Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

Compact Key based Client Control Mechanism for Cloud Environment

M. Pandi, C. Manikandan, Final Year BE (CSE)
Mr. S. Velliangiri, ME (Ph.D), Professor/CSE
Vidhya Mandhir Institute of Technology, Ingur, Erode, Tamilnadu, India

Abstract - Cloud environment is used to share data between the clients. Data owner maintains the shared data under cloud data centers. Semi-trusted third party servers are used to manage medical records. Personal Health Record (PHR) is used to manage patient personal and diagnosis details. PHR service allows a patient to create, manage and control their personal health data in one place through the web. Patient data can be shared with healthcare providers, family members and friends. Sensitive details are managed in medical records. Data owner decides the access privileges for the medical records.

The common key management mechanism supports constant cipher text with common decryption process for all users. One can aggregate any set of secret keys and make them as compact as a single key. The secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. The key aggregation system is divided into five major steps. They are Setup, KeyGen, Encrypt, Extract and Decrypt. The setup process is designed to creating an account on an untrusted server by the data owner. The keygen process is executed by the data owner to randomly generate a public/master-secret key pair. Encrypt process is executed by anyone using public key and index value. Extract is carried out by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate. Decrypt is executed by a delegate who received an aggregate key KS generated by Extract. Patient controlled encryption scheme is designed using Key Aggregate cryptosystem (KAC).

The cloud security system is improved with unlimited user access levels. The system is improved with device independent key distribution mechanism. The key distribution process is enhanced with security features to protect key leakage. The key parameter transmission process is integrated with the ciphertext download process.

I. INTRODUCTION

Cloud computing is a promising computing paradigm which recently has drawn extensive attention from both academia and industry. By combining a set of existing and new techniques from research areas such as Service-Oriented Architectures (SOA) and virtualization, cloud computing is regarded as such a computing paradigm in which resources in the computing infrastructure are provided as services over the Internet. Along with this new paradigm, various business models are developed, which can be described by terminology of "X as a service (XaaS)" where X could be software, hardware, data storage and etc. Successful examples are Amazon's EC2 and S3, Google App Engine and Microsoft Azure which provide users with scalable resources in the pay-as-you use fashion at relatively low prices. For example, Amazon's S3 data storage service just charges $0.12 to $0.15 per giga byte month. As compared to building their own infrastructures, users are able to save their investments significantly by migrating businesses into the cloud. With the increasing development of cloud computing technologies, it is not hard to imagine that in the near future more and more businesses will be moved into the cloud.

As promising as it is, cloud computing is also facing many challenges that, if not well resolved, may impede its fast growth. Data security, as it exists in many other applications, is among these challenges that would raise great concerns from users when they store sensitive information on cloud servers. These concerns originate from the fact that cloud servers are usually operated by commercial providers which are very likely to be outside of the trusted domain of the users. Data confidential against cloud servers is hence frequently desired when users outsource data for storage in the cloud. In some practical application systems, data confidentiality is not only a security/privacy issue, but also of juristic concerns. For example, in healthcare application scenarios use and disclosure of protected health information (PHI) should meet the requirements of Health Insurance Portability and Accountability Act (HIPAA) and keeping user data confidential against the storage servers is not just an option, but a requirement.

There are also cases in which cloud users themselves are content providers. They publish data on cloud servers for sharing and need fine-grained data access control in terms of which user has the access privilege to which types of data. In the healthcare case, for example, a medical center would be the data owner who stores millions of healthcare records in the cloud. It would allow data consumers such as doctors, patients, researchers and etc, to access various types of healthcare records under policies admitted by HIPAA. To enforce these access policies, the data owners on one hand would like to take advantage of the abundant resources that the cloud provides for efficiency and economy; on the other hand, they may want to keep the data contents confidential against cloud servers.

II. RELATED WORK

This paper is mostly related to works in cryptographically enforced access control for outsourced data and attribute based encryption. To realize fine-grained access control, the traditional public key encryption (PKE) based schemes, [10] either incur high key management overhead, or require encrypting multiple copies of a file using different users' keys. To improve upon the scalability of the above solutions, one-to-many encryption methods such as ABE can be used. In Goyal et. al's seminal paper on ABE, data is encrypted under a set of attributes so that multiple users who possess proper keys can decrypt. This potentially makes encryption and key management more efficient [2]. A fundamental property of ABE is preventing against user collusion. In addition, the encryptor is not required to know the ACL.

II.I. ABE for Data Access Control

A number of works used ABE to realize fine-grained access control for outsourced data. Especially, there has been an increasing interest in applying ABE to secure electronic healthcare records (EHRs). Narayan et al. proposed an attribute-based infrastructure for EHR systems, where each patient's EHR files are encrypted using a broadcast variant of CP-ABE [6] that allows direct revocation. The ciphertext length grows linearly with the number of unrevoked users. In [7], a variant of ABE that allows delegation of access rights is proposed for encrypted EHRs. Ibraimi et.al. applied ciphertext policy ABE (CP-ABE) to manage the sharing of PHRs and introduced the concept of social/professional domains. Akinyele et al. investigated using ABE to generate self-protecting EMRs, which can either be stored on cloud servers or cellphones so that EMR could be accessed when the health provider is offline.

There are several common drawbacks of the above works. First, they usually assume the use of a single trusted authority (TA) in the system. This not only may create a load bottleneck, but also suffers from the key escrow problem since the TA can access all the encrypted files, opening the door for potential privacy exposure. In addition, it is not practical to delegate all attribute management tasks to one TA, including certifying all users' attributes or roles and generating secret keys. In fact, different organizations usually form their own domains and become suitable authorities to define and certify

different sets of attributes belonging to their domains. For example, a professional association would be responsible for certifying medical specialties, while a regional health provider would certify the job ranks of its staffs. Second, there still lacks an efficient and on-demand user revocation mechanism for ABE with the support for dynamic policy updates/changes, which are essential parts of secure PHR sharing. Finally, most of the existing works do not differentiate between the personal and public domains, which have different attribute definitions, key management requirements and scalability issues. Our idea of conceptually dividing the system into two types of domains is similar with that; however a key difference is a single TA is still assumed to govern the whole professional domain.Yu et al. (YWRL) applied key-policy ABE to secure outsourced data in the cloud, where a single data owner can encrypt her data and share with multiple authorized users, by distributing keys to them that contain attribute-based access privileges. They also propose a method for the data owner to revoke a user efficiently by delegating the updates of affected ciphertexts and user secret keys to the cloud server. Since the key update operations can be aggregated over time, their scheme achieves low amortized overhead. It would be inefficient to be applied to a PHR system with multiple data owners and users, because then each user would receive many keys from multiple owners, even if the keys contain the same sets of attributes.

II.II Revocable ABE

It is a well-known challenging problem to revoke users/ attributes efficiently and on-demand in ABE. Traditionally this is often done by the authority broadcasting periodic key updates to unrevoked users frequently, which does not achieve complete backward/ forward security and is less efficient. Recently, proposed two CP-ABE schemes with immediate attribute revocation capability, instead of periodical revocation. They were not designed for MAABE. Ruj et al. [11] proposed an alternative solution for the same problem in our paper using Lewko and Waters's (LW) decentralized ABE scheme [4]. The main advantage of their solution is, each user can obtain secret keys from any subset of the TAs in the system, in contrast to the CC MA-ABE. In this paper, we bridge the above gaps by proposing a unified security framework for patient-centric sharing of PHRs in a multi-domain, multi-authority PHR system with many users. The framework captures application-level requirements of both public and personal use of a patient's PHRs and distributes users' trust to multiple authorities that better reflects reality. We also propose a suite of access control mechanisms by uniquely combining the technical strengths of both CC MA-ABE and the YWRL ABE scheme. Using our scheme, patients can choose and enforce their own access policy for each PHR file and can revoke a user without involving high overhead. We also implement part of our solution in a prototype PHR system.

III. PRIVACY OF ELECTRONIC MEDICAL RECORDS

Data privacy is a traditional way to ensure it is to rely on the server to enforce the access control after authentication (e.g., [1]), 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 [8]. 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 [9]. 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. On February 13, 2009, U.S. President Barack Obama signed into law the American Recovery and Reinvestment Act of 2009, which contained provisions authorizing the federal government to spend 19 billion dollars to digitize U.S. health records. President Obama stated, we will make the immediate investments necessary to ensure that within five years all of America's medical records are computerized," and that digital medical records could prevent medical errors, save lives and create hundreds of thousands of jobs. Moving to electronic health records is important to the modernization and revamping of our healthcare system, but solving the great challenges of ensuring the safety, security and privacy of patients is equally critical, of which the federal government is acutely aware.

Computerized medical records are open to potential abuses and threats. Some have pointed out that large amounts of sensitive healthcare information held in data centers are vulnerable to loss, leakage, or theft. In the last few years, personal health information on hundreds of thousands of people has been compromised because of security lapses at hospitals, insurance companies and government agencies." Medical data is also susceptible to misuse by those seeking to profit from it. Furthermore, disputes among lawmakers seeking to regulate the computerization of medical records and lobbyists from pharmaceutical companies and insurance companies may delay or derail effective privacy safeguards from being put in place to protect patients' rights and safety. Given the plan to widely deploy electronic medical records systems, challenges of interoperability and standardization come to the fore. In the absence of regulations which assure patient privacy, standards for interoperability may impose designs that limit that privacy. On the other hand, the focus of attention on potential standards and interoperability provides a rich context of developing systems that can protect privacy as data flows across multiple systems. Today, many organizations provide electronic medical records (EMR) solutions. The primary method of guaranteeing privacy in today's systems is access control. In a system which relies solely on access control, the servers that store data run an access control program, which verifies that any party accessing a patient's healthcare record has appropriate permissions. These access control systems keep a log of all accesses and communications are securely encrypted. Overall, this has been a fairly effective approach. Patients must trust the third party storing their data with their private health record. If a data center is compromised, patients' private data may be revealed. Similarly, such a storage design will be vulnerable to privacy and trust issues. Thus, we propose instead that each patient should generate her own decryption key and use that key to encrypt her records. Encryption schemes with strong security properties will guarantee that the patient's privacy is protected. Adherence to a simple encryption scheme can interfere with the desired functionality of health record systems. In particular, we would like to employ encryption, yet support such desirable functions as allowing users to share partial access rights with others and to perform various searches over their records. In what follows, we consider encryption schemes that enable patients to delegate partial decryption rights and that allow patients to search over their health data.

We shall propose a design that we refer to as Patient Controlled Encryption (PCE) as a solution to secure and private storage of patients' medical records. PCE allows the patient to selectively share records among doctors and healthcare providers. The design of the system is based on a hierarchical encryption system. The patient's record is partitioned into a hierarchical structure, each portion of which is encrypted with a corresponding key. The patient is required to store a root secret key, from which a tree of subkeys is derived. The patient can selectively distribute subkeys for decryption of various portions of the record. The patient can also generate and distribute trapdoors for selectively searching portions of the record. Our design prevents unauthorized access to patients' medical data by data storage providers, healthcare providers, pharmaceutical companies, insurance companies, or others who have not been given the appropriate decryption keys. Public-key cryptosystems produce constant-size ciphertexts with

efficient delegation of decryption rights for any set of ciphertexts. One can aggregate any set of secret keys and make them as compact as a single key. The secret key holder can release a constant-size aggregate key for flexible choices of ciphertext set in cloud storage. This compact aggregate key can be conveniently sent to others or be stored in a smart card with very limited secure storage. The key aggregation system is divided into five major steps. They are Setup, KeyGen, Encrypt, Extract and Decrypt. The setup process is designed to creating an account on an untrusted server by the data owner. The keygen process is executed by the data owner to randomly generate a public/master-secret key pair. Encrypt process is executed by anyone using public key and index value. Extract is carried out by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate. Decrypt is executed by a delegate who received an aggregate key KS generated by Extract. Patient controlled encryption scheme is designed using Key Aggregate cryptosystem (KAC). The following problems are identified from the security scheme for cloud data services. Predefined boundary for the maximum number of secret keys, Limited size for ciphertext classes, trusted hardware is required for key distribution and Key leakage possibility is high.

IV. SECURITY SOLUTIONS TO PROTECT PATIENT RECORDS

We first give the framework and definition for key aggregate encryption. Then we describe how to use KAC in a scenario of its application in cloud storage. A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows. The data owner establishes the public system parameter via Setup and generates a public/master-secret3 key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what ciphertext class is associated with the plaintext message to be encrypted. The data owner can use the master-secret to generate an aggregate decryption key for a set of ciphertext classes via Extract. The generated keys can be passed to delegates securely finally, any user with an aggregate key can decrypt any ciphertext provided that the ciphertext's class is contained in the aggregate key via Decrypt.

- Setup(1,n): executed by the data owner to setup an account on an untrusted server. On input a security level parameter 1 and the number of ciphertext classes n, it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.
- KeyGen: executed by the data owner to randomly generate a public/master-secret key pair (p_k ,ms_k).
- Encrypt(p_k I,m): executed by anyone who wants to encrypt data. On input a public-key p_k , an index I denoting the ciphertext class and a message m, it outputs a ciphertext C.
- Extract(msk, S): executed by the data owner for delegating 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 aggregate key for set S denoted by KS.
- Decrypt(KS, S, i) C: executed by a delegatee who received an aggregate key KS generated by Extract. On input KS, the set S , an index i denoting the ciphertext class the ciphertext C belongs to and C, it outputs the decrypted result m if $i \in S$.

IV.I. RSA Algorithm

The domain name service sensitive attributes are secured using the RSA algorithm. The Rivert, Shamir, Adelman (RSA) scheme is a block cipher in which the Plaintext and cipher text are integers between 0 and n-1 for some n. A typical size for n is 1024 bits or 309 decimal digits.

Key Generation

Select p,q	p and q both prime , p≠q
Calculate n = p x q	
Calculate $\phi(n)=(p-1)(q-1)$	
Select integer e	$gcd(\phi(n),e) = 1; 1 < e < \phi(n)$
Calculate d	$d = e^{-1} \bmod \phi(n)$
Public key	KU = {e, n}
Private key	KR = {d, n}

Encryption

Plaintext	M <n
Cipher text	$C = M^e \pmod n$

Decryption

Cipher text	C
Plaintext	$M = C^d \pmod n$

IV.II. Secure Hashing Algorithm

The Secure Hash Algorithm is a family of cryptographic hash functions published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS) and may refer to:-

A 160-bit hash function which resembles the earlier MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm. Cryptographic weaknesses were discovered in SHA-1 and the standard was no longer approved for most cryptographic uses after 2010.

V. SECURED MULTI USER DATA SHARING IN CLOUDS

The system is designed to share data with access control based security model described in Fig. 5.1. Individual and role based access policy model is used in the system. Data integrity verification process is integrated with the system. The system is divided into six major modules. They are Cloud Data Center, Data Owner, Key Management, Policy Controller, Data Security Process and Client. The cloud data center manages the data owner and user accounts. Data owner maintains the shared data and user access privileges. Key generation and distribution operations are handled in key management module. User access permissions are managed under the policy controller module. Data security module is used to protect the shared data uploading process. The client module is used to access the shared data values provided by the data owner.

V.I. Cloud Data Center

Cloud data center maintains the data owner and client details. Cloud data center allocates storage space for the data owners. Data center manages the shared data uploaded by the data owner. Client requests are processed by the data center.

V.II. Data Owner

Data owner uploads their data in to the cloud data center. Data owner creates different user account to access their data values. User accounts are managed with access level based groups. Policy and key management operations are carried out by the data owner.

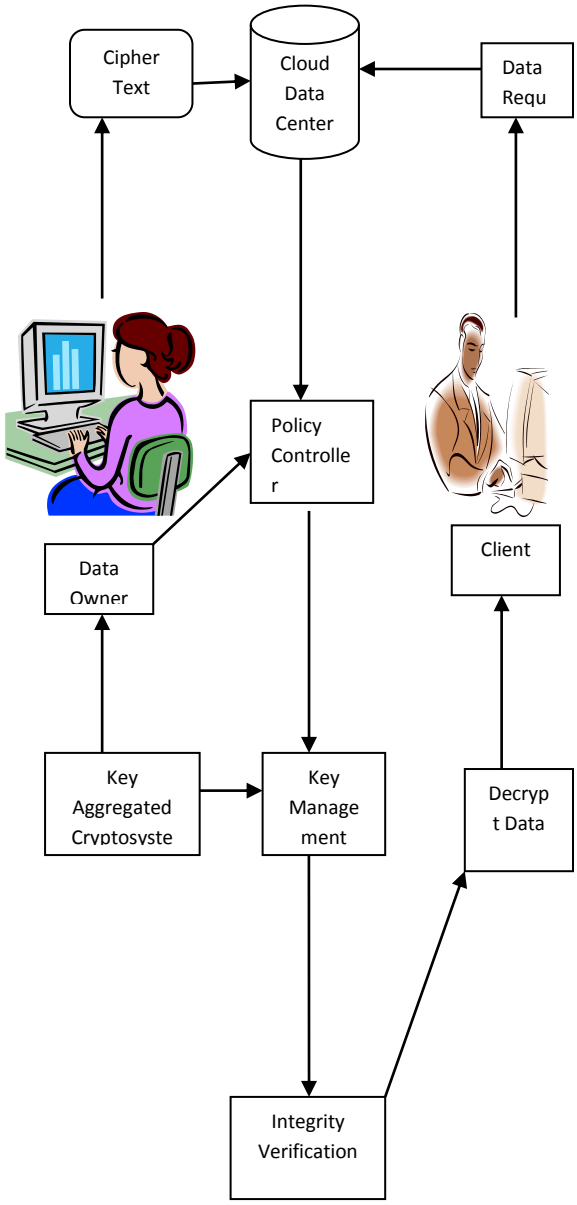

Fig. No: 5.1. Secured Multi User Data Sharing in Clouds

V.III. Key Management

Key generation and distribution tasks are handled in the key management process. Master key and multiple secret keys are generated by the data owner. Master key is used for the encryption process and the secret keys are used for the decryption process. Compact key is constructed with the aggregated secret key values.

V.IV. Policy Controller

User access permissions are maintained in the policy controller. User level and group level access rights are issued by the data owner. Shared data values are provided with reference to the user access rights. Secret keys are assigned for the users with reference to their access levels.

V.V. Data Security Process

The data security process protects the uploaded data. Data owner encrypts and upload the data to the data center. Multiple cipher text values are maintained for each shared data item. Data integrity verification is performed in the data upload and downloads process.

V.VI. Client

Client application is designed to access the shared data provided by the data owner. Encrypted data is downloaded from the data center. Decryption process is carried out using the secret key associated with the client access levels. The client data access details are reported to the data owner.

VI. CONCLUSION

The cloud data storages are used to share data from different data owners. Key Aggregated Cryptosystem (KAC) is used to share data with multiple users with security. The KAC scheme is enhanced with multiple ciphertexts classes. The system is improved with secured key distribution scheme to protect key leakages. Patient-controlled encryption scheme is applied for flexible hierarchy. Policy based security model is used to provide data security with access control mechanism. Distributed storage is supported with data encryption and access control mechanism. Device independent key distribution model is used in the system. Secured key distribution process is adopted to protect the key values.

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, 2012.
[2] M. Li, W. Lou and K. Ren, "Data security and privacy in wireless body area networks," *IEEE Wireless Communications Magazine*, Feb. 2010.
[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, Feb. 2013.
[4] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," *Advances in Cryptology–EUROCRYPT*, pp. 568–588, 2011.
[5] 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.
[6] S. Narayan, M. Gagn´e and R. Safavi-Naini, "Privacy preserving ehr system using attribute-based infrastructure," ser. CCSW '10, 2010, pp. 47–52.
[7] X. Liang, R. Lu, X. Lin and X. S. Shen, "Patient self-controllable access policy on phi in ehealthcare systems," in *AHIC 2010*, 2010.
[8] L. Hardesty, Secure Computers Aren't so Secure. MIT press, http:// www.physorg.com/news176107396.html, 2009.
[9] 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, 2013.
[10] C. Dong, G. Russello and N. Dulay, "Shared and searchable encrypted data for untrusted servers," in *Journal of Computer Security*, 2010.
[11] S. Ruj, A. Nayak and I. Stojmenovic, "Dacc: Distributed access control in clouds," in *10th IEEE TrustCom*, 2011.

