreases the costs of storing and transmitting cipher texts.

Secret keys are usually stored in the tamper-proof memory, which is relatively expensive.

The costs and complexities involved generally increase with the number of the decryption keys to be shared.

II.LITERATURE REVIEW

In Cloud Storage, users can distantly store their data and benefit from the on-demand high quality applications and services from a shared pool of configurable computing possessions, without the burden of local data storage and maintenance.

.

To firmly introduce asuccessful TPA, the auditing method should convey in no new vulnerabilities to user data privacy, and begin no additional online trouble to user.Wepropose a safe cloud storage system following privacy-preserving public auditing. We further extend our outcome to enable the TPA to execute audits for multiple users concurrently and professionally. Wide security and

concert analysis show the planned schemes are provably protected and highly efficient. Storing thepooled Data on the Cloud via Security-Mediator,we suggest a simple, well-organized, and publicly verifiable advance to guarantee cloud data integrity not includingsacrifice the obscurityof neither data owners nor requiring important overhead. Exclusively, we initiate a security-mediator (SEM), which is able to produceproof metadata (i.e., signatures) on outsourced data for data owners.

Thus, a company can employ its own unidentifiedvalidation mechanism, and the cloud is unaware to that since it only deals with typical PDP-metadata, subsequently, the characteristics of the data owner is not exposed to the cloud, and there is no additional storage overhead distinct existing unknown PDP solutions.

The individual features of our method also incorporate data privacy, such that the SEM does not studysomething about the data to be uploaded to the cloud at all, and thus the belief on the SEM is minimized.

An aggregate signature proposal is a digital signature that wires aggregation: Given n signatures on n discrete messages from n different users, it is possible to combine all these signatures into a single short signature. This single signature willinfluence the verifier that the n users did definitely sign the n unique message. In this article we establish the concept of ancollective signature, present security models for such signatures, and give numerous applications for aggregate signatures. We create an efficient aggregate signature from a current short signature format based on bilinear maps due to Boneh, Lynn, and Shacham. Aggregate signatures are valuable for dropping the size of certificate chains (by aggregating all signatures in the chain) and for dropping message size in protected routing protocols such as SBGP. We also explain that aggregate signatures give increase to verifiably encrypted signatures. Such signatures allow the verifier to test that a given cipher text C is the encryption of a signature on a given message M. verifiably encrypted signatures are use in contract-signing protocols. Finally, we prove that similar information can be used to broaden the short signature proposal to give simple ring signatures.

II.PROPOSED METHODOLOGY

In this paper, we are going to attain a decryption key as more powerful in that it allows decryption of multiple cipher texts, without extend its size. We are introducing a public-key encryption which we call key-compressed cryptosystem (KCC). In KCC, users encrypt a message not only under a public-key, but also under an identifier of cipher text called class. The master secret can be hold by a key owner called master-secret key, different classes can be extracted by using the master secret key. More importantly, the extracted key can be acompressed key which is as compact as a secret key for a single class, but compress the power of all such keys. The sizes of cipher text, public-key, and master-secret key and compressed key in our KCC schemes are all of fixed size.

Some advantages

1. The delegation of decryption can be efficiently implemented with the aggregate key, which is only of fixed size.
2. Number of cipher text classes is large.
3. It is easy to key management.
4. A part from reduction in storage costs data outsourcing to the cloud also helps in reducing the maintenance.
5. Avoiding local storage of data.

By reducing the costs of storage, maintenance and personnel.

6. It reduces the chance of losing data by hardware failures.
7. Not cheating the owner.

KEY –COMPRESSED ENCRYPTION

In this we are going to see about the framework and definition of key –compressed cryptosystem (KCC).

A. Framework

KCC consist of five polynomial –time algorithms namely setparam, keyGen, Encrypt, Extract, decrypt.

a**.** SetParam (1^{λ}, n):

To setup an account with aun trusted server setparam algorithm can be used. It can be executed by the data owner. The input can be as follows

1^{λ}- security level parameter

n - Number of cipher text classes.

It output a public key parameter param. The data owner establishes the public system parameter via setparam.

b**.** KeyGen:

It can be used to generate a public/master secret key pair (p_k,msk) randomly. It is also executed by the data owner.

c**.** Encrypt (p_k, i, m):

It can be executed by anyone who wants to execute the data. on input a public key p_k, an index i, and a message m, it outputs a cipher texts called C.

d. Extract (msk, S):

It can be used to assigning the decrypting power for a certain set of ciphertext classes to a delegatee. On input the master-secret key msk and a set S of indices corresponding to different classes, it outputs the compressed key for set S denoted by K_s.

e**.** Decrypt (K_s, S, i, C):

It can be run by a receiver who received a compressed key K_s generated by Extract. On input Ks, the set S, an index i denoting the cipher text C belongs to, and C, it outputs the decrypted result m if I ϵS.

There are two functional requirements we are going to use for data sharing in cloud storage namely,

1. Correctness
2. Compactness

II. SHARING OF ENCRPTED DATA

Data sharing is the canonical application of KCC. When we expect the delegation to be more flexible and efficient that time we can use the key compressed property. This property can be used to enable the content provider for

sharing the data confidential with a constant and small cipher text expansion by circulating the authorized user a single and small compressed key. The main idea of data sharing in cloud storage using KCC is illustrated in below figure.

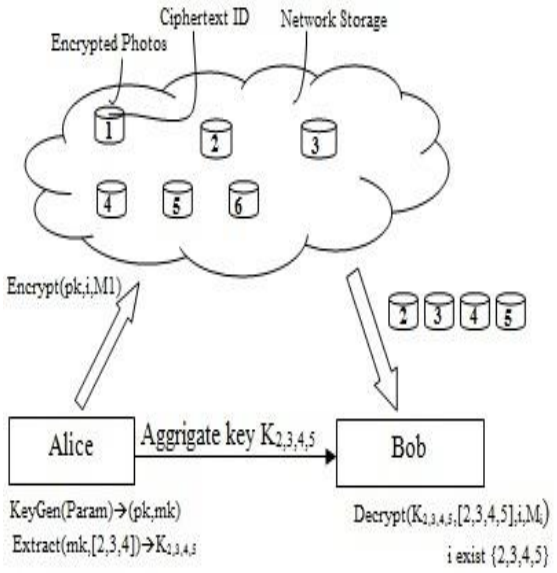

Fig 1: using KCC for data sharing in cloud storage

III.RELATED WORK

1. Cryptographic key assignment

Cryptographic key assignment schemes intend to minimize the price in storing and control secret keys for common cryptographic use. In a tree structure, a key for agreed branch can be used to obtain the keys of its successor nodes. Just yielding the parent key completely grants every key of its successor nodes. We projected amethodto produce a tree hierarchy of symmetric - keys byusing frequentevaluations of pseudorandom function/block cipher on a fixed secret. The theory can be comprehensive from a tree to a graph. More difficult cryptographic key assignment schemes sustain access policy that can be modelled by an acyclic graph or a cyclic graph. The majority of these methodsconstruct keys for symmetric-key cryptosystems.

2. Identity based encryption scheme

IBE is a kind of public-key encryption. In that an identity string of a user can be set as the public key of the user (e.g., an email address). There is a reliable party called private key author in IBE which holds a master-secret key and secret key can be issued to each user with revere to the user identity. The encryptor can take the user identity and public parameter to encrypt a message. The beneficiary can decrypt this ciphertext by his secret key. Guotried to build IBE with key aggregation.

3. Other encryption scheme

Attribute-based encryption (ABE) allows every ciphertext to be connected with afeature, and the master-secret key controller can take out a secret key for a policy of these attributes so that a ciphertext can be decrypted by this key if its connected attribute conforms to the policy. A PRE method allows Alice to assign to the server (proxy) the capacity to convert the ciphertexts encrypted below her public-key into ones for Bob. PRE is fine to have several applications with cryptographic file system. All the same, Alice has to belief the proxy that it only converts ciphertexts according to her order, which is what we desire to keep away from at the first place.

IMPLEMENTATION

A. Construction of KCC

Let G and G_T be the two cyclic groups of prime order p and e^: G*

G $\rightarrow G_T$. Be a map.

The collusion –resistant broadcast encryption is the basic design of our scheme and it supports fixed size secret keys, all the key only has the power for decrypting cipher text related to a particular index. For that we need to invent new Extract algorithm and corresponding Decryption algorithm as we discussed above.

B. Extract algorithm:

It can be executed by the owner of the data.

The extract algorithm can be of Extract (msk=γ, S). In this S indicates the set it can be indices j'S, the compressed key is calculated by

$K_s=\prod_{j\epsilon S} g^{\gamma}{}_{n+1-j.}$

C. Decrypt algorithm:

It can be executed by the delegatee of the data when he/she want to decrypt the data.it can be of the following form

Decrypt (K_s, S, i, C=<c1, c2,c3>)

If I €S

Output is ⊥

Otherwise return the message

$M=c3.e^{(K_s.\prod_{j\epsilon S, j\neq i} g_{n+1j+i,}c_i)/e^{(\prod_{j\epsilon S} gn+1-j,c2)}}$.

D. Extension of public key

If the users want to sort out their cipher texts into more than n classes, he needs to register for extra key pairs (pK_2, msK2)… (pK_r,msK_r). Now each class can be indexed by two-level indexed in {(i,j)| 1 ≤ i ≤r, 1≤ j ≤n} then the number of classes is extended by n for each keys.

In this we cannot attain key compression more than one user it is obviously not possible. But the secret key belongs to the same branch can be compressed by using 'local aggregation' method.

The proposed system considers the overall process as four sub processes.

The modules of the project are:

Cloud storage
Simply Archives
Sentinels
Verification Phase

A. Cloud Storage:

Data outsourcing to cloud storage servers is raising trend among many firms and users owing to its economic advantages. This essentially means that the owner (client) of the data moves its data to a third party cloud storage server which is supposed to - presumably for a fee - faithfully store the data with it and provide it back to the owner whenever required.

B. Simply Archives:

This problem tries to obtain and verify a proof that the data that is stored by a user at remote data storage in the cloud (called cloud storage archives or simply archives) is not modified by the archive and thereby the integrity of the data is assured. Cloud archive is not cheating the owner, if cheating, in this context, means that the storage archive might delete some of the data or may modify some of the data. While developing proofs for data possession at un trusted cloud storage servers we are often limited by the resources at the cloud server as well as at the client.

C.Sentinels:

In this scheme, unlike in the key-hash approach scheme, only a single key can be used irrespective of the size of the file or the number of files whose retrievability it wants to verify. Also the archive needs to access only a small portion of the file F unlike in the key-has scheme which required the archive to process the entire file F for each protocol verification. If the proven has modified or deleted a substantial portion of F, then with high probability it will also have suppressed a number of sentinels.

D. Verification Phase:

The verifier before storing the file at the archive, pre processes the file and appends some Meta data to the file and stores at the archive. At the time of verification the verifier uses this Meta data to verify the integrity of the data. It is important to note that our proof of data integrity protocol just checks the integrity of data i.e. if the data has been illegally modified or deleted. It does not prevent the archive from modifying the data.

III. PERFORMANCE OF OUR PROPOSED SCHEME

In this we are going to investigate about the space requirement of tree-based key assignment method. In this we can use the complete sub-tree scheme and it is the representative solution to the broadcast encryption problem. It materialized with the binary key tree of height h and it can support up to 2^h cipher texts classes, the selected part can be securely send to the authorized delegatee. The receiver can be granted access to $2^{hs}$with only one key, where share the height of the sub tree. In our approach the delegation should be chosen randomly.

The compression factor F to be tuneable parameter at the cost of O (n) is sized system parameter. an encryption should be done in fixed time , while decryption time should be done in O(ıSı) group multiplication with two pairing operations.

Table I. PERFORMANCE OF OUR BASIC CONSTRUCTION .WHERE H=16 WITH RESPECT TO DIFFERENT DELEGATION RATIO R.

R	0.1	0.2	0.3	.4	0.5	0.6	.7
Setparam	8.4						
Extract	2	4	5	7	8	9	10
Decrypt	4	6	9	12	14	15	16

Our approach can be used for fine grained data sharing because it can deal with up to 65,536 numbers of classes. Finally, in our proposed method the number of cipher text classes is large but limited storage of non-confidential data.

IV. KEY ASSIGNMENT

Key Revocation: A key owner always has the option of changing (essentially revoking) keys by decrypting portions of data locally and re-encrypting with new subkeys. This might be attractive if an owner suffers a key concession or wants to terminate access to her data for a particular source, or family member, or other proxy.

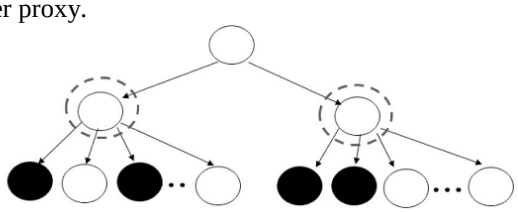

Fig 2. Key Assignment

1. Performance

For encryption, the value e^ðg1; gnÞ can be pre computed and put in the system parameter. On the other hand, we can see that decryption only takes two pairings while only one of them involves the aggregate key. That means we only need one pairing computation within the security chip storing the (secret) aggregate key. It is fast to compute a pairing nowadays, even in resource-constrained devices. Efficient software implementations exist even for sensor nodes.

2. Discussions

The “magic” of getting constant-size aggregate key and constant-size cipher text simultaneously comes from the linear-size system parameter. Our motivation is to reduce the secure storage and this is a trade off between two kinds of storage. The parameter can be placed in non confidential local storage or in a cache provided by the service company.

They can also be fetched on demand, as not all of them are required in all occasions.

The system parameter can also be generated by a trusted party, shared between all users and even hard-coded to the user program (and can be updated via “patches”). In this case, while the users need to trust the parameter-generator for securely erasing any ephemeral values used, the access control is still ensured by a cryptographic mean instead of relying on some server to restrict the accesses honestly.

V. CONCLUSION

In this paper we consider how to compress the secret keys in public-key cryptosystems and it support the delegation of secret keys in cloud storage. In that the receiver can always get the compressed key of fixed size. Our proposed scheme is more powerful than hierarchal key assignment method and it can be more flexible. The number of maximum cipher text class is predefined bound which is the limitation of our work. For future extension we have to reserve the cipher text classes because the number of cipher texts classes grows rapidly in cloud storage.

Even though the constraint can be downloaded with ciphertexts, it would be enhanced if its size is autonomous of the highest number of ciphertext classes. On the other hand, when one carries the delegated keys in the region of in a mobile device without using uniqueconfidential hardware, the key is without delay to leakage, designing a leakage-resilient cryptosystem yet allows proficient and flexible key designation is also an attractive direction.

REFERNCES

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

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

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

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

[5] Boneh.D, Gentry.C, Lynn.B, and Shacham.H, "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.

[6] Chick.G.C and Tavares.S.E, "Flexible Access Control with Master Keys," Proc. Advances in Cryptology (CRYPTO '89), vol. 435, pp. 316-322, 1989.

[7] Chow.S.S.M, He.Y.J, Hui.L.C.K, and Yiu.S.M, "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.

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

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

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