IJCTA, 10(02), 2017, pp. 13-20

Strong Individuality Based Encryption for Key Aggregate Cryptosystem in Cloud Storage

R. Danu* M. Gowthami* P. Senthil* and R. Raja Saranya Kumari*

Abstract : Data sharing is an essential functionality in cloud storage. In this article, we illustrate how to *securely, efficiently, and flexibly* share data through others in cloud storage. We define *new* public-key crypto systems which yield constant-size cipher texts such that effective delegation of decryption rights for any set of cipher texts are possible. The newness is that one can cumulative several set of secret keys and create them as compact as a single key, but incorporating the power of all the keys actuality aggregated. In other words, the secret key holder can issue a fixed-size aggregate key for stretchy choices of cipher text fixed in cloud storage, but the extra encrypted files outside the set persist confidential. This compact aggregate key can be appropriately sent to others or be deposited in a smart card with very narrow secure storage. On the, when one carries the delegated keys around in a mobile device without using special trusted hardware ,the key is prompt to leakage, we provide a leakage resilient cryptosystem which allows efficient and Flexible key delegation.

Keywords: Cloud storage, patient controlled encryption, data sharing, key-aggregate encryption, resilient key, identity based encryption.

1. INTRODUCTION

Cloud storage is achieving popularity recently. In enterprise surroundings, we realize the rise in demand for data out sourcing, which supports in the strategic administration of company data. It is alsoused as a core technology behind several online services for individual applications. Currently, it is easy to apply for unrestricted accounts for email, photo album, and file sharing and/or remote access, by storage size more than 25GB (or a few dollars for more than 1TB). Together with the existing wireless technology, users can access almost all of their files and emails by a mobile phone in anywhere of the world.

Considering data privacy, a traditional way to confirm it is to rely on the server to implement the access control after authentication, which means some unexpected privilege increase will expose all data. In a Distributed-tenancy cloud computing location, things become even worse. Data from various clients can be hosted on isolated virtual machines (VMs) but reside on a particular physical machine. Data in a target VM might be taken by instantiating supplementary VM co-resident with the target one. Regarding availability of files, there are a series of cryptographic schemes which drive as far as permitting a third-party auditor to find out the availability of files on behalf of the data proprietor without leaky anything about the data, or without negotiating the data owner's privacy. Similarly, cloud users possibly will not embrace the strong belief that the cloud server is undertaking a good job in terms of confidentiality. A cryptographic solution, with demonstrated security trusted on number-theoretic expectations is more desirable, when the

* Assistant Professor, Department of Computer Science and Engineering, Vel Tech High Tech Dr.Rangarajan Dr.Sakunthala Engineering College, Avadi, Chennai-62, Tamilnadu, India. *danu@velhightech.com, gowthamim@velhightech.com, senthil@velhightech.com, rajasaranya@velhightech.com*

user is not seamlessly happy with believing the security of the VM or the trustworthiness of the technical operator. These users are encouraged to encrypt their data with their individual keys formerly uploading them to the server.

Data sharing is a significant functionality in cloud storage. For example, bloggers can let their groups view a subset of their isolated pictures; an enterprise may grant her employees admittance to a portion of. The inspiring problem is how to successfully share encrypted data. Of course users can take the encrypted data from the storage, decrypt them, then direct them to others for distribution, but it misses the value of cloud storage. Users must be able to give the access rights of the distribution data to others so that they can access these data from the server straightly. However, finding a well-organized and secure way to share incomplete data in cloud storage is not trivial. Beneath we will take Dropbox1 as a sample for illustration.

Consider that Alice places all her isolated photos on Dropbox, and she does not need to expose her photos to everybody. Owing to numerous data leakage option Alice cannot feel pleased by just trusting on the confidentiality protection techniques provided by Dropbox, so she encrypts all the photos spending her own keys earlier uploading. One day, Alice's colleague, Bob, asks her to distribute the photos took over all these years which Bob looked in.

1. **https://www.dropbox.org :** Alice can then make use of the share function Dropbox, but the tricky now is how to give the *decryption rights* for these photographs to Bob. A potential option Alice can select is to strongly send Bob the secret keys involved. Certainly, there are two exciting ways for her below the traditional encryption standard:
   - Alice encrypts all files with a particular encryption key and provides Bob the corresponding secret key unswervingly.
   - Alice encrypts files with separate keys and sends Bob the parallel secret keys.

Clearly, the first method is insufficient since all unchosen data also leaked to Bob. For the second method, there are real-world concerns on efficiency. The number of such keys is as numerous as the number of the collective photos, say, a thousand. Transporting these secret keys fundamentally needs a secure channel, and loading these keys requires somewhat luxurious secure storage. The costs and complications involved usually rise with the amount of the decryption keys to be shared. In short, it is very weighty and costly to organize that.

Encryption keys also emanate with two flavors symmetric key or asymmetric (public) key. By means of symmetric encryption, when Alice needs the data to be created from a third party, she has to offer encrypt or her secret key; clearly, this is not always needed. By contrast, the encryption key and decryption key are dissimilar in public-key encryption. The use of public-key encryption provides more flexibility for our solicitations. For instance, in initiative settings, every operative can upload encrypted data on the cloud storing server without the information of the corporation's master-secret key.

So, the top solution for the overhead problem is that Alice encrypts files with different public-keys, but only shows Bob a single (constant-size) decryption key. Subsequently the decryption key must be sent via a safe channel and retained secret, small key size is constantly desirable. For instance, we cannot think large storage for decryption keys in the resource-constraint expedients like smart cards, smart phones or wireless sensor nodes. Particularly, these secret keys are regularly stored in the tamper-proof memory, which is moderately expensive. The current research efforts mostly focus on falling the communication necessities (such as bandwidth, circles of communication) like comprehensive signature .Though, not considerable has been complete around the key itself.

1.1. Our Contributions

In current cryptography, an ultimate problem we frequently study is about leveraging the confidentiality of a slight piece of knowledge into the capability to perform cryptographic roles (*e.g.* encryption, authentication) numerous times. In this paper, we learn how to make a decryption key further powerful in the sense that it permits decryption of numerous cipher texts, without collective its size. Exactly, our difficult statement is

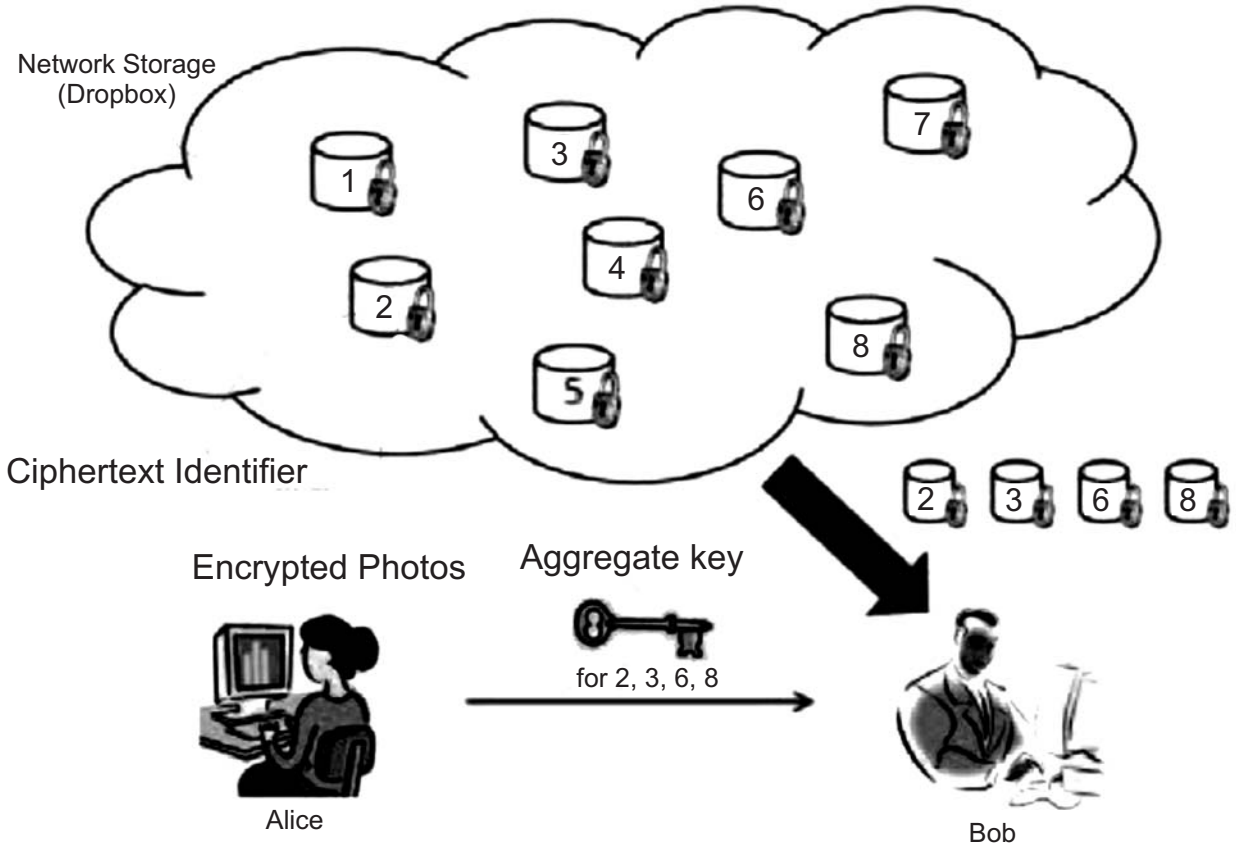


Figure 1: Alice distributes files with identifiers 2, 3, 6 and 8 with Bob by exchange him a single aggregate key

"To design a capable public-key encryption scheme which supports flexible allocation in the sense that any subgroup of the cipher texts (produced by the encryption scheme) is decrypt proficient by a fixed-size decryption key".

We accept this problem by giving an extraordinary type of public-key encryption which we name key-aggregate cryptosystem (KAC). Popular KAC, users encrypt a text not only below a public-key, but also below an identifier of cipher message called *class*. That means that cipher texts are further characterized into distinct classes. The key owner embraces a master-secret named *master-secret key*, which be able to be used to extract secret keys for dissimilar classes. More essentially, the extracted key take can be an aggregate key which is as compressed as a secret key for a sole class, but collections the power of numerous such keys, *i.e.*, the decryption power for some subsection of cipher text classes.

With our explanation, Alice can basically send Bob a single *aggregate key* through a secure send. Bob can transfer the encrypted photographs from Alice's Dropbox galaxy and before use this aggregate key to decrypt these encrypted photographs. The situation is depicted in Figure 1.

The sizes of public-key, cipher text, master-secret key and aggregate key in our KAC systems are all of fixed size. The public system limitation has dimension linear in the amount of cipher text classes, but simply a small part of it is wanted each time and it can be made on demand from enormous (but non-confidential) cloud storage.

Earlier results may attain a similar property containing a fixed-size decryption key, but the classes want to conform to certain pre-defined categorized relationship. Our work is stretchy in the sense that this limit is eliminated, that is, no different relation is required among the classes.

We recommend several real KAC schemes with various security levels and extensions in this paper. Constructions can be recognized secure in the standard model. To the best of our understanding, our aggregation Mechanism2 in KAC has not been examined.

2. KEY-AGGREGATE ENCRYPTION

We first give the outline and explanation for key aggregate encryption. Then we define how to use KAC in a situation of its application in cloud storage.

2.1. Framework

A key-aggregate encryption system consists of five polynomial-time algorithms. This algorithm follows.

1. The data proprietor begins the public system parameter through Setup and yields a public/master-secret3 key pair through Key Gen. Messages can be encrypted through Encrypt by anybody who also chooses what cipher text class is related with the plaintext communication to be encrypted. The data proprietor can use the master-secret to make a collective decryption key for a set of cipher text classes through Extract. The produced keys can be accepted to agents securely (through protected e-mails or protected devices) lastly, any user with a collective key can decrypt some cipher text delivered that the cipher text's class is controlled in the aggregate key through Decrypt4.
   - **Setup(1λ, *n*) :** implemented by the data proprietor to format an account on an *untrusted* server. Arranged input a security level bound 1λ and the amount of cipher text classes *n* (*i.e.*, class index would be an integer restricted by 1 and *n*), it outputs the public scheme parameter param, which is mislaid from the feedback of the extra algorithms for shortness.
   - **KeyGen :** Implemented by the data proprietor to arbitrarily make a public/master-secret key couple (*pk, msk*).
   - **Encrypt(*pk, i,m*) :** Performed by anybody who needs to encrypt data. On input as a public-key *pk*, an catalogue I meaning the cipher text class, and a communication *m*, it outputs a cipher text C.
   - **Extract(*msk,* S) :** Performed by the data proprietor for giving the decrypting control for a positive set of cipher text classes to a delegate. On involvement the master secret key msk and a set S of guide's equivalent to different classes, it outputs the collective key for set *S* denoted by KS.
   - **Decrypt(KS, S, *i,* C) :** Performed by a representative who established an aggregate key KS made by Extract. On input KS, the set S, an index *i* meaning the cipher text class the cipher text C goes to, and C, it outputs the decrypted outcome *m* if $i \in S$.
2. It is clear that we are *not* suggesting an algorithm to compress the decryption key. On one indicator, cryptographic keys originate from a high-entropy basis and are scarcely compressible. On the other hand, decryption keys for entirely possible groupings of cipher text classes are all in fixed-size — information hypothetically speaking such compression system cannot exist.
3. We demand this as master-secret key to elude misperception with the deputized key we will clarify later.
4. For easiness, we omit the presence of a decryption algorithm for the novel data proprietor using the master-secret key. In our precise constructions, we resolve show how the information of the master-secret key permits a faster decryption than using Extract charted by Decrypt.

There are two well-designed requirements :

- **Accuracy :** For every integers λ and *n*, every set $S \subseteq \{1, \cdots, n\}$, every index $i \in S$ and every message *m*, Pr[Decrypt (KS, S, *i*, C) = *m* : param ← Setup(1λ*, n*),(*pk, msk*) ← KeyGen(), C ← Encrypt(*pk, i,m*), KS ← Extract(msk, S)] = 1.
- **Compression :** For every integers λ, *n*, every set S, every index $i \in S$ and every message *m*; param ← Setup (1λ*, n*), (*pk, msk*) ← KeyGen(), KS ← Extract(*msk,* S) and C ← Encrypt(*pk, i,m*); |KS| and |C| first depend on the security parameter λ but self-determining of the numeral of classes *n*.

2.2. Sharing Encrypted Data

An unquestioned application of KAC is data distribution. The key aggregation property is particularly useful when we imagine the allocation to be efficient and flexible. The systems enable a content provider to part her data in a personal and careful way, with a secure and small cipher text extension, by allocating to each approved user a single and small aggregate key.

Here we define the main knowledge of data distribution in cloud storage by KAC, demonstrated in Figure 2. Assume Alice needs to share her data $m1, m2, \ldots, mv$ on the server. She first completes Setup($1\lambda, n$) to become param and execute KeyGen to grow the public/master-secret key pair(pk, msk). The system parameter param and public-key pk can be finished public and master-secret key msk should be reserved secret by Alice. Anybody (with Alice herself) can then encrypt all mi by Ci = Encrypt(pk, i,mi). The encrypted data are uploaded to the server.

By param and pk, people who collaborate with Alice can update Alice's data on the server. Once Alice is keen to share a set S of her records with a colleague Bob, she can compute the aggregate key KS for Bob by execution Extract(msk, S). Since KS is just a persistent size key, it is easy to be sent to Bob through a protected e-mail.

After obtaining the aggregate key, Bob can transfer the data he is certified to access. That is, for every $i \in$ S, Bob transfers Ci(and some needed values in param)from the server. With the collective key KS, Bob can decrypt each Ci by Decrypt(KS, S, i, Ci) for each $i \in$ S.

3. RELATED WORK

This section we relate our basic KAC system with additional possible solutions on distribution in secure cloud storage.

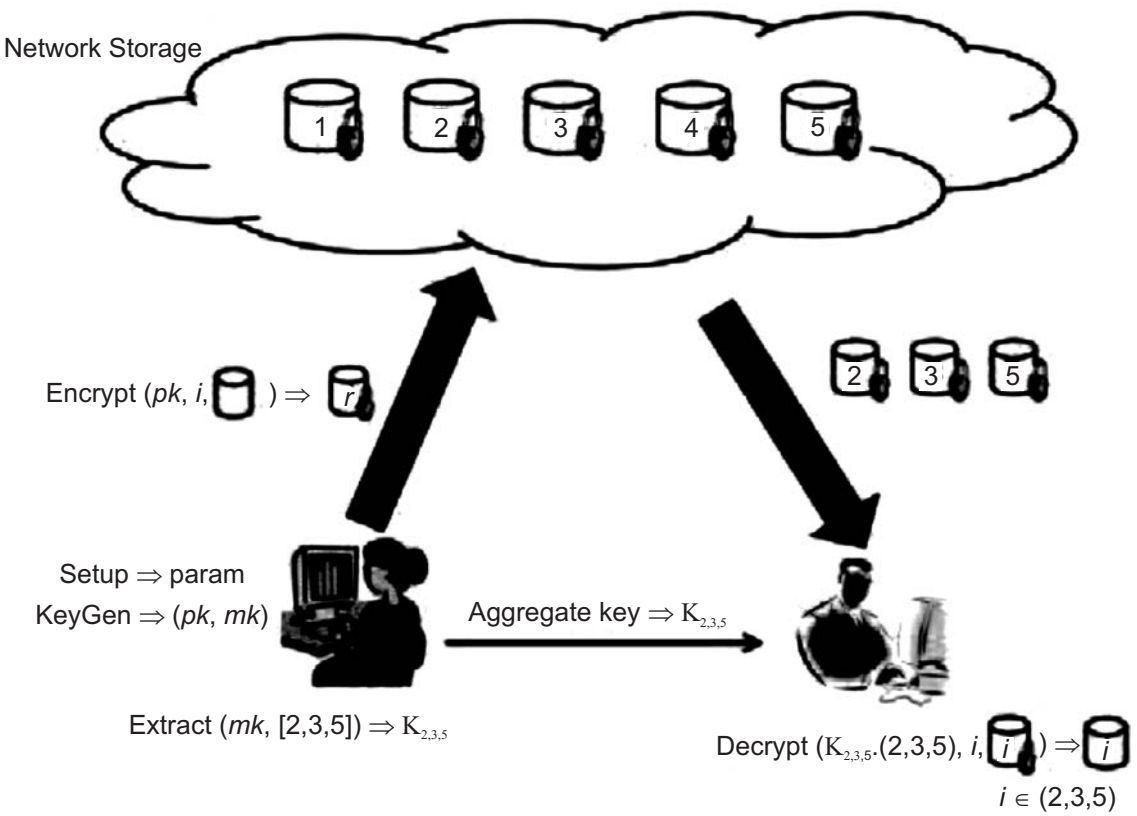


Figure 2: Using KAC for data sharing in cloud storage

3.1. Cryptographic Keys for a Predefined Hierarchy

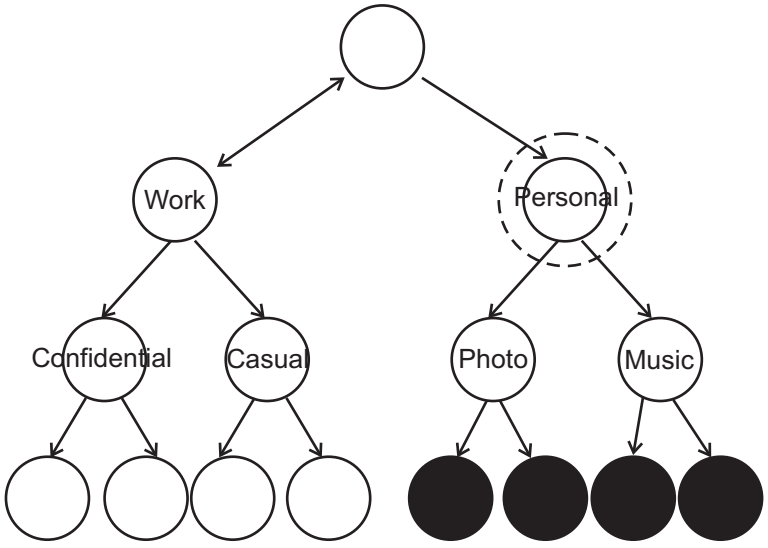
We start by conversing the greatest relevant study in the literature of cryptography/security. Cryptographic key assignment systems aim to reduce the payment in storing and handling secret keys for universal cryptographic use. Applying a tree structure, a key for a particular branch can be used to develop the keys of its successor nodes (but not the other round). Just permitting the parent key indirectly grants all the keys of its successor nodes. Sandhu planned a method to produce a tree hierarchy of symmetric keys by using frequent evaluations of pseudorandom function/block-cipher on a static secret key. The idea can be universal from a tree to a graph. Further innovative cryptographic key assignment systems support access policy that can be demonstrated by an acyclic graph or a cyclic graph. Most of these systems yield keys for *symmetric-key* cryptosystems.

Even though the key origins may want modular arithmetic to be used in public-key cryptosystems, which are typically more costly than "symmetric-key actions" like pseudo random function. We yield the tree arrangement as an example. Alice can first organize the cipher text classes allowing to their subjects comparable Figure 3. Each node in a tree denotes a secret key, whereas the leaf nodes denotes the keys for specific cipher text classes. Occupied circles denote the keys for the classes to be given and circles bypassed by dotted lines denote the keys to be granted. Reminder that every key of the non-leaf node can arise the keys of its successor nodes.

In Figure 3(*a*), if Alice desires to share totally the files in the "personal" taxonomy, she only wants to grant the key planned for the node "personal", which mechanically awards to give the keys of all the successor nodes ("photograph", "music"). This is the perfect case, where greatest classes to be shared fit in to the same branch and consequently a parent key of them is adequate.

However, it is still challenging for overall cases. As shown in Figure 3(*b*), if Alice shares her sample music work ("effort"→"unexpected"→ "sample"and"effort"→"confidential"→"sample") with a partner who also has the privileges to perceive some of her personal data, what she be able to do is to give extra keys, which leads to an rise in the total key size. One can get that this method is not stretchy when the organizations are more difficult and she needs to share diverse sets of files to diverse people. For this delgate in our specimen, the amount of granted secret keys develops the equal as the number of classes.

In universal, hierarchical methods can solve the problematic partially if one means to share all files below an assured branch in the hierarchy. On normal, the numeral of keys rises with the number of branches. It is questionable to come up through a hierarchy that can accept the number of total keys to be approved for all individuals (which can contact a diverse set of leaf-nodes) instantaneously.



(*a*)

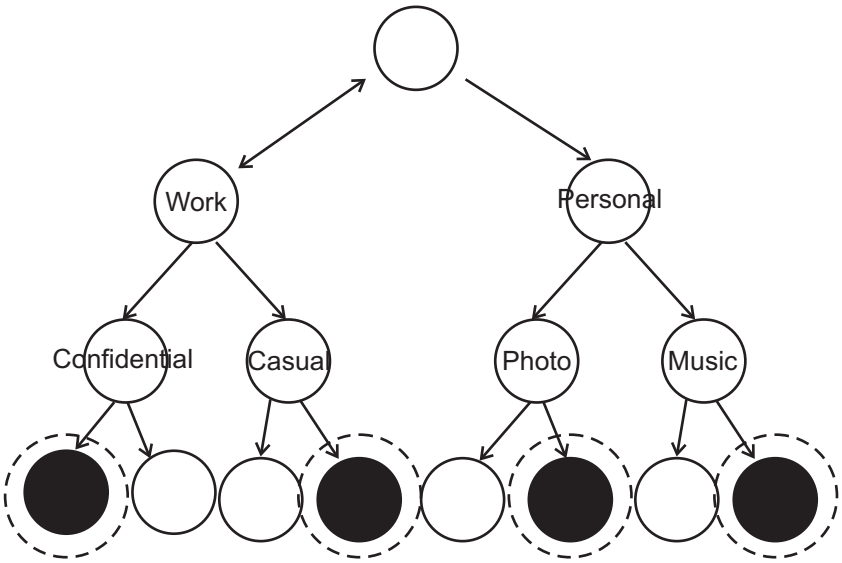


Figure 3: Compact key is not constantly possible for a stable Hierarchy

3.2. Compact Key in Symmetric-Key Encryption

Motivated by the similar problem of supporting flexible hierarchy in decryption power allocation (but in symmetric-key setting), Benaloh*et al.* presented an encryption scheme which is first proposed for briefly transmitting large number of keys in broadcast situation. The building is simple and we concisely review its key derivation procedure here for an actual description of what are the wanted properties we want to attain. The derivation of the key for a usual of classes (which is a subset of all possible cipher text classes) is as follows. A composite modulus $N = p \cdot q$ is selected where p and q are two huge random primes. A master secret key Y is selected at random from Z^*_N. Each class is connected with a separate prime ei. All these prime numbers can be placed in the public system parameter.

A fixed-size key for set S_ can be created (with the acquaintance of φ(N)) as kS_ = Y1/_ $j \in$ S_ (ej) mod cN. For those who consume been delegated the entrée rights for S where S_ ⊂ S, kS_ can be calculated by k_(∈S\S_ (ej)S. As a real example, a key for classes denoted by e1, e2, e3 can be produced as Y1/(e1·e2·e3), from which each of Y1/e1 , Y1/e2 ,Y1/e3 can simply be derived (while providing no information nearby keys for any other class, say, e4). This method achieves related properties and performances as our systems. However, it is designed for the symmetric-key location instead.

3.3. Compact Key in Identity-Based Encryption

Identity-based encryption (IBE) is a kind of public-key encryption in which the public-key of a user be able to set as an identity-string of the user (*e.g.*, an email address). There is a trustworthy party called private key generator (PKG) in IBE which clamps a master-secret key and disputes a secret key to every user with respect to the user identity. The encrypt or can take the public parameter and a user individuality to encrypt a message. The recipient can decrypt this cipher text by his secret key. Guo*et al.* tried to build IBE with key aggregation. One of their systems assumes random oracles but another does not. In their systems, key combination is controlled in the sense that all keys to be collected must come from altered "identity divisions". Though there are an exponential number of individualities and thus secret keys, only a polynomial number of them can be combined.

3.4. Other Encryption Schemes

Attribute-based encryption (ABE) permits each cipher text to be associated with an attribute, and the master-secret key holder can extract a secret key for a policy of these attributes. So that a cipher text can be decrypted through this key if its related attribute conforms to the policy. For instance, with the secret key for the policy (2 ∨3 ∨6 ∨8), one can decrypt cipher text labelled with class 2, 3, 6 or 8. Nevertheless, the main concern in ABE is collusion-resistance then not the compactness of secret keys. Certainly, the

size of the key frequently increases linearly with the numeral of attributes it encompasses, or the cipher text-size is not continuous.

To give the decryption power of certain cipher texts deprived of sending the secret key to the delegate, a useful basic is proxy re-encryption (PRE). A PRE system permits Alice to delegate to the server (proxy) the capability to translate the cipher texts encrypted below her public-key into ones for Bob.

4. NEW PATIENT-CONTROLLED ENCRYPTION

Moved by the nationwide effort to automate America's medical records, the knowledge of patient controlled encryption (PCE) has been planned. In PCE, the health record is disintegrated into a hierarchical demonstrator based on the use of dissimilar ontologies, and patients are the parties who produce and pile secret keys. When near is a need for a healthcare people to contact part of the record, a patient will issue the secret key for the concerned portion of the record.

Our work delivers a candidate solution for the lost piece, public-key PCE for flexible hierarchy, which the presence of an effectual construction was an open question. Every patient can moreover define her own hierarchy permitting to her need, or follow the set of categories advised by the electronic medical record scheme shies spending, such as "clinic visits", "*x*-rays", "allergies", "medications" and so on. Once the patient wishes to provide access rights to her doctor, she can able to choose any subset of these groups and concern a single key, from which keys for all these categories can be calculated.

Thus, we can basically use several hierarchy we choose, which is particularly useful when the hierarchy can be difficult. Lastly, one healthcare personnel deals with numerous patients and the patient record is conceivable stored in cloud storage owing to its huge size (*e.g.*, high resolution medical imaging engaging *x*-ray), compact key size and informal key management are of paramount significance.

5. CONCLUSION AND FUTURE WORK

To safeguard users' data privacy is a central problem of cloud storage. With more mathematical tools, cryptographic systems are getting more adaptable and often include multiple keys for a sole application. In this paper, we consider how to "compress" secret keys happening in public-key cryptosystems which care delegation of secret keys for diverse cipher text classes in cloud storage. No matter which one between the power set of classes, the delegate can continuously get an aggregate key of fixed size.

In cloud storage, the amount of cipher texts usually develops rapidly. So we have to reserve sufficient cipher text classes for the forthcoming extension.

Though the parameter can be transferred with cipher texts, it would be superior if its size is independent of the maximum number of cipher text classes in Multiple-key Leakage. In all leakage-resilient IBE systems, including the ones offered in this paper, leakage is permitted from only one secret key per identity. In these cases changed secret keys have to be produced for the similar identities, and as a result it is more challenging to apply leakage resilient systems in Master Secret Key Leakage. As we saw no leakage is permitted from the master secret key of our systems.

6. REFERENCES

1. 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, 2014.
2. B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Dataon the Cloud via Security-Mediator," in *International Conferenceon Distributed Computing Systems - ICDCS 2013*. IEEE, 2013.
3. L. Hardesty, "Secure computers aren't so secure," MITpress,2009,http://www.physorg.com/news176107396.html.
4. 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 – ACNS2012*, ser. LNCS, vol. 7341. Springer, 2012, pp. 526–543.
5. S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng,"Dynamic Secure Cloud Storage with Provenance," in *Cryptographyand Security: From Theory to Applications - Essays Dedicatedto 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, "Aggregateand Verifiably Encrypted Signatures from Bilinear Maps," in*Proceedings of Advances in Cryptology - EUROCRYPT '03*, ser. LNCS,vol. 2656. Springer, 2003, pp. 416–432.