DYNAMIC ACCESS CONTROL THROUGH CRYPTOGRAPHY IN CLOUD

Submitted in partial fulfillment of the requirements for the award

of

Bachelor of Engineering degree in Computer Science and Engineering

By

SUBISHA M (Reg.No - 39110978)
PRIYANKA S S (Reg.No – 39110807)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SUBISHA M(Reg.No - 39110978) and PRIYANKA S S(Reg.No - 39110807)** who carried out the Project Phase-2 entitled **"DYNAMIC ACCESS CONTROL THROUGH CRYPTOGRAPHY IN CLOUD"** under my supervision from January 2023 to April 2023.

Internal Guide

Dr. M.SANKARI M.E., Ph.D

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 20.04.2023

Internal Examiner **External Examiner**

DECLARATION

I, **SUBISHA M(Reg.No- 39110978),** hereby declare that the Project Phase-2 Report entitled **"DYNAMIC ACCESS CONTROL THROUGH CRYPTOGRAPHY IN CLOUD"** done by me under the guidance of **Dr.M.SANKARI, M.E.,Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 14.04.2023

PLACE: Chennai **SIGNATURE OF THECANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D**, **Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.M.SANKARI M.E.,Ph.D,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks toall Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In recent days, encrypting the data in the untrusted cloud with access control is necessary for many users and organizations. However, it is a challenging thing to design a efficient access control. This paper proposes DAC in cryptography. It is a system that provides practical encryption for dynamic access control. It encrypts the users file with a symmetric key list that has a file key and set of lock keys. After every revocation, the administrator must upload a new revocation key to encrypt the file and update the encryption key list accordingly with new encryption level. As a result, dynamic access control is enforced using cryptography. This does not require expensive decryption/re-encryption or upload/reuploading of large amount of data on the part of administrator. It ensures security through revoking permissions immediately. Thereby it improves efficiency. DAC in cryptography proposes three key techniques to limit the size of key list and encryption layer. We use formalization framework and system implementation to demonstrate the security and efficiency of our construction.

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

DAC in cryptography encrypts files with a symmetric key list that records a file key and a set of lock keys. Each revocation requires a dedicated administrator to upload a new revocation key to the cloud, encrypt files with the new encryption level, and update the encryption key list accordingly. Tremendous advances in cloud computing, users and organizations are increasingly attracted to storing and sharing data via cloud services. Cloud service providers (Amazon, Microsoft, Apple, etc.) offer a multitude of cloud-based services, from small personal services to large industrial services. However, recent data breaches, such as the exposure of private photos, have raised concerns about the privacy of cloud-managed data.

In fact, cloud service providers are typically insecure due to software design flaws and system vulnerabilities. Therefore, how to enforce data access control in potentially untrusted clouds is a key issue. In response to these security concerns, number of works have been proposed to support controlling access to untrusted cloud services using cryptographic primitives. Sophisticated cryptographic primitives are applied to enforce many access control paradigms. For example, attribute-based encryption (ABE) is the cryptographic equivalent of the attribute-based access control (ABAC) model. However, previous work has primarily considered static scenarios where access control policies rarely change. If you need to change the access control policy, the previous work will incur high overhead.

Seemingly, you can revoke a user's permissions by depriving of access to the keys used to encrypt their files. However, this solution is not secure as the user can keep a local copy of the key before revoking it. To avoid such problems, the files should be re-encrypted with a new key. This requires the file's owner to download the file, re-encrypt the file, and upload it again so the cloud updates the previously encrypted file.

The first scheme requires the administrator to re-encrypt the file with a new key, as described above. This scheme involves significant communication overhead. Instead, the second scheme delegates the file to be re-encrypted when the user needs to change it, so that the administrator does not have to re-encrypt the file her data itself. However, this scheme has a security drawback as it delays the undo

process until the user next modifies the file. This gives the newly revoked user access to the file until the next write. Also proposed another locking scheme that encrypts files using a symmetric homomorphic encryption scheme. Such a design allows the cloud to directly re encrypt files without decrypting. Encryption/decryption operation has overhead comparable to public-key cryptography, so this scheme incurs expensive file read/write overhead. Overcome access control systems in untrusted clouds. DAC in cryptography delegates to the cloud to update files encrypted with permission revocation.

DAC in cryptography proposes three main techniques: First, DAC in cryptography, proposes a delegation-enabled encryption strategy that delegates policy data updates to the cloud. For files, the administrator adds the new lock key to the end of its key list and asks the cloud to update this key list in the policy data. However, the size of the key list increases with revocation operations, requiring users to download and decrypt large key lists each time they access a file. To avoid this issue, we've enabled administrators to define an allowed file limit. Once the encryption layer reaches its size limit, it cannot be increased by delegating encryption operations to the cloud. This gives administrators the flexibility to set permissible limits per file (depending on file type, access patterns, for balance. DAC in cryptography proposes a delayed deionization encryption strategy that periodically updates a file's symmetric key list and removes the limited encryption layer above it with write operations.

Specifically, the next user who writes to the file encrypts the write content with a new symmetric key list containing only the new file key and updates the key list in the policy data. With this strategy, DAC in cryptography removes the limited layer of encryption from files, making it easier for many writers. Overall, DAC in cryptography simultaneously achieves efficient locking, efficient file access, and immediate locking. Due to revocation efficiency, DAC in cryptography incurs very little communication overhead on the part of administrators, as file data does not need to be downloaded and re-uploaded. With immediate revoke, user permissions are immediately revoked when files are re-encrypted. Files are still encrypted with a symmetric key for efficient file access.

Ali cloud has implemented DAC in cryptography and some recent work. Practical experiments suggest that DAC in cryptography is three orders of magnitude more efficient in communicating access revocations and almost two orders of

magnitude more efficient in computing file accesses than the first scheme. Finally, compared to the second scheme, DAC in cryptography DAC can revoke permissions instantly. To solve these problems, we introduce DAC in cryptography.

With the considerable advancements in cloud computing, users and organizations are finding it increasingly appealing to store and share data through cloud services. Cloud service providers (such as Amazon, Microsoft, Apple, etc.) provide abundant cloud-based services, ranging from small-scale personal services to large-scale industrial services. However, recent data breaches, such as releases of private photos, have raised concerns regarding the privacy of cloud-managed data. A cloud service provider is usually not secure due to design draw backs of software and system vulnerability. As such, a critical issue is how to enforce data access control on the potentially untrusted cloud.

In response to these security issues, numerous works have been proposed to support access control on untrusted cloud services by leveraging cryptographic primitives. Advanced cryptographic primitives are applied for enforcing many access control paradigms. For example, attribute-based encryption (ABE) is a cryptographic counterpart of attribute-based access control (ABAC) model.

However, previous works mainly consider static scenarios in which access control policies rarely change. The previous works incur high overhead when access control policies need to be changed in practice. At a first glance, the revocation of a user's permission can be done by revoking his access to the keys with which the files are encrypted. This solution, however, is not secure as the user can keep a local copy of the keys before the revocation. To prevent such a problem, files have to be re-encrypted with new keys. This requires the file owner to download the file, re-encrypt the file, and upload it back for the cloud to update the previous encrypted file, incurring prohibitive communication overhead at the file owner side.

Currently, only a few works investigated the problem of dynamic data access control. Garrison et al. proposed two revocation schemes. The first scheme requires an administrator to re-encrypt file with new keys as dis cussed above. This scheme incurs a considerable communication overhead. Instead, the second scheme delegates users to re-encrypt the file when they need to modify the file, relieving the administrator from re-encrypting file data by itself. This scheme, however, comes with a security penalty as the revocation operation is delayed to the next user's modification to the file. As a result, a newly revoked user can still access the file

before the next writing operation. Wang et al. proposed another revocation scheme, in which the symmetric homomorphic encryption scheme is used to encrypt the file. Such a design enables the cloud to directly re-encrypt file without decryption. However, this scheme incurs expensive file read/write overhead as the encryption/decryption operation involves comparable overhead with the public key encryption schemes.

To overcome these problems, we present Crypt-DAC, a cryptographically enforced dynamic access control system on untrusted cloud. Crypt-DAC delegates the cloud to update encrypted files in permission revocations. In Crypt-DAC, a file is encrypted by a symmetric key which records a file key and a sequence of revocation keys. In a revocation, the administrator uploads a new revocation key to the cloud, which encrypts the file with a new layer of encryption and updates the encrypted key list accordingly. Same as previous works, we assume a honestbut-curious cloud, i.e., the cloud is honest to perform the required commends (such as re-encryption of files and properly update previous encrypted files) but is curious to passively gathering sensitive information. Although the basic idea of layered encryption is simple, it entails tremendous technical challenges. For instance, the size of key list and encryption layers would increase as the number of revocation operations, which incurs additional decryption overhead for users to access files.

To overcome such a problem, Crypt-DAC proposes three key techniques as follows. First, Crypt-DAC proposes delegation-aware encryption strategy to delegate the cloud to update policy data. For a file, the administrator appends a new revocation key at the end of its key list and requests the cloud to update this key list in the policy data. The size of the key list however increases with the revocation operations, and a user has to download and decrypt a large key list in each file access. To overcome this problem, we adopt the key rotation technique to compactly encrypt the key list in the policy data. As a result, the size of the key list remains constant regardless of revocation operations.

Second, Crypt-DAC proposes adjustable onion encryption strategy to delegate the cloud to update file data. For a file, the administrator requests the cloud to encrypt the file with a new layer of encryption. Similarly, the size of the encryption layers increases with the revocation operations, and a user has to decrypt multiple times in each file access. To overcome this problem, we enable the administrator to define a tolerable bound for the file. Once the size of encryption layers reaches the

bound, it can be made to not increase anymore by delegating encryption operations to the cloud. As a result, the administrator can flexibly adjust a tolerable bound for each file (according to file type, access pattern, etc.) to achieve a balance between efficiency and security.

During the life cycle of a file, its encryption layers continuously increase until a pre-defined bound is reached. Crypt-DAC proposes delayed de-onion encryption strategy to periodically refresh the symmetric key list of the file and remove the bounded encryption layers over it through writing operations. In specific, the next user to write to the file encrypts the writing content by a new symmetric key list only containing a new file key, and updates the key list in the policy data. With this strategy, Crypt-DAC periodically removes the bounded encryption layers of files while amortizing the burden to a large number of writing users. Altogether, Crypt-DAC achieves efficient revocation, efficient file access and immediate revocation simultaneously. For revocation efficiency, Crypt-DAC incurs lightweight communication overhead at the administrator side as it does not need to download and re-upload file data. For immediate revocation, the permissions of users are immediately revoked as the files are re-encrypted. For file access efficiency, the files are still encrypted by symmetric keys.We have implemented Crypt-DAC as well as several recent works on Ali-cloud.

Real experiments suggest that Crypt-DAC is three orders of magnitude more efficient in communication in access revocation compared with the first scheme in, and is nearly two orders of magnitude more efficient in computation in file access compared with the scheme. Finally, Crypt-DAC is able to immediately revoke access permissions compared with the second scheme. The remainder of this paper is organized as follows. The system model and assumptions, background on RBAC0 and the cryptographic techniques used in our system design has been introduced. Several critical issues for cryptographically enforced dynamic access control has been identified, from which the principles of Crypt-DAC have been derived. Also describes the design details of Crypt-DAC. The security of Crypt-DAC has been formally analyzed. The performance of Crypt-DAC is been compared through experiments.

CHAPTER 2

LITERATURE SURVEY

Vipul Goyal,Omkant Pandeyy and Amit Sahai[1] proposed Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data in 2006 that as more sensitive data is shared and stored by third-party sites on the Internet, there will be a need to encrypt data stored at these sites. One drawback of encrypting data is that it can be selectively shared only at a coarse-grained level (i.e., giving another party your private key). We develop a new cryptosystem for ne-grained sharing of encrypted data that we call Key-Policy Attribute-Based Encryption (KP-ABE). In our cryptosystem, cipher texts are labeled with sets of attributes and private keys are associated with access structures that control which cipher texts a user is able to decrypt. They demonstrate the applicability of our construction to sharing of audit-log information and broadcast encryption. The construction supports delegation of private keys which subsumes Hierarchical Identity-Based Encryption.

Vipul Goyal, Abhishek Jain and Omkant Pandey[2] proposed Bounded Cipher text Policy Attribute Based Encryption in 2009 that in a cipher text policy attribute based encryption system, a user's private key is associated with a set of attributes (describing the user) and an encrypted cipher text will specify an access policy over attributes. A user will be able to decrypt if and only if his attributes satisfy the cipher text's policy. In this work, they present the first construction of a cipher text-policy attribute-based encryption scheme having a security proof based on a number theoretic assumption and supporting advanced access structures. Previous CP-ABE systems could either support only very limited access structures or had a proof of security only in the generic group model. Our construction can support access structures which can be represented by a bounded size access tree with threshold gates as its nodes. The bound on the size of the access trees is chosen at the time of the system setup. Our security proof is based on the standard Decisional Bilinear Different Hellman assumption.

John Bethencourt and Amit Sahai[3] proposed Cipher text-Policy Attribute-Based Encryption in 2008 that in several distributed systems a user should only be able to access data if a user possess a certain set of credentials or attributes. Currently, the only method for store the data and mediate access control. However, if any server storing the data is compromised, then the confidentiality of the data will be compromised. In this paper they present a system for realizing complex access control on encrypted data that they call Cipher text-Policy Attribute-Based Encryption. By using our techniques encrypted data can be kept confidential even if the storage server is entrusted; moreover, these methods are secure against collusion attacks. Previous Attribute- Based Encryption systems used attributes to describe the encrypted data and built policies into user's keys; while in this system attributes are used to describe a user's credentials, and a party encrypting data deter- mines a policy for who can decrypt. Thus, these methods are conceptually closer to traditional access control methods such as Role-Based Access Control (RBAC). In addition, we provide an implementation of this system and give performance measurements.

Dr. Ragesh G. K.a and Dr. K. Baskaranb[4] proposed Cryptographically Enforced Data Access Control in Personal Health Record Systems in 2016 that personal Health Record (PHR) systems play a vital role during digital transformation of healthcare. These systems provide many value-added features like viewing one's health related information, secure transmission and tracking of that information with the health service providers. A cloud assisted PHR system maximizes the possibility for PHR systems to interoperate with other systems in health information management environments. Each patient needs to encrypt his/her PHR data before uploading it in the cloud since the patients will lose their physical access to their health data stored in cloud servers. Moreover, to achieve fine-grained data access control on encrypted PHR data in an effective and scalable manner is a challenging task. Since there are multiple owners or patients are available in a PHR system and existing data access control schemes are mostly designed for the single-authority/owner scenarios, a novel patient-centric data access control scheme called Revocable Multi Authority Attribute Set Based Encryption (R- MA- ASBE) is proposed. The proposed scheme inherits flexibility, scalability and fine-grained patient centric data access control.

Xiaoguang Wang and Yong Qi[5] proposed Design and Implementation of Sec Pod, A Framework for Virtualization-based Security Systems in 2017 that the OS kernel is critical to the security of a computer system. Many systems have been proposed to improve its security. A fundamental weakness of those systems is that page tables, the data structures that control the memory protection, are not isolated from the vulnerable kernel, and thus subject to tampering. This fundamentally conflicts with the recent advances in the hardware virtualization support. In this paper, they present the design and implementation of Sec Pod, a practical and extensible framework for virtualization-based security systems that can provide both strong isolation and the compatibility with modern hardware. Sec Pod has two key techniques: paging delegation delegates and audits the kernel's paging operations to a secure space; execution trapping intercepts the (compromised) kernel's attempts to subvert Sec Pod by misusing privileged instructions. They have implemented a prototype of Sec Pod based on KVM. Their experiments show that Sec Pod is both effective and efficient.

Mikhail J. Atallah, Keith B. Frikken, and Marina Blanton[6] proposed Efficient Key Management for Access Hierarchies in 2005 that the problem of key management in an access hierarchy has elicited much interest in the literature. The hierarchy is modeled as a set of partially ordered classes (represented as a directed graph), and a user who obtains access (i.e., a key) to a certain class can also obtain access to all descendant classes of her class through key derivation. Our solution to the above problem has the following properties: (i) only hash functions are used for a node to derive a descendant's key from its own key; (ii) the space complexity of the public information is the same as that of storing the hierarchy; (iii) the private information at a class consists of a single key associated with that class; (iv) updates (revocations, additions, etc.) are handled locally in the hierarchy; (v) the scheme is provably secure against collusion; and (vi) key derivation by a node of its descendant's key is bounded by the number of bit operations linear in the length of the path between the nodes. Whereas many previous schemes had some of these properties, their's is the first that satisfies all of them. Moreover, for trees (and other "recursively decomposable" hierarchies), we are the first to achieve a worst- and average-case

number of bit operations for key derivation that is exponentially better than the depth of a balanced hierarchy (double-exponentially better if the hierarchy is unbalanced, i.e., “tall and skinny”); this is achieved with only a constant increase in the space for the hierarchy. They also show how with simple modifications our scheme can handle extensions proposed by Crampton of the standard hierarchies to “limited depth” and reverse inheritance

Sascha M¨uller and Stefan Katzenbeisser[7] proposed Hiding the Policy in Cryptographic Access Control in 2011that recently, cryptographic access control has received a lot of attention, mainly due to the availability of efficient Attribute-Based Encryption (ABE) schemes. ABE allows to get rid of a trusted reference monitor by enforcing access rules in a cryptographic way. However, ABE has a privacy problem: The access policies are sent in clear along with the cipher texts. Further generalizing the idea of policy-hiding in cryptographic access control,they introduce policy anonymity where similar to the well-understood concept of k-anonymity the attacker can only see a large set of possible policies that might have been used to encrypt, but is not able to identify the one that was actually used. They show that using a concept from graph theory we can extend a known ABE construction to achieve the desired privacy property.

Keywords: access control, privacy, tree majors, abe, anonymity, hidden policies

Xiaofeng Chen, Jin Li and Xinyi Huang[8] proposed New Publicly Verifiable Databases with Efficient Updates in 2014 that the notion of verifiable database (VDB) enables a resource-constrained client to securely outsource a very large database to a un trusted server so that it could later retrieve a database record and update it by assigning a new value. Also, any attempt by the server to tamper with the data will be detected by the client. Very recently, Catalano and Fiore proposed an elegant framework to build efficient VDB that supports public verifiability from a new primitive named vector commitment. In this paper, we point out Catalano-Fiore’s VDB framework from vector commitment is vulnerable to the so-called forward automatic update (FAU) attack. Besides, we propose a new VDB framework from vector commitment based on the idea of commitment binding. The construction is not only

public verifiable but also secure under the FAU attack. Furthermore, they proved that our construction can achieve the desired security properties.

Rafail Ostrovsky,Amit Sahai and Brent Waters[9] proposed Attribute Based Encryption with Non-Monotonic Access Structures in 2007 that they construct an Attribute-Based Encryption (ABE) scheme that allows a user's private key to be expressed in terms of any access formula over attributes. Previous ABE schemes were limited to expressing only monotonic access structures. They provide a proof of security for our scheme based on the Decisional Bilinear Diffie Hellman (BDH) assumption. Furthermore, the performance of this new scheme compares favorably with existing, less-expressive schemes.

2.1 INFERENCES FROM LITREATURE SURVEY

The CP-ABE may help us prevent security breach from outside attackers. But when an insider of the organization is suspected to commit the “crimes” related to the redistribution of decryption rights and the circulation of user information in plain format for illicit financial gains, how could we conclusively determine that the insider is guilty, Is it also possible for us to revoke the compromised access privileges? In addition to the above questions, we have one more which is related to key generation authority. A cloud user’s access credential (i.e., decryption key) is usually issued by a semi-trusted authority based on the attributes the user possesses. How could we guarantee that this authority will not (re-)distribute the generated access credentials to others. The widespread improvements in cloud computing, customers and groups are locating it an increasing number of attractive to save and proportion statistics thru cloud offerings. Cloud provider providers (consisting of Amazon, Microsoft, Apple, etc.) offer plentiful cloud primarily based totally offerings, starting from small-scale private offerings to big-scale business offerings. However, current statistics breaches, consisting of releases of personal photos, have raised worries concerning the privateness of cloud-controlled statistics. Actually, a cloud provider issuer is typically now no longer steady because of layout drawbacks of software program and device vulnerability. Then the crypt-DAC proposes 3 key techniques. The administrator appends a brand-new revocation key on the stop of its key listing and requests the cloud to replace this key listing with inside the coverage statistics. The length of the important thing listing but will increase with the revocation operations, and a person has to down load and decrypt a big key listing in every document access. This approach

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

A. This scheme incurs a considerable communication overhead.

B. This scheme, however, comes with a security penalty as the revocation operation is delayed to the next user's modification to the file. As a result, a newly revoked user can still access the file before the next writing operation.

C. This scheme, however, comes with a security penalty as the revocation operation is delayed to the next user's modification to the file. As a result, a newly revoked user can still access the file before the next writing operation.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

3.1.1 Software Requirements:

A. Windows 10

B. JDK 1.8

C. J2EE

D. Tomcat 7.0

E. MySQL

3.1.2 Hardware Requirements:

A. Hard Disk : 80GB and Above B.

RAM : 4GB and Above C.

Processor : PIV and Above

3.1.3 Algorithm Used:

A.AES-Advanced Encryption Standard:

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

It is based on a substitution-permutation network, also known as an SP network. It consists of a series of linked operations, including replacing inputs with specific outputs (substitutions) and others involving bit shuffling (permutations).

B. RSA-Rivest-Shamir-Adleman:

RSA (Rivest-Shamir-Adleman) is public key cryptosystem that is widely used for secure data transmission. It is also one of the oldest. The "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who publicly described the algorithm in 1977. An equivalent system was developed secretly in 1973 at Government Head Communication Quaters (GCHQ) (the British signal intelligence agency) by the English mathematician clifford clocks. That system was declassified in 1997.

In a public-key cryptosystem, the encryption system is public and distinct from the decryption key, which is kept secret (private). An RSA user creates and publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers are kept secret. Messages can be encrypted by anyone, via the public key, but can only be decoded by someone who knows the prime numbers.

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". Breaking RSA encryption is known as the RSA s as difficult as the factoring problem is an open question.There are no published methods to defeat the system if a large enough key is used.

RSA is a relatively slow algorithm. Because of this, it is not commonly used to directly encrypt user data. More often, RSA is used to transmit shared keys for symmetric key cryptography, which are then used for bulk encryption-decryption.

3.1.4 Technology Used:

A.J2E E (JSP, Servlets), JavaScript, HTML, CSS, AJAX.

3.1.5 SPECIFICATIONS

(i) Windows 10:

It is a major release of Microsoft's Windows NT operating system.It is direct successor to Windows 8.1.It is designed to adapt its user interface based on the type of device being used and available input methods.It offers two separate user interface optimized for mouse and keyboard, and a "Tablet mode" designed for touch screens.

(ii) JDK 1.8:

The JDK is a development environment for building applications using the java programming language.The JDK includes tools useful for developing and testing programs written in the java programming language and running on the java platform.

(iii) J2EE:

The J2EE platform provides choices for graphical user interfaces across a company's intranet or on the World Wide Web.Clients can run on desktops,laptops,PDAs,cell phones and other devices.

(iv) Tomcat 7.0:

It implements the Servlet 3.0 and javaServer Pages 2.2 specifications from the java community process, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

(v) AES algorithm:

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

(vi) RSA algorithm:

RSA is the most common public-key algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**It is a public key crypto-system that is widely used for secure data transmission.

(vii) JavaScript:

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user.

(viii) HTML:

HTML is the standard markup language for web pages.It is used to structure a web page and its content.

*(ix)*CSS:

It is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments.

(x) AJAX:

AJAX = Asynchronous JavaScript and XML. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

In this work, we have addressed the challenge of credential leakage in CP-ABE based cloud storage system by designing an accountable authority and revocable Crypt Cloud which supports white-box traceability and auditing. This is the first CP-ABE based cloud storage system that simultaneously supports white-box traceability, accountable authority, auditing and effective revocation. Specifically Crypt-DAC, a cryptographically enforced dynamic access control system on untrusted cloud. Crypt-DAC delegates the cloud to update encrypted files in permission revocations. Our approach can be also used in the case where the users' credentials are redistributed by the semi-trusted authority. We aim to provide confidentiality and access control for the cloud-hosted file data.

(i) Confidentiality:

our system stores encrypted data on the cloud, but never reveals the decryption keys to the cloud.This protects the confidentiality of the file data.

(ii) Read Access Control:

our system uses cryptography to enforce access control so that users can only read file data according to their access permissions.

(iii) Write access control:

For writing permission enforcement, our system relies on the cloud provider to validate.

The proposed system presents Crypt-DAC, a cryptographically enforced dynamic access control system on un trusted cloud. CryptDAC delegates the cloud to update encrypted files in permission revocations. In CryptDAC, a file is encrypted by a symmetric key list which records a file key and a sequence of revocation keys. In a revocation, the administrator uploads a new revocation key to the cloud, which encrypts the file with a new layer of encryption and updates the encrypted key list accordingly. Same as previous works, we assume a honest-butcurious cloud, i.e., the cloud is honest to perform the required commends (such as re encryption of files and properly update

previous encrypted files) but is curious to passively gathering sensitive information. Although the basic idea of layered encryption is simple, it entails tremendous technical challenges.

For instance, the size of key list and encryption layers would increase as the number of revocation operations, incurs additional decryption overhead for users to access files. To overcome such a problem, Crypt-DAC proposes three key techniques as follows. ϖ First, Crypt-DAC proposes delegation-aware encryption strategy to delegate the cloud to update policy data. For a file, the administrator appends a new revocation key at the end of its key list and requests the cloud to update this key list in the policy data. The size of the key list however increases with the revocation operations, and a user has to download and decrypt a large key list in each file access. To overcome this problem, we adopt the key rotation technique to compactly encrypt the key list in the policy data. As a result, the size of the key list remains constant regardless of revocation operations.

Second, Crypt-DAC proposes adjustable onion encryption strategy to delegate the cloud to update file data. For a file, the administrator requests the cloud to encrypt the file with a new layer of encryption. Similarly, the size of the encryption layers increases with the revocation operations, and a user has to decrypt multiple times in each file access. To overcome this problem, we enable the administrator to define a tolerable bound for the file. Once the size of encryption layers reaches the bound, it can be made to not increase anymore by delegating encryption operations to the cloud. As a result, the administrator can flexibly adjust a tolerable bound for each file (according to file type, access pattern, etc.)

To achieve a balance between efficiency and security. ϖ Crypt- DAC proposes delayed de-onion encryption strategy to periodically refresh the symmetric key list of the file and remove the bounded encryption layers over it through writing operations. In specific, the next user to write to the file encrypts the writing content by a new symmetric key list only containing a new file key, and updates the key list in the policy data. With this strategy, Crypt-DAC periodically removes the bounded encryption layers of files while amortizing the burden to a large number of writing users.

4.1 PROCESS MODEL:

(i) System Model:

We consider a scenario where companies contract with a commercial cloud provider (e.g., cloud, Microsoft Azure) to outsource enterprise storage. There are three types of entities in our system model: a cloud provider, an access control administrator, and a large number of users. The cloud provider is responsible for the data storage and management. The data includes file data of users in the company, as well as policy data regulating access policies for these files. Both the policy/file data are encrypted prior to being uploaded to the cloud provider. The access control administrator is responsible for managing access policies of the file data. It assigns/revokes access permissions by creating, updating, and distributing cryptographic keys used to encrypt files. Users may download any policy/file data from the cloud, but are only allowed to decrypt and read files according to their access permissions. We do not consider data deduplication issue. IF needed, secure data deduplication technique can be used here. We also assume that all parties communicate via pairwise secure channels (e.g., SSL/TLS tunnels).

(ii) Threat Model*:*

In our threat model, we consider that the administrator is honest. The users may try to access the file data out. Cloud enabled data access control. Restrictions apply. access permissions by compromising the cloud provider. Same as previous works we assume a honest but-curious cloud provider. Honest means that the cloud provider honestly follows the commands of the administrator/users such as re-encryption of policy/file data and properly update previous policy/file data. Since any errors due to the provider's misbehavior will harm its reputation, we believe that the cloud provider has incentive to follow the commands required by its customers. However, the cloud provider may be curious to passively gathering sensitive information to acquire commercial benefits as it is hard to be detected. We notice that the honest-but-curious assumption is critical to resist collusion between the cloud provider and revoked users. Generally, state of the art works fall in two ways to revoke access privileges. The first is for a data owner/administrator to download, re-encrypt, and upload the policy/file data to the cloud provider. The second is to delegate the cloud provider to directly re-encrypt the policy/file data. In both cases, if a malicious cloud provider does not properly update the previous policy/file data and retains a copy of it before the re-encryption, then the revoked users can continuously access the files. As policy/file data is fully managed by the cloud

provider, how to resist such collusion attack without the honest-but-curious assumption is still an open problem.

***(iii)*Security Goals**:

We aim to provide confidentiality and access control for the cloud-hosted file data. Confidentiality: our system stores encrypted data on the cloud, but never reveals the decryption keys to the cloud. This protects the confidentiality of the file data. Read Access Control: our system uses cryptography to enforce access control so that users can only read file data according to their access permissions. Write access control: for writing permission enforcement, our system relies on the cloud provider to validate write privileges of users prior to file updates.

(iv) Role based Access Control:

We design and analyze Crypt-DAC based on the role-based access control model named (RBAC0) [13], which is widely used in practical applications. RBAC0 model describes per mission management through the use of abstraction: roles describe the access permissions associated with a particular (class of) job function, users are assigned to the set of roles entailed by their job responsibilities, and a user is granted access to an object if they are assigned to a role that is permitted to access that object. More formally, the state of an RBAC0 model can be described as follows:

- U: a set of users
- R: a set of roles
- P: a set of permissions (e.g., (file, op))
- PA R × P: a permission assignment relation
- UR U × R: a user assignment relation
- auth(u, p) ¼ 9 r: [(u, r) 2 UR] ^ [(r, p) 2 PA]: the authorization predicate auth: U × P ! B determines whether user u has permission p.

(iv) Cryptographic Tools:

Symmetric/Asymmetric Cryptography. Our construction makes use of symmetric-key encryption scheme (GenSym, EncSym, DecSym), public-key encryption scheme (GenPub,

EncPub, DecPub) and digital signature scheme (GenSig, Sign, Ver). Key Rotation Scheme. Key rotation [15] is a scheme (GenRo, B-Dri, F-Dri) in which a sequence of keys can be produced from an initial key and a secret key. Only the owner of the secret key can derive the next key in the sequence, but any user knowing a key in the sequence can derive all previous versions of the key. We next describe the algorithms of this scheme. On input a security parameter 1n, this algorithm outputs a secret-public key pair. On input a key ki in a key sequence and a secret key, this algorithm outputs the next key kiþ1 in the sequence. On input a key ki in a key sequence and a public key, this algorithm outputs all previous versions of the key in the sequence

(v) Basic Construction:

In a cryptographically enforced RBAC0 system, a user u is associated with a user read key and a user write key. A role r is associated with a role key .A file is associated with a file key

Access Enforcement. For each user u, the administrator distributes a certificate for the user write key of u through a user (U) tuple.

This tuple contains a user name u, a verification key , and a signature of the administrator signed over .For each (u, r) the RBAC0 state (i.e., there exists a user u that is a member of r), the administrator distributes the role key of r to u through a role key (RK) tuple.

This tuple provides u with cryptographically-enforced access to the decryption key of r. For each (r, (f, op)) 2 PA in the RBAC0 state (i.e., there exists a role r with permission to f), the administrator distributes the file key off to r through a file key (FK) tuple.

This tuple provides r with cryptographically-enforced access to the file key k for f. For each file f, the administrator distributes f through a file (F) tuple:.

This tuple contains a file name fn of f and a ciphertext of f. File Access. If a user u with authorization to read a file f (i.e., 9r: (u, r) 2 UR ^ (r, f, Read) 2 PA) wishes to do so, u downloads an RK tuple for the role r to decrypt the decryption key dkr by dku. u also downloads an FK tuple for the file f to decrypt the file key k by dkr. Finally, u downloads a F tuple to decrypt the file f by k. If a user u with authorization to write to a file f via membership in role r (i.e., 9r: (u, r) 2 UR ^ (r, f, RW) 2 PA) wishes to do so, u uploads a F tuple encrypting the new file content f' as well as a signature du over the F tuple signed by the user write key of u. On the other hand, the cloud provider checks (1) if there is an RK tuple assigning u as a member of r and an FK tuple assigning r with permission RW to f; and (2) if there is a U tuple

containing vku that verifies du as a valid signature over the F tuple. If both checks passed, the cloud provider executes the writing operation. Access Revocation. The administrator may need to revoke a permission of a role, or revoke a membership of a user. We only describe the user revocation case as the role revocation case is analogous. Removing a user from a role r entails: (1) re-encrypting a new role key of r stored in RK tuples, (2) re-encrypting new file keys stored in FK tuples accessible by r, and (3) re-encrypting files stored in F tuples by the new file keys. All these steps must be carried out by an administrator as only the administrator can modify the access authorization. In the first step, a new role key of r is generated. This key is then encrypted into a new RK tuple for each remaining member u of r to replace the old RK tuple. This step prevents user from accessing the new role key of r. In the second step, a new file key is generated for each file f accessible by r. This key is then encrypted into a new FK tuple for each role r' (including r) that has access to f, and the new FK tuple is uploaded to replace the old FK tuple. This step prevents usere from accessing the new file keys. In the third step, each file to which r has access is re-encrypted into a new F tuple by the new file key. The new F tuple is then uploaded to replace the old F tuple. This step prevents user from accessing the files by using cached old file keys. We emphasize that it is important to re-encrypt the files by the new file keys as may cache the old file keys of these files and try to access them after the revocation. For example, suppose user is assigned to three roles and can access 200 files. Can first access all of these files to cache their file keys when user join the system. Later, evoked from one of its three roles and cannot access some of the 200 files anymore. However, user can still use the cached file keys to access these files if they are not re-encrypted by new file keys in the revocation. Design Issues for Revocation The revocation in this basic construction is not suitable for realistic dynamic access control scenario due to its prohibitive overhead in file data re-encryption. Due to the multi-to one property of access permission relations among users, roles, and files in RBAC0 model, the administrator needs to download, decrypt, re-encrypt, and upload a large number of F tuples, incurring potentially high bandwidth consumptions. For example, suppose the administrator needs to revoke the membership from r and r has permission to 100 files. Then, the administrator needs to re-encrypt all of the 100 files in the revocation. Previous Designs. In response, Garrison et al. [12] proposed two revocation schemes. The first scheme requires the administrator to re-encrypt file data by itself in a revocation. This scheme completes the revocation immediately with a potentially high communication overhead. Differently, the second scheme relies on next users writing to the F tuples to re-encrypt the F tuples. This scheme, however, comes with security penalty as it

delays the revocation to the next writing, creating a vulnerability window in which revoked users can continuously access the F tuples which they have accessed previously and cached the file keys. To alleviate the overhead of file data re-encryption, Wang et al. proposed another revocation scheme, in which the symmetric homomorphic encryption scheme is used to encrypt the file data. Instead by re-encrypting the file data by itself, the scheme enables the administrator to delegate the cloud to update F tuples from old file keys to new file keys without decryption. The problem, however, is that the cost of homomorphic symmetric encryption is com parable with public key encryption schemes, incurring prohibitive computation overhead during file reading/writing.

4.2 ARCHITECTURE OF CRYPT- DAC

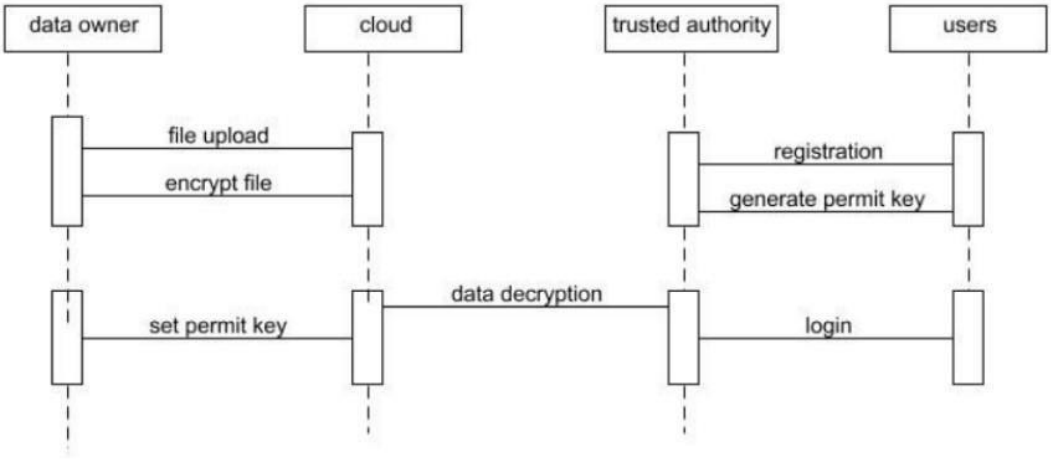


Fig.4.1-sequence diagram of crypt-dac

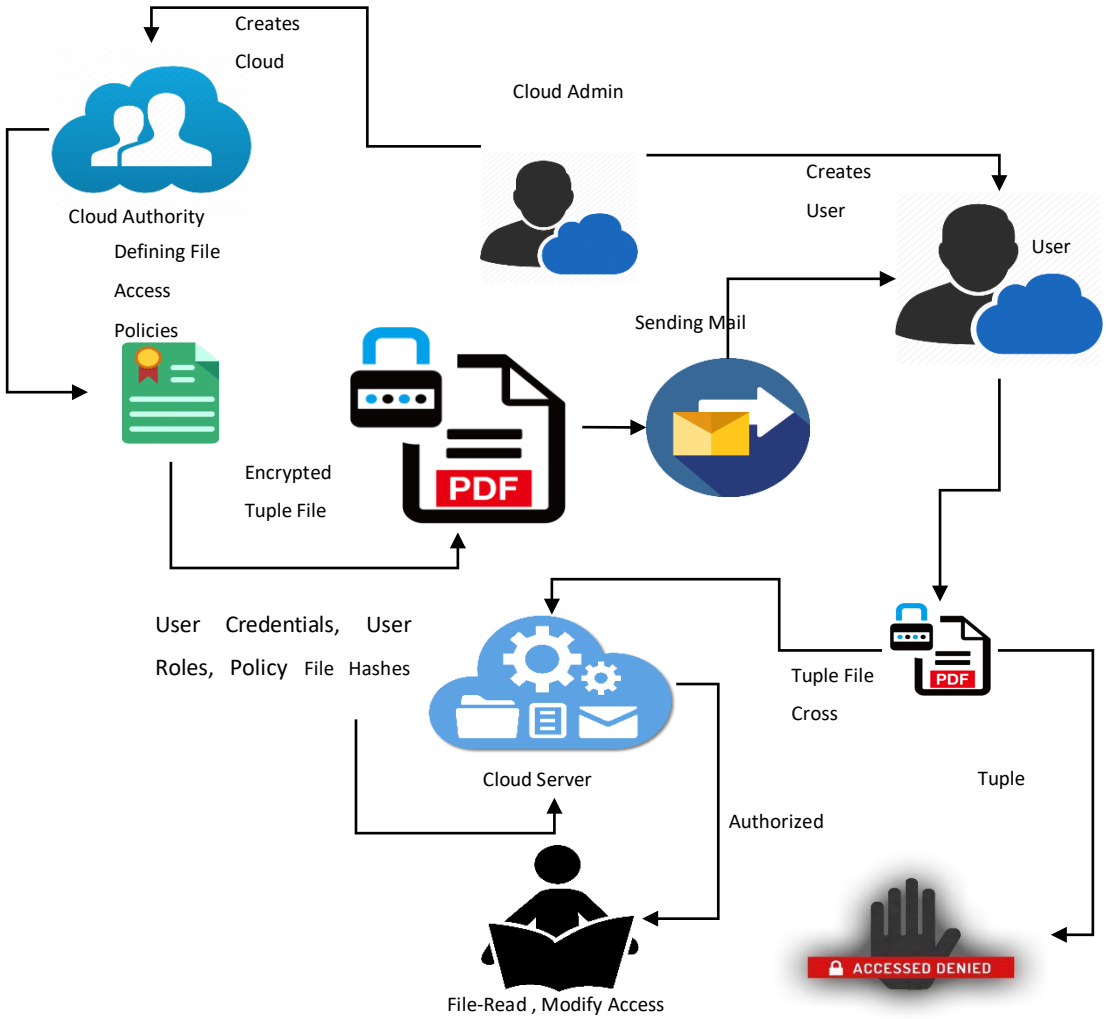

Creates
Cloud
Cloud Admin
Cloud Authority
Creates
User
User
Defining File
Access
Policies
Sending Mail
PDF
Encrypted
Tuple File
User Credentials, User
Roles, Policy File Hashes
Tuple File
Cross
Cloud Server
Tuple
Authorized
ACCESSED DENIED
File-Read , Modify Access

Fig.4.2-System architecture of crypt-dac

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED SYSTEM

Crypt-DAC develops new techniques using lightweight symmetric encryption scheme. In Crypt-DAC, a F tuple (file) is encrypted by a symmetric key list (k0, k1,..., kt) which records a file key k0 and a sequence of revocation keys k1,..., kt . Crypt-DAC uses the innermost encryption layer to protect the file against the cloud provider and the outermost encryption layer to protect the file against revoked users. In the ith revocation, the administrator uploads a new revocation key ki to the cloud, which encrypts the file with a new layer of encryption. After this procedure, the revoked user cannot access the file as he cannot access key. An illustrative example is shown in Fig. 3. Compared with previous designs, Crypt-DAC achieves efficient revocation, immediate revocation, and efficient file access simultaneously. For revocation efficiency, Crypt-DAC incurs lightweight communication overhead at the administrator side as it does not need to download and reupload file data but only needs to upload keys to the cloud. For immediate revocation, the permissions of users are immediately revoked as the files are re-encrypted. For file access efficiency, the files are still encrypted by symmetric keys. To further avoid users to decrypt multiple encryption layers in file access operations, Crypt-DAC proposes three key techniques to constrain the size of key list and encryption layers for files as follows.

4.4 PROJECT MANAGEMENT PLAN

(i) Organization profile creation & Key Generation

User has an initial level Registration Process at the web end. The users provide their own personal information for this process. The server in turn stores the information in its database. Now the Accountable STA (semi-trusted Authority) generates decryption keys to the users based on their Attributes Set (e.g. name, mail-id, contact number etc..,). User gets the provenance to access the Organization data after getting decryption keys from Accountable STA.

(ii) Data Owners File Upload

In this module data owners create their accounts under the public cloud and upload their data into public cloud. While uploading the files into public cloud data owners will encrypt their data using RSA Encryption algorithm.

(iii) File Permission & Tuple Creation

Different data owners will generate different file permission keys to their files and issues those keys to users under the organization to access their files. And also generates policy files to their data that who can access their data. Policy File will split the key for read the file, write the file, download the file and delete the file.

(iv) Tracing who is guilty

Authorized DUs are able to access (e.g. read, write, download, delete and decrypt) the outsourced data. Here file permission keys are issued to the employees in the organization based on their experience and position to their registered. Senior Employees have all the permission to access the files (read, write, delete, & download).

(v) Background work

Current cryptographically enforced access control schemes can be classified as follows.

A. Hierarchy access control:

Gudes et al. explore cryptography to enforce hierarchy access control without considering dynamic policy scenarios. Akl et al. propose a key assignment scheme to simplify key management in hierarchical access control policy. Also, this work does not consider policy update issues. Later, Atallah et al. propose a method that allows policy updates, but in the case of revocation, all descendants of the affected node in the access hierarchy must be updated, which involves high computation and communication overhead.

B. Role based access control:

Ibraimi et al. cryptographically supports role-based access control structure using mediated public encryption. However, their revocation operation relies on additional trusted infrastructure and an active entity to re-encrypt all affected files under the new policy. Similarly, Nali et al. enforce role-based access control structure using public-key cryptography, but requires a series of active security mediators. Ferrara et al. define a secure model to formally prove the security of a cryptographically enforced RBAC system. They further show that an ABE-based construction is secure under such model. However, their work focuses on theoretical analysis.

C. Attribute based access control:

Pirretti et al. propose an optimized ABE-based access control for distributed file systems

and social networks, but their construction does not explicitly address the dynamic revocation. Sieve is a attribute based access control system that allows users to selectively expose their private data to third web services. Sieve uses ABE to enforce attribute-based access policies and homomorphic symmetric encryption to encrypt data. With homomorphic symmetric encryption, a data owner can delegate revocation tasks to the cloud assured that the privacy of the data is preserved. This work however incurs prohibitive computation overhead since it adopts the homomorphic symmetric encryption to encrypt files.

D. Access matrix:

GORAM allows a data owner to enforce an access matrix for a list of authorized users and provides strong data privacy in two folds. First, user access patterns are hidden from the cloud by using ORAM techniques. Second, policy attributes are hidden from the cloud by using attribute-hiding predicate encryption. The cryptographic algorithms, however, incur additional performance overhead in data communication, encryption and decryption. Also, GORAM does not support dynamic policy update. Over encryption is a cryptographical method to enforce an matrix on outsourced data. Over-encryption uses double encryption to enforce the whole access matrix. As a result, the administrator has to rely on the cloud to run complex algorithms over the matrix to update access policy, assuming a high level of trust on the cloud.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The Application at this side controls and communicates with the following three main general components. Embedded browser in charge of the navigation and accessing to the web service;Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following. Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database system, Verification Engine, Output Module.

A. Environment of TOMCAT:

Environment Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets. The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments. Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

5.1.1 ADJUSTABLE ONION ENCRYPTION:

Adjustable onion encryption enables the administrator to delegate the cloud provider to update Files. The administrator only needs to upload a new revocation key to the cloud provider. Upon receiving the key, the cloud provider uses it to encrypt the files with a new layer of encryption and then deletes it. To constrain the
size of the encryption layers, adjustable onion encryption provides two modes: security mode and efficiency mode. Such a design enables the administrator to define a tolerable bound for a file. Initially, the strategy works in the security mode and the encryption layers increase as revocations happen. Once the size of the encryption
layers reach the bound, it turns to the efficiency mode to constrain the encryption layers by putting more trust on the cloud. As a result, the administrator can flexibly adjust a tolerable bound for each file according to file type, access pattern, etc., to achieve a balance between efficiency and security

5.1.2 KEY ROTATION SCHEME:

Key rotation is a scheme which a sequence of keys can be produced from an initial key and a secret key. Only the owner of the secret key can derive the next key in the sequence, but any user knowing a key in the sequence can derive all previous versions of the key.

5.1.3 ACCESS CONTROL ADMINISTRATOR:

The access control administrator is responsible for managing access policies of the file data. It assigns/revokes
access permissions by creating, updating, and distributing cryptographic keys used to encrypt files.

D. USERS:

Users may download any policy/file data from the cloud, but are only allowed to decrypt and read files
according to their access permissions.

E. CLOUD PROVIDER:

The cloud provider is responsible for the data storage and management. The data includes file data of users in
the company, as well as policy data regulating access policies for these files. Both the policy/file data are
encrypted prior to being uploaded to the cloud provider.

5.1.4 Constraints in Implementation

The hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation.

5.2 ALGORITHM

Algorithm 1. revokeUser(u)

1: For each role r that u is assigned to:

2: REVOKEU(u, r);

3: Req C.P. to delete hU, (u, vku), dSUi;

4:

5: procedure REVOKEU(u, r)

6: DAE-RK(r);

7: DAE-FK(r);

8: For each fn with hFK, r, (fn, op), ci:

9: ONION-ENCRYPTION(fn);

10:

11: procedure DAE-RK(r)

12: Generate a new role key (ekr, dkr) for r;

13: For each hRK, u', r, ci with u' 6¼ u:

14: Admin:

15: Send EncPub eku0 ðdkr) to C.P.;

16: C.P.:

17: Update hRK, u', r, ci as hRK, u', r, EncPub eku0 ðdkrÞi; 18:

19: procedure DAE-FK(r)

20: For each fn with hFK, r, (fn, op), ci:

21: Admin:

22: Generate a new revocation key ktþ1 D Driðkt, rskfn);

23: For each role r' with permission to fn:

24: Compute c0 = EncPub ekr0 ðk0, ktþ1, rpkfn , t þ 1);

25: Send c0 to C.P.;

26: C.P.:

27: For each role r' with permission to fn:

28: Update hFK, r', (fn, op), ci as hFK, r', (fn, op), c'i;

29:

30: procedure ONION-ENC(fn)

31: Admin:

32: Compute k^tþ1 hashðktþ1, t+1);

33: Send (k^tþ1/k^tþ1, k^t) to C.P.;

34: C.P.: hF, fn, ci:

35: Compute c EncSym k^tþ1 (c)/c EncSym k^tþ1 ðDecSym k^t (c));

Permission assignment includes adding a new user addUser(u), adding a new role addRole(ur), assigning a user to a role assignU(u,r); rÞ, and assigning a role to a file with Read/RW permission assign(ur)hfn; Due to the space constraint, we omit the algorithm details of these operations. A user u uses addUser(u,r) to join the system. The administrator uses addRole(u,r) to add a new role r into the system, uses assign(u,r); rÞ to assign a user u with a role r, and uses assign(u,r); opiÞ to assign a role to a file with Read/ RW permission. File Operation. File operation includes file creation addFile(u,r), file read read (as(u,r) described in Algorithm 3), and file write writeðfn; uÞ (as described in Algorithm 4). A user u uses addFile(u,r)to create a new file fn in the system and uses read(u,r); uÞ to read a file fn from the cloud provider. Also, a user u uses write(u,r); u to write to a file fn stored in the cloud provider. This algorithm invokes DELAYED DE-ONION(hRK, u, r, ci), which implements the delayed de-onion encryption strategy, to refresh the symmetric key list of fn. We noticed that in some

cases, u does not need to execute the delayed de-onion encryption strategy as no revocation happens subject to fn

Algorithm 2. revokeRole(r)

1: Remove (r, ekr) from ROLES;

2: For each permission p = hfn, opi that r has access to:

3: REVOKEP(r, hfn, Readi);

4:

5: procedure REVOKEP(r, hfn, RWi)

6: Admin:

7: Send hFK, r, (fn, Read), ci to C.P.;

8: C.P.:

9: Update hFK, r, (fn, RW), ci as hFK, r, (fn, Read), ci;

10:

11: procedure REVOKEP(r, hfn, Readi)

12: Req C.P. to delete all hRK, –, r, – i;

13: Req C.P. to delete hFK, r, (fn, –), –i;

14: VDAE-FK(fn);

15: ONION-ENCRYPTION(fn);

16:

17: procedure VDAE-FK(fn)

18: Admin:

19: Generate a new revocation key ktþ1 D Driðkt, rskfn);

20: For each role r’ 6¼ r with permission to fn:

21: Compute c0 = EncPub ek0 r ðk0, ktþ1, rpkfn , t þ 1);

22: Send c0 to C.P.;

23: C.P.:

24: For each role r’ 6¼ r with permission to fn:

25: Update hFK, r’, (fn, op), ci as hFK, r’, (fn, op), c’);

Algorithm 3. read(fn, u)

1: User u:

2: Send (u, r, fn) to C.P.;

3: C.P.:

4: If there exists an RK tuple hRK, u, r, ci ^

5: a FK tuple hFK, r, (fn, op), c’i ^

6: a Ftuple hF, fn, c”i:

7: Then

8: Return the RK tuple, FK tuple, and F tuple to u;

9: Else

10: Return ? to u;

11: Useru:

12: Compute dkr DecPub dku (c);

13: Compute (k0, kt, rpkfn , t) DecPub dkr (c’);

14: For i = t & i > 1 & i :

15: Compute ki 1 F Drirpkfn ðki);

16: Compute ^ kt = hashðkt, t);

17: Compute c” DecSym k^ t (c”);

18: For i = T & i 0 & i :

19: Compute ^ ki = hashðki, i);

20: Compute c” DecSy

Algorithm 4. write(fn, u)

1: User u:

2: Send (fn, write request) to C.P.;

3: C.P.:

4: Include all the FK tuples containing fn into FK-set;

5: Return hRK, u, r, ci to u;

6: User u:

7: DELAYED DE-ONION(hRK, u, r, ci);

8:

9: procedure DELAYED DE-ONION(hRK, u, r, ci)

10: User u:

11: Compute k00 GenSym();

12: Compute c’ EncSym k00 (f’);

13: Compute du Signsku (F, fn, c’);

14: Send hF, fn, c’, dui to C.P.;

15: Send (k00, fn) to Admin;

16: Admin:

17: For each role r’ with permission to fn:

18: Compute c” EncPub pkr ðk00, rpkfn , 0);

19: Insert c” into C-set;

20: Send C-set to C.P.;

21: C.P.:

22: If there exists a U tuple hU, (u, vku), dSUijvalid Vervku (F, fn, c’) ^

23: an RK tuple hRK, u, r, ci ^ a FK tuple hFK, r, (fn, RW), c’i:

24: Then Write hF, fn, c’i;

25: For each hFK, r’, (fn, RW), c’i 2 FK-set and a proper c” 2 C-set:

26: Update hFK, r’, (fn, RW), c’i to hFK, r’, (fn, RW), c”i;

27: Else

28: Return to u;

Instead, we propose to use subitizable signature scheme to ensure data integrity. Subitizable signature scheme is a special signature scheme in which a signer can delegate a proxy to modify a certain portion of a signed message without invalidating the attached signature. For RK and FK tuples, the administrator uses subitizable signature scheme to sign them and delegates the cloud provider to modify the encryption portion of them. In this way, the cloud provider can directly update the RK and FK tuples without invalidating the attached signatures. Differently for F tuples, the administrator uses general signature scheme to sign them. The reason is that in a revocation, the administrator only delegates the cloud provider to update the encryption portion of F tuples by adding a new encryption layer over it. This operation does not change the encrypted file contents, and thus does not invalidate the attached signatures. As a result, we do not need to use subitizable signature scheme to sign F tuples. With our extension, a malicious cloud provider cannot arbitrarily modify RK, FK, and F tuples or generate fake tuples. Also, maliciously modifying the encryption portion of RK and FK tuples can be easily detected by users.

5.3 TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough. Testing is one of the important steps in the software development phase. Testing is one of the important steps in th e software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

* Static analysis is used to investigate the structural properties of the Source code.

*Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

RESULT AND DISCUSSION

The main contribution to this document is DAC in cryptography. This is the engine that enables practical encryption enforcement of dynamic access to manage within untrusted cloud publishers. DAC in cryptography realizes the dream of using the 3 techniques. Let the cloud delegation replace privacy coverage information while maintaining the use of delegation-aware encryption methods. This is widely used to keep away from fancy re-encryption in reports Information on the admin facet includes use of the tune able union encryption method. Additionally, the non-bonded encryption method is behind schedule to avoid the overhead of inspecting reports. A theoretical evaluation and an overall performance evaluation show that DAC in cryptography, compared to previous schemes, not only guarantees the same protection under the honest but curious version, but also allows revocation. It shows that it achieves more significant performance in terms of other schemes.

In previous work the architecture is built to provide the security which monitors and protects the virtual machines in real time. However, the attack behavior of malicious VM

needed a further analysis. In our paper, malicious behaviors are analyzed and any visit of malicious user to the cloud hosted data will be detected. This have been done using white box traceability.

Also, Security and privacy had been improved towards remote cloud storage. An attribute-based d proxy re-encryption method had been proposed to encrypt the data in the cloud without downloading any data. In this paper a new revocation scheme has been proposed that enables cloud to directly re-encrypt the file without decryption.

In previous works to explore cryptography to enforce hierarchy access control without considering dynamic policy scenarios. Key assignment scheme had been proposed to simplify key management. Also, this work does not consider policy update issues. Later, a key hierarchy method proposed to enable policy updates, For
revocation, DAC in cryptography is best for all communication overhead problems.

In 2019, Garrison proposed revocation scheme which comes with a security penalty because in this scheme revocation operation will be delayed to the next user's modification to the data.
This has been rectified in this paper using Access Based Enumeration.

In previous papers revocation schemes always comes with security penalty. But this paper supports effective revocation. The proposed system delegates the cloud to update the encrypted files in permission revocations.

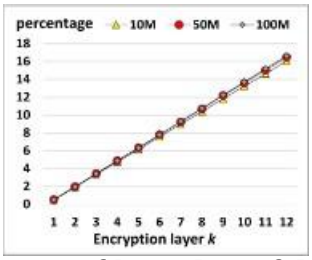


Fig.6.1. performance in file reading of dac-crypt and HOre at user side

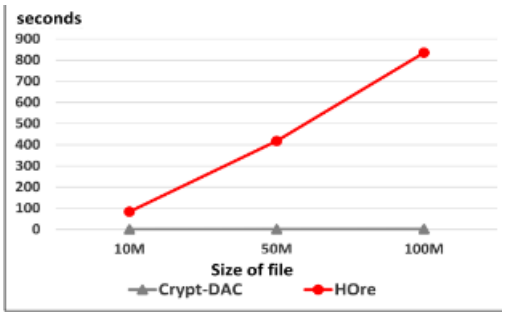


Fig.6.2. performance of dac-crypt and HOre at cloud side

Current cryptographically enforced access control schemes can be classified as follows. Hierarchy Access Control: Gudes et al. explore cryptography to enforce hierarchy access control without considering dynamic policy scenarios. Akl et al. propose a key assignment scheme to simplify key management in hierarchical access control policy. Also, this work does not consider policy update issues. Later, Atallah et al. propose a key hierarchy method to enable policy updates, for revocation however, all descendants of the affected node in the hierarchy need to be updated, which involves high computation and communication overhead. Role based Access Control: Ibraimi et al. cryptographically support role-based access control structure using mediated public encryption. However, their revocation operation relies on additional trusted infrastructure and an active entity to re-encrypt all affected files under the new policy. Also, Nali et al. enforce role-based access control structure using public-key cryptography, but involves a series of active security mediators. Ferrara et al. define a secure model to formally prove the security of a crypto graphically-enforced RBAC system. They further show that an ABE-based construction is secure under such model. However, their work focuses on theoretical analysis. Attribute based Access Control. Pirretti et al. propose an optimized ABE-based access control for distributed file systems and social networks, but their construction does not explicitly address the dynamic revocation. Sieve is a attribute based access control system that allows users to selectively expose their private data to third web services. Sieve uses ABE to enforce attribute-based access policies and homomorphic symmetric encryption to encrypt data. With homomorphic symmetric encryption, a data owner can delegate revocation tasks to the cloud assured that the privacy of the data is preserved. This work however incurs prohibitive computation overhead since it adopts the homomorphic symmetric encryption to encrypt files. Access Matrix. GORAM allows a data owner to enforce an access matrix for a list of authorized users and provides strong data privacy in two folds. First, user access patterns are hidden from the cloud by using ORAM techniques. Second,

policy attributes are hidden from the cloud by using attribute-hiding predicate encryption. The cryptographic algorithms, however, incur additional performance overhead in data communication, encryption and decryption. Also, GORAM does not support dynamic policy update. Over-encryption is a cryptographical method to enforce an access matrix on outsourced data. Over-encryption uses double encryption to enforce the whole access matrix. As a result, the administrator has to rely on the cloud to run complex algorithms over the matrix to update access policy, assuming a high level of trust on the cloud

CONCLUSION

We have presented, DAC in cryptography, a system that provides convenient dynamic access control through cryptography in untrusted cloud providers. DAC in cryptography uses three techniques to achieve the goal. We propose using a delegate to the cloud to update policy data in a privacy preserving 4300 ways. We propose to use a tune able onion encryption strategy to avoid costly re-encryption of file data on the admin side. In side. In addition, we proposed a delayed de-onion encryption strategy 4304 to avoid file read overhead. Theoretical analysis and performance evaluation show that DAC in cryptography achieves an order of magnitude higher blocking efficiency while ensuring the same security properties under an honest but curious threat model compared to previous schemes.

REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute based encryption,” in Proc. IEEE Symp. Security Privacy, 2007, pp. 321–334.

[2] X. Wang, Y. Qi, and Z. Wang, “Design and implementation of SecPod: A framework for virtualization-based security systems,” IEEE Trans. Depend. Secure Comput., vol. 16, no. 1, pp. 44–57, Jan./ Feb. 2019.

[3] J. Ren, Y. Qi, Y. Dai, X. Wang, and Y. Shi, “AppSec: A safe execution environment for security sensitive applications,” in Proc. 11th ACM SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environments, 2015, pp. 187–199.

[4] V. Goyal, A. Jain, O. Pandey, and A. Sahai, “Bounded ciphertext policy attribute based encryption,” in Proc. Int. Colloquium Automata Languages Programm., 2008, pp. 579–591.

[5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in Proc. 13th ACM Conf. Comput. Commun. Security, 2006, pp. 89–98.

[6] J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” in Proc. Theory Appl. Cryptographic Techn., 27th Annu. Int. Conf. Advances Cryptology, 2008, pp. 146–162.

[7] S. Muller and S. Katzenbeisser, “Hiding the policy in cryptographic access control,” in Proc. Int. Workshop Security Trust Manage., 2011, pp. 90–105

[8] R. Ostrovsky, A. Sahai, and B. Waters, “Attribute-based encryption with non-monotonic access structures,” in Proc. 14th ACM Conf. Comput. Commun. Security, 2007, pp. 195–203.

[9] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in Proc. 24th Annu. Int. Conf. Theory Appl. Cryptographic Techn., 2005, pp. 457–473.

[10] T. Ring, “Cloud computing hit by celebgate,” 2015. [Online]. Available: http://www.scmagazineuk.com/cloud-computinghit-by-celebgate/article/370815/

[11] X. Jin, R. Krishnan, and R. S. Sandhu, “A unified attribute-based access control model covering DAC, MAC and RBAC,” in Proc. 26th Annu. IFIP WG 11.3 Conf. Data Appl. Security Privacy, 2012, pp. 41–55.

[12] W. C. Garrison III , A. Shull, S. Myers, and, A. J. Lee, “On the practicality of cryptographically enforcing dynamic access control policies in the cloud,” in Proc. IEEE Symp. Security Privacy, 2016, pp. 819–838.

[13] R. S. Sandhu, “Rationale for the RBAC96 family of access control models,” in Proc. ACM Workshop Role-based Access Control, 1995, Art. no. 9.

[14] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, “Secure and efficient cloud data deduplication with randomized tag,” IEEE Trasactions Inform. Forensics Security, vol. 12, no. 3, pp. 532–543, Mar. 2017.

[15] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, K. Fu, “Plutus: Scalable secure file sharing on untrusted storage,” in Proc. 2nd USENIX Conf. File Storage Technol., 2003, pp. 29-42.

[16] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, “Verifiable auditing for outsourced database in cloud computing,” IEEE Trans. Comput., vol. 64, no. 11, pp. 3293–3303, Nov. 2015.

[17] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, “Towards achieving flexible and verifiable search for outsourced database in cloud computing,” Future Generation Comput. Syst., vol. 67, pp. 266–275, 2017.

[18] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, “New publicly verifi able databases with efficient updates,” IEEE Trans. Depend. Secure Comput., vol. 12, no. 5, pp. 546–556, Sep.-Oct. 2015.

[19] T. Jiang, X. Chen, and J. Ma, “Public integrity auditing for shared dynamic cloud data with group user revocation,” IEEE Trans. Comput., vol. 65, no. 8, pp. 2363–2373, Aug. 2016.

[20] D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” SIAM J. Comput., vol. 32, no. 3, 2003.

[21] J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” in Proc. Theory Appl. Cryptographic Techn., 27th Annu. Int. Conf. Advances Cryptology, 2008, pp. 146–162.

[22] E. Shen, E. Shi, and B. Waters, “Predicate privacy in encryption systems,” in Proc. 6th Theory Cryptography Conf. Theory Cryptogra phy, 2009, pp. 457–473.

[23] F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan, “Sieve: Cryptographically enforced access control for user data in untrusted clouds,” in Proc. USENIX Symp. Netw. Syst. Des. Imple mentation, 2016, pp. 611–626.

[24] D. Boneh, K. Lewi, H. Montgomery, and A. Raghu Raghunathan , “Key homomorphic PRFs and their applications,” in Proc. Annu. Cryptology Conf., 2013, pp. 410–428.

[25] M. Maffei, G. Malavolta, M. Reinert, and D. Schroder, “Privacy and access control for outsourced personal records,” in Proc. IEEE Symp. Security Privacy, 2015, pp. 341–358. [26] J. R. Lorch, B. Parno, J. W. Mickens, M. Raykova, and J. Schiffman, “Shroud: Ensuring private access to large-scale data in the data center,” in Proc. USENIX Conf. File Storage Technol., 2013, pp. 199–214.

[27] E. Gudes, “The design of a cryptography based secure file system,” IEEE Trans. Softw. Eng., vol. 6, no. 5, pp. 411–420, Sep. 1980.

[28] S. G. Akl and P. D. Taylor, “Cryptographic solution to a problem of access control in a hierarchy,” ACM Trans. Comput. Syst., vol. 1, no. 3, 1983.

[29] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, “Dynamic and efficient key management for access hierarchies,” ACM Trans. Inf. Syst. Security, vol. 12, no. 3, 2009, Art. no. 18.

[30] L. Ibraimi, “Cryptographically enforced distributed data access con□trol,” Ph.D. dissertation, Electrical Engineering, Mathematics and Computer Science, Univ. Twente, Enschede, Netherlands, 2011.

[31] D. Nali, C. M. Adams, and A. Miri, “Using mediated identity□based cryptography to support role-based access control,” in Proc. Int. Conf. Inf. Security, 2004, pp. 245–256.

[32] A. L. Ferrara, G. Fuchsbauer, and B. Warinschi, “Crypto□graphically enforced RBAC,” in Proc. IFIP Annu. Conf. Data Appl. Security Privacy, 2013, pp. 3–19.

[33] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, “Secure attrib□utebased systems,” in Proc. 13th ACM Conf. Comput. Commun. Security, 2006, pp. 99–112.

[34] S. De Capitani di Vimercati , S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Over-encryption: Management of access control evolution on outsourced data,” in Proc. 33rd Int. Conf. Very Large Data Bases, 2007, pp. 123–134.

[35] S. De Capitani di Vimercati , S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Encryption policies for regulating access to out sourced data,” ACM Trans. Database Syst., vol. 35, no. 2, 2010, Art. no. 12.

[36] T. L. Hinrichs, D. Martinoia, W. C. Garrison III , A. J. Lee, A. Pane bianco, and L. Zuck, “Application-sensitive access control evalua tion using parameterized expressiveness,” in Proc. IEEE 26th Comput. Security Foundations Symp., 2013, pp. 145–160.

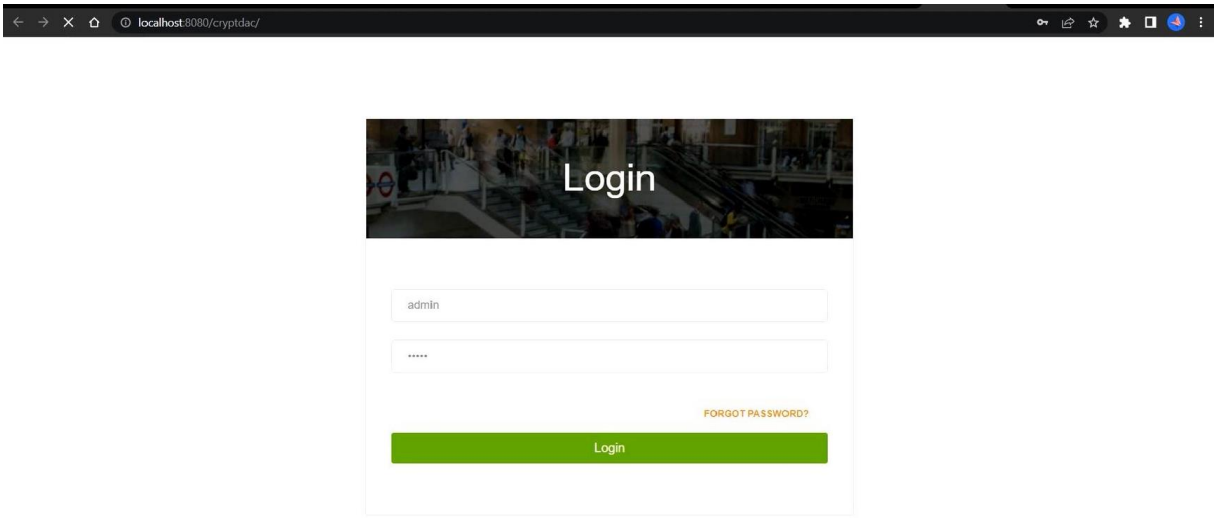
[37] 2019. [Online]. Available: https://www.cryptopp.com/

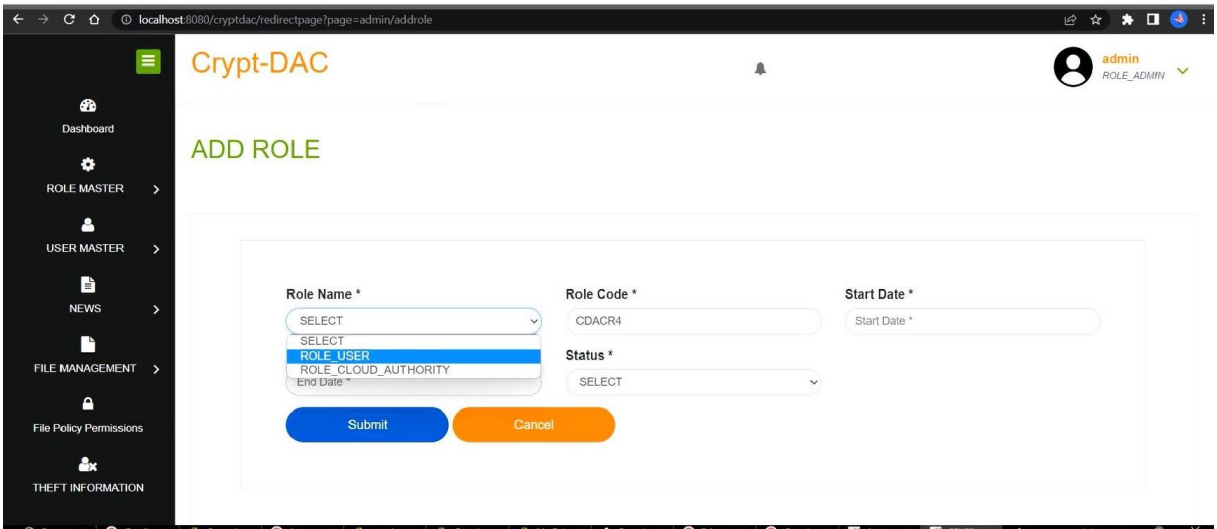
[38] W. C. Garrison III , A. J. Lee, and T. L. Hinrichs, “An actor-based, application-aware access control evaluation framework,” in Proc. 19th ACM Symp. Access Control Models Technol., 2014, pp. 199–210.

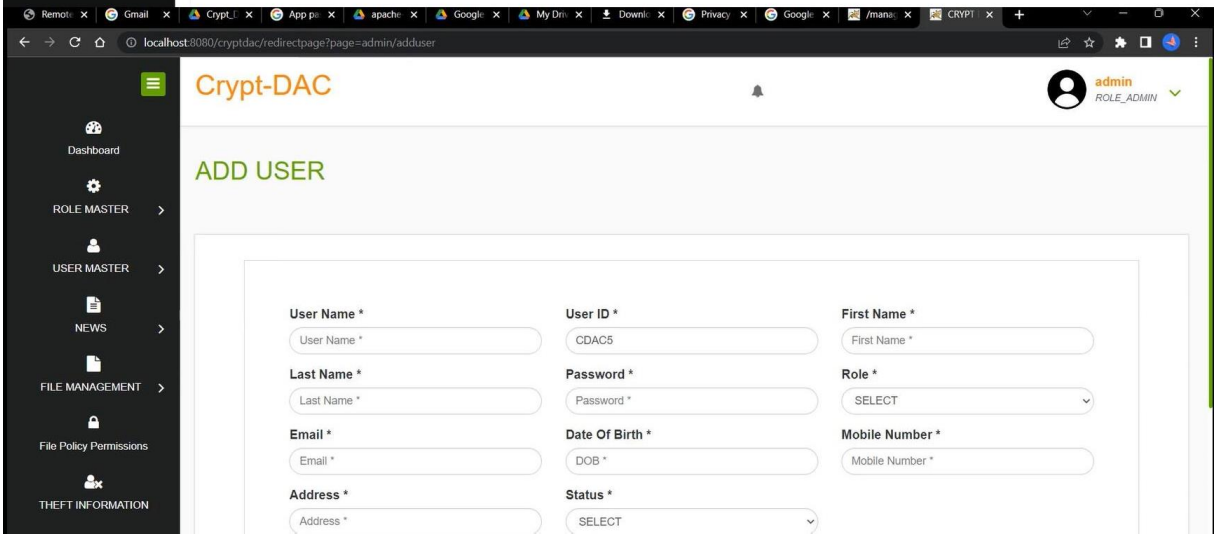
[39] G. Ateniese, D. H. Chou, B. Medeiros, and G. Tsudik, “Sanitizable Signatures,” in Proc. Eur. Symp. Res. Comput. Security, 2005, pp. 159–177

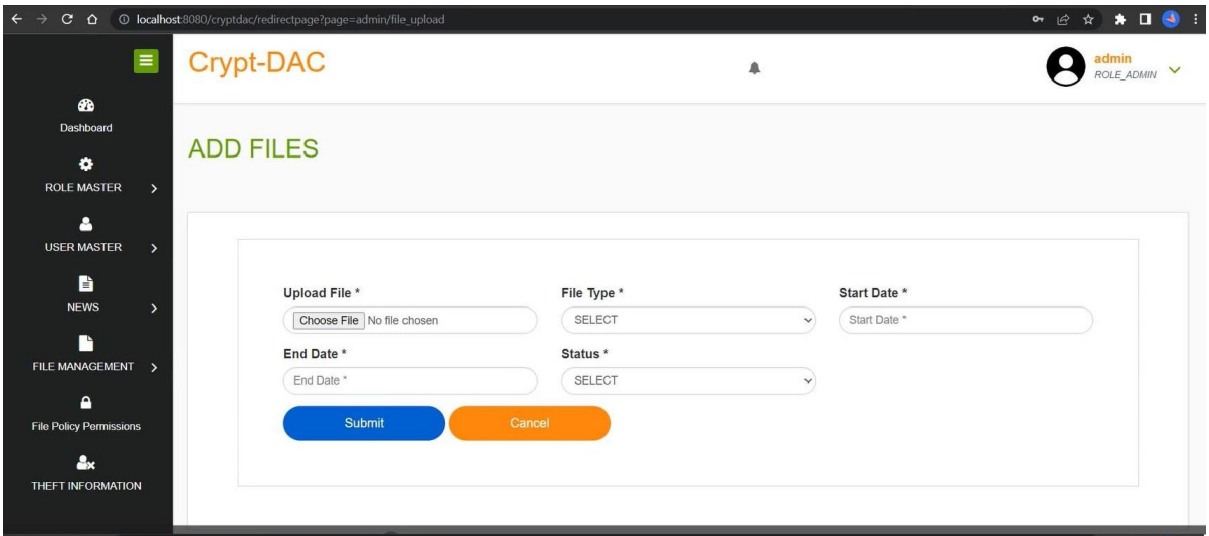
APPENDIX

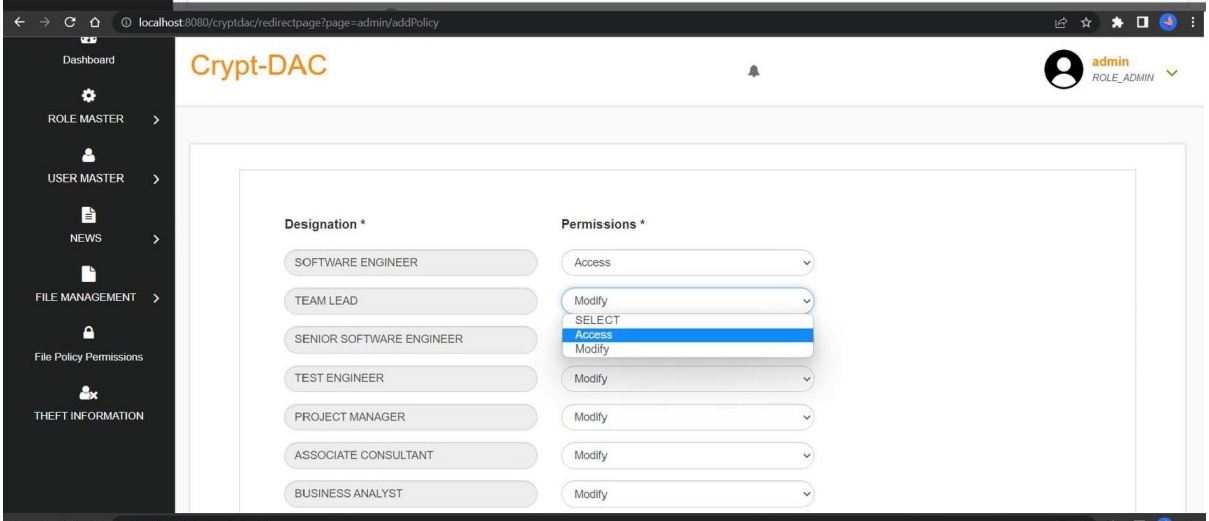
A. SCREEN SHOTS

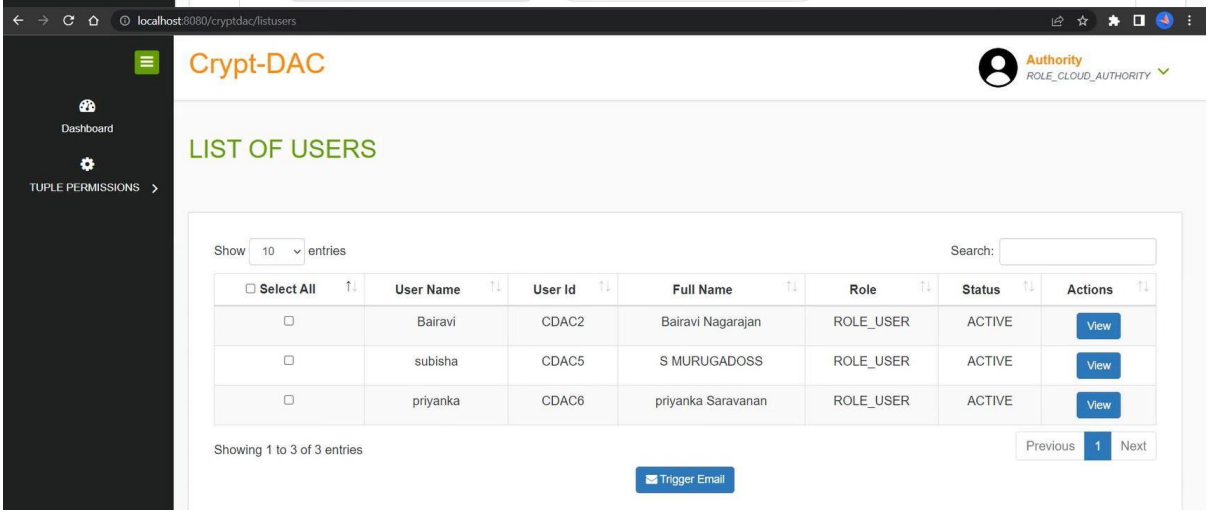


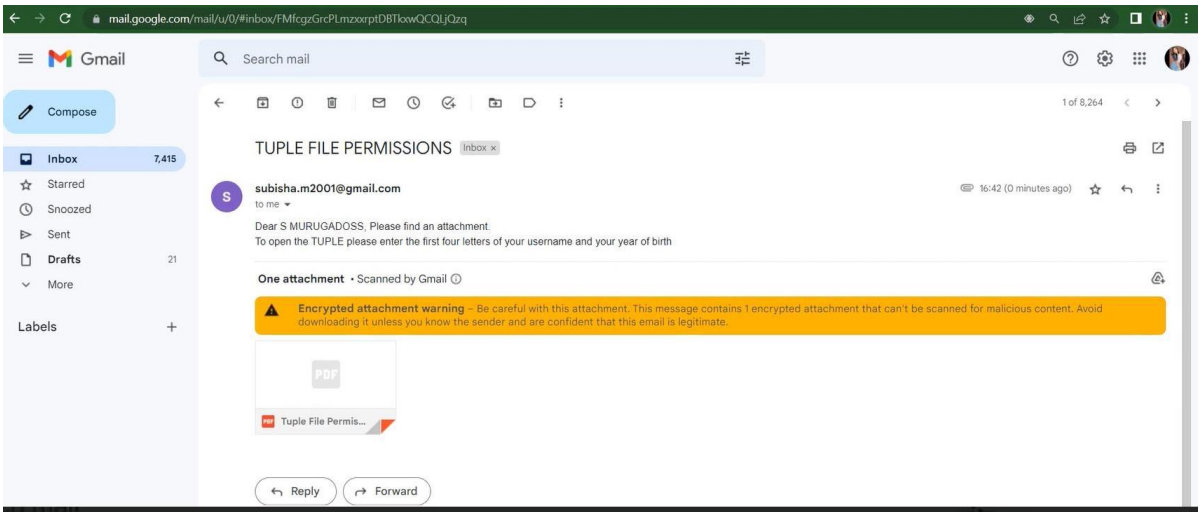


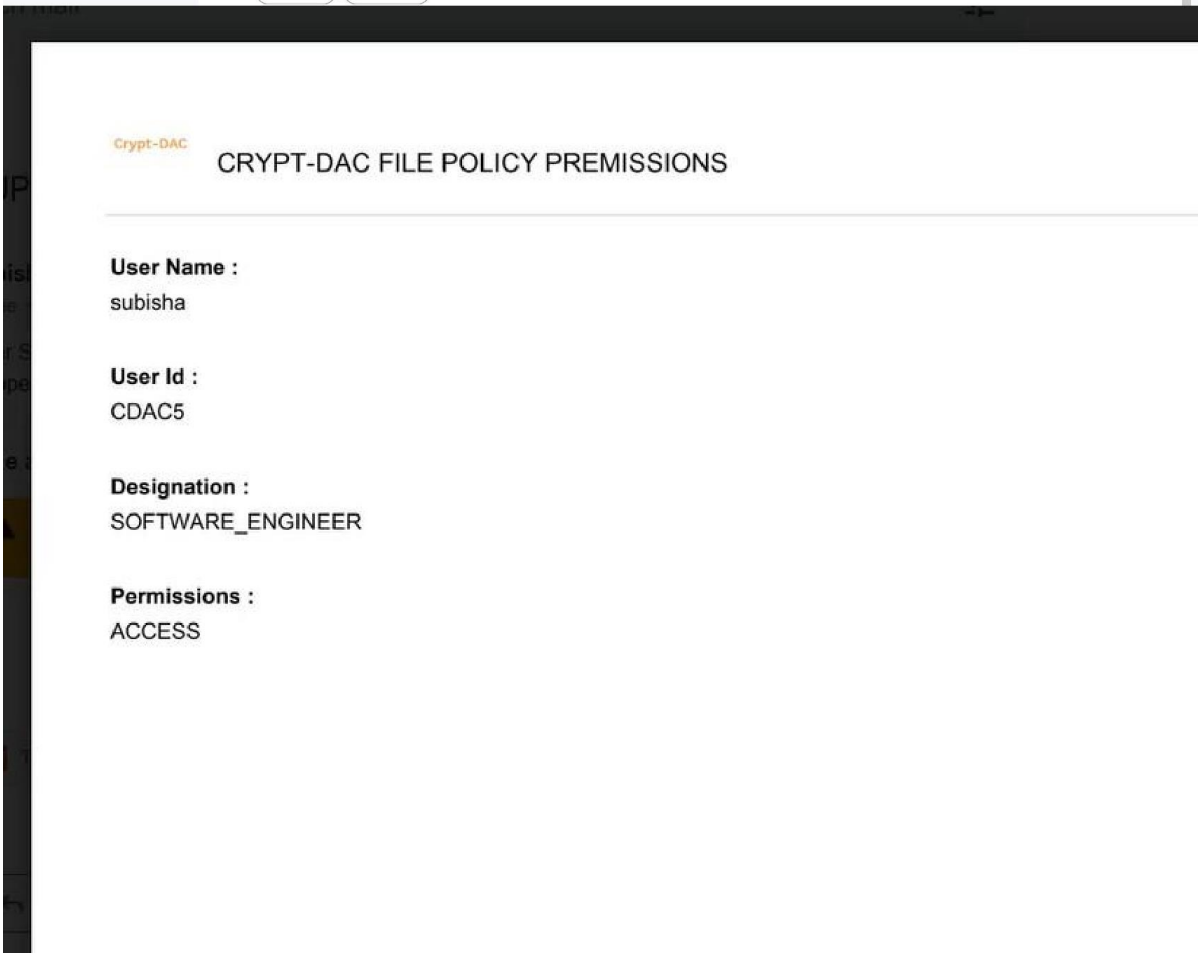


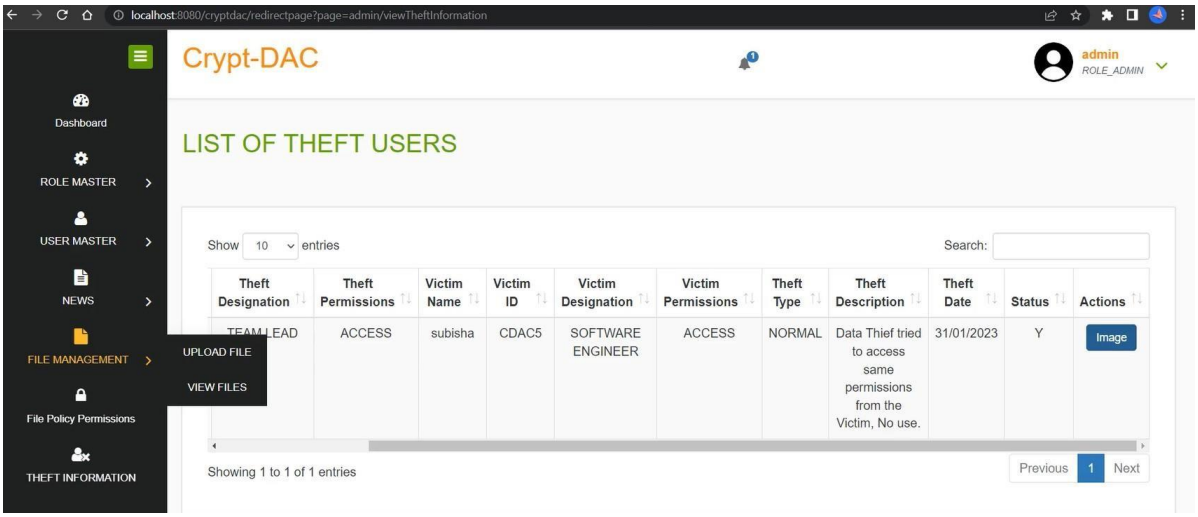

localhost:8080/cryptdac/redirectpage?page=admin/file_upload
Crypt-DAC
admin
ROLE_ADMIN
Dashboard
ROLE MASTER
USER MASTER
NEWS
FILE MANAGEMENT
File Policy Permissions
THEFT INFORMATION
ADD FILES
Upload File *
Choose File
No file chosen
File Type *
SELECT
Start Date *
Start Date *
End Date *
End Date *
Status *
SELECT
Submit
Cancel


localhost:8080/cryptdac/redirectpage?page=admin/addPolicy
Crypt-DAC
admin
ROLE_ADMIN
Dashboard
ROLE MASTER
USER MASTER
NEWS
FILE MANAGEMENT
File Policy Permissions
THEFT INFORMATION
Designation *
Permissions *
SOFTWARE ENGINEER
Access
TEAM LEAD
Modify
SELECT
Access
Modify
SENIOR SOFTWARE ENGINEER
TEST ENGINEER
Modify
PROJECT MANAGER
Modify
ASSOCIATE CONSULTANT
Modify
BUSINESS ANALYST
Modify


localhost:8080/cryptdac/listusers
Crypt-DAC
Authority
ROLE_CLOUD_AUTHORITY
Dashboard
TUPLE PERMISSIONS
LIST OF USERS
Show 10 entries
Search:
Select All
User Name
User Id
Full Name
Role
Status
Actions
Bairavi
CDAC2
Bairavi Nagarajan
ROLE_USER
ACTIVE
View
subisha
CDAC5
S MURUGADOSS
ROLE_USER
ACTIVE
View
priyanka
CDAC6
priyanka Saravanan
ROLE_USER
ACTIVE
View
Showing 1 to 3 of 3 entries
Previous
1
Next
Trigger Email


mail.google.com/mail/u/0/#inbox/FMfcgzGrcPLmzxxrptDBTkxwQCQLjQzq
Gmail
Search mail
Compose
Inbox 7,415
Starred
Snoozed
Sent
Drafts 21
More
Labels
1 of 8,264
TUPLE FILE PERMISSIONS
Inbox ×
subisha.m2001@gmail.com
to me
16:42 (0 minutes ago)
Dear S MURUGADOSS, Please find an attachment.
To open the TUPLE please enter the first four letters of your username and your year of birth
One attachment • Scanned by Gmail
Encrypted attachment warning – Be careful with this attachment. This message contains 1 encrypted attachment that can't be scanned for malicious content. Avoid downloading it unless you know the sender and are confident that this email is legitimate.
PDF
Tuple File Permis...
Reply
Forward


Crypt-DAC
CRYPT-DAC FILE POLICY PREMISSIONS
User Name :
subisha
User Id :
CDAC5
Designation :
SOFTWARE_ENGINEER
Permissions :
ACCESS


localhost:8080/cryptdac/redirectpage?page=admin/viewTheftInformation
Crypt-DAC
admin
ROLE_ADMIN
Dashboard
ROLE MASTER
USER MASTER
NEWS
FILE MANAGEMENT
UPLOAD FILE
VIEW FILES
File Policy Permissions
THEFT INFORMATION
LIST OF THEFT USERS
Show 10 entries
Search:
Theft Designation
Theft Permissions
Victim Name
Victim ID
Victim Designation
Victim Permissions
Theft Type
Theft Description
Theft Date
Status
Actions
TEAM LEAD
ACCESS
subisha
CDAC5
SOFTWARE ENGINEER
ACCESS
NORMAL
Data Thief tried to access same permissions from the Victim, No use.
31/01/2023
Y
Image
Showing 1 to 1 of 1 entries
Previous
1
Next

B RESEARCH PAPER:

DYNAMIC ACCESS CONTROL THROUGH CRYPTOGRAPHY IN CLOUD

Subisha.M
School of Computing
Sathyabama Institute Of Science and Technology(UGC)
Chennai,India
subisha.m2001@gmail.com

Priyanka.S.S
School of Computing
Sathyabama Institute of Science and Technology(UGC)
Chennai,India
sspriyankasaravanan@gmail.com

Dr.Sankari.M
School of Computing
SathyabamaInstitue Of Science and Technology(UGC)
Chennai,India
sankari.cse@sathyabama.ac.in

Abstract*— In recent days, encrypting the data in the untrusted cloud with access control is necessary for many users and organizations. However, it is a challenging thing to design a efficient access control. This paper proposes DAC in cryptography. It is a viable encryption scheme for dynamic access control. It encrypts the user's file using a symmetric key list that includes a file key as well as a set of lock keys. The administrator must upload a new revocation key to encrypt the file after each revocation and update the encryption key list with the current encryption level. As a result, cryptography is used to establish dynamic access control. This does not necessitate costly decryption/re-encryption or the uploading/reuploading massive amounts of data. It provides security by instantly cancelling permissions. As a result, efficiency improves. DAC proposes three key strategies in cryptography to minimize the size of the key list and encryption layer.

***Keywords*—** *dynamic access control, key list, cryptography, revocation, encryption, data, cloud, decryption, encryption layer*.

1. Introduction

In cryptography, DAC encrypts data using a symmetric key list that stores a file key as well as a set of lock keys. Each revocation necessitates the upload of a new revocation key to the cloud, the encryption of data with the new encryption level, and the updating of the encryption key list. Because of tremendous improvements in cloud computing, people and organizations are becoming increasingly interested in storing and sharing data via cloud services. Amazon, Microsoft, Apple, and other cloud service companies provide a wide range of cloud-based services, ranging from modest personal services to big industrial services. Recent data breaches, such as the publication of private images, have increased worries about the protection of cloud-managed data.

Indeed, cloud service providers are frequently insecure due to defects in programmed architecture and system vulnerabilities. As a result, how to impose data access control in potentially untrustworthy clouds is critical. In response to these security concerns, a number of works have been proposed that use cryptographic primitives to control access to untrusted cloud services. Many access control paradigms are enforced using sophisticated cryptographic primitives. Attribute-based encryption (ABE), for example, is the cryptographic equivalent of the attribute-based access control (ABAC) approach. Previous research, on the other hand, has mostly focused on static environments in which access control regulations rarely change. If you need to change the access control policy, the prior work will be extremely time-consuming.

You appear to be able to revoke a user's permissions by denying them access to the keys used to encrypt their files. This technique, however, is not secure because the user can store a local copy of the key before cancelling it. To avoid similar issues, re-encrypt the files with a fresh key. This necessitates the file's owner downloading the file, re-encrypting it, and uploading it again so that the cloud may update the previously encrypted file.

As previously mentioned, the first scheme requires the administrator to re-encrypt the file with a new key. This technique has a high communication overhead. The second technique, on the other hand, delegated the file to be re-encrypted when the user needed to update it, so that the administrator did not have to re-encrypt the file or data itself. This technique, however, has a security flaw in that it delays the undo procedure until the user next updates the file. The freshly revoked user now has access to the file until the next write. Another locking approach that encrypts files using a symmetric homomorphic encryption scheme was also proposed. With such a system, the cloud can directly re-encrypt files without first decrypting them. Because the overhead of the encryption/decryption procedure is comparable to that of public-key cryptography, this technique incurs expensive file read/write overhead. This Overcome untrusted cloud access control systems.

In cryptography, DAC provides three major techniques: First, DAC provides a delegation-enabled encryption technique in cryptography that delegates policy data updates to the cloud. In the case of files, the administrator adds the new lock key to the end of the key list and requests that the cloud update the key list in the policy data. However, when revocation actions are performed, the size of the key list

grows, requiring users to download and decode enormous key lists each time they access a file. To avoid this problem, we have enabled administrators to set a file limit. Once the encryption layer has reached its maximum size, it cannot be expanded by outsourcing encryption activities to the cloud. In cryptography, DAC presents a delayed deionization encryption approach that uses write operations to update a file's symmetric key list and remove the limiting encryption layer above it.

The next user who writes to the file, in particular, encrypts the write content with a new symmetric key list containing only the new file key and updates the key list in the policy data. DAC in cryptography eliminates the limited layer of encryption from files using this method, making it easy for many writers. Overall, DAC in cryptography achieves efficient locking, efficient file access, and immediate locking all at the same time. DAC in cryptography incurs relatively minimal communication cost on the part of administrators due to revocation efficiency, as file data does not need to be downloaded and re-uploaded. When files are re-encrypted, user permissions are immediately removed with immediate revoke. For efficient file access, files are still encrypted with a symmetric key. Ali cloud has implemented DAC in cryptography and some recent work. Practical experiments suggest that DAC in cryptography is three orders of magnitude more efficient in communicating access revocations and almost two orders of magnitude more efficient in computing file accesses than the first scheme. Finally, compared to the second scheme, DAC in cryptography DAC can revoke permissions instantly. To solve these problems, we introduce DAC in cryptography.To address these issues, we bring DAC into cryptography.

2 Literature Survey

This problem statement has been extensively studied over the past 12 years by researchers.

Vipul Goyal, Abhishek Jain, and Omkant Pandey described an attribute-based encryption system for ciphertext policy,in which a user's private key is associated with a set of attributes.A user can decrypt only if the user's attribute satisfies the ciphertext policy. Their safety case is based on the standard assumption Decisional Bilinear Different Hellman.

Dr.Ragesh G.K.a and Dr. K. Baskaranb proposed encrypted data access control in personal health record (PHR) systems, as they play an important role in the digital transformation of healthcare. These systems offer many value-added features such as viewing health-related information, securely transmitting and tracking information with healthcare providers. Patients no longer have physical access to their health data stored on cloud servers, so each patient must encrypt their PHR data before uploading it to the cloud. Additionally, achieving fine-grained data access control of her encrypted PHR data in an effective and scalable manner is a difficult task. The proposed scheme inherits flexibility, scalability, and fine-grained patient-centric data access control.

Xiaoguang Wang and Yong Qi proposed the design and implementation of Sec Pod, a framework for virtualization-based security systems. This is because the operating system kernel is critical to computer system security. Many systems have been proposed to improve security. A fundamental weakness of these systems is that the page tables, the data structures that control memory protection, are not isolated from the vulnerable kernel and can be tampered. They implemented a prototype of Sec Pod based on KVM. Their experiments show that Sec Pod is effective and efficient.

Sascha Müller and Stefan Katzenbeisser proposed Hiding the Policy in Cryptographic Access Control that recently, cryptographic access control has received a lot of attention, mainly due to the availability of efficient Attribute-Based Encryption (ABE) schemes. With ABE, you can get rid of your trusted reference monitor by encrypting and enforcing access rules. However, ABE has privacy issues. Access policies are sent unencrypted with ciphertext. Further generalizing the idea of policy-hiding in cryptographic access control, they introduce policy anonymity where similar to the well-understood concept of k-anonymity the attacker can only see a large set of possible policies that might have been used to encrypt, but is not able to identify the one that was actually used. They show that using a concept from graph theory we can extend a known ABE construction to achieve the desired privacy property.

Keywords: access control, privacy, tree majors, abe, anonymity, hidden policies

Xiaofeng Chen, Jin Li and Xinyi Huang proposed New Publicly Verifiable Databases with Efficient Updates that the notion of verifiable database (VDB) enables a resource-constrained client to securely outsource a very large database to a un trusted server so that it could later retrieve a database record and update it by assigning a new value. Also, any attempt by the server to tamper with the data will be detected by the client. Very recently, Catalano and Fiore proposed an elegant framework to build efficient VDB that supports public verifiability from a new primitive named vector commitment. This whitepaper points out that Vector Commitment's Catalano-Fiore VDB framework is vulnerable to the so-called Forward Automatic Update (FAU) attack. They also propose a new vector commitment VDB framework based on the idea of commitment bindings. The structure is not only publicly verifiable, but also secure against FAU attacks.

3 Existing System

The CP-ABE may assist us prevent security breaches from outside attackers in our current system. But, if an organisation's insider is accused of committing "crimes" including the redistribution of decryption privileges and the circulation of user information in plain format for illegal financial gain, how can we be certain that the insider is guilty? Is it feasible for us to withdraw the compromised access privileges as well? In addition to the questions listed above, we have one more concerning key generation

authority. A semi-trusted authority often issues a cloud user's access credential (i.e., decryption key) depending on the attributes the user holds. How can we be certain that the generated access credentials will not be (re-)distributed to others?

3.1 Open problems in Existing System

1. This scheme causes a considerable communication overhead.
2. However, this scheme has a security drawback because it delays the undo process until the user next modifies the file. This allows the newly revoked user to continue accessing the file before the next write operation.

4 PROPOSED SYSTEM

To solve the difficulty of credential exposure in CP-ABE-based cloud storage systems, an accountable authority and revocable crypto cloud that supports white-box traceability and auditing has been designed in this study. This is the first CP-ABE-based cloud storage system that enables white-box traceability, accountability, auditing, and effective lockdown all at the same time. A cryptographically enforced dynamic access control mechanism on untrusted clouds is known as DAC. When permissions are withdrawn, DAC in cryptography delegates updating encrypted files to the cloud. When user credentials are disseminated from semi-trusted authorities, this strategy can also be used. The purpose is to deliver file data housed in the cloud with secrecy and access control.

Confidentiality: We do not exchange decryption keys with the cloud. This safeguards the privacy of your file data.

Read Access Control: To implement access control, our system employs encryption, ensuring that users can only read file data that corresponds to their permissions.

Write Access Control: To enforce write permissions, this system relies on cloud provider validation.

5 IMPLEMENTATION

Several algorithms were employed in this research. They are as follows: AES Algorithm: The AES encryption algorithm (also known as the Rijndael algorithm) is a 128-bit symmetric block encryption technique. Use 128-, 192-, and 256-bit keys to transform these individual blocks. We weave these encrypted blocks together to generate the ciphertext.

The RSA Algorithm: The most widely used public key algorithm is RSA, which is called after its creators, Rivest, Shamir, and Adelman (RSA). It is a popular public-key cryptosystem for secure data transmission.

This project requires the following software: Windows 10, JDK1.8, J2EE, Tomcat7.0, and MySQL. This project's hardware requirements are as follows: Hard disc: 80GB or more, RAM: 4GB or greater, Processor: PIV or greater.

Modules:

1. Organization profile creation & Key Generation
2. Data Owners File Upload
3. File Permission & Tuple Creation
4. Tracing who is guilty

1. Organization profile creation & Key Generation:

On the web end, the user goes through the initial registration process. This process requires the user to enter personal information. The server saves data in a database. An Accountable STA (Semi-Trusted Authority) produces a user's decryption key depending on a set of attributes provided by the user (name, email ID, contact number, and so on). The user acquires an origin to access organizational data after acquiring the decryption key from the Accountable STA.

2. Data Owners File Upload:

In this module data owners create their accounts under the public cloud and upload their data into public cloud. While uploading the files into public cloud data owners will encrypt their data using RSA Encryption algorithm and generates public key and secret key. And generates one unique file access permission key for the users under the organisation to access their data.

3. File Permission & Tuple Creation:

Separate data owners establish separate file permission keys for their files and distribute these keys to organization users to allow access to their files. It also creates policy files that govern which data can be accessed. The keys for reading files, writing files, downloading files, and deleting files are separated in the policy file.

4. Tracing who is guilty:

Offloaded data can be accessed (read, written, downloaded, deleted, decrypted, and so on) by authorised DUs. This is where file authorization keys are assigned to employees depending on their experience and position within the organisation in relation to the registered individual. Senior staff have complete access to files (read, write, delete, and download). Fresher has only read-only access to files. Some employees have read/write access. Furthermore, some employees have all permissions save the ability to delete data. If a senior employee discloses or shares a private authentication key with a junior employee, the junior employee will be asked to download or destroy the data of the data owner. The system will generate the role's attribute set in the background when the user enters the re-encryption password. They are guilty when their attribute set does not match the policy file of the data owner. If you ask them, they will know who provided the keys to the others.

6 ARCHITECTURE DIAGRAM

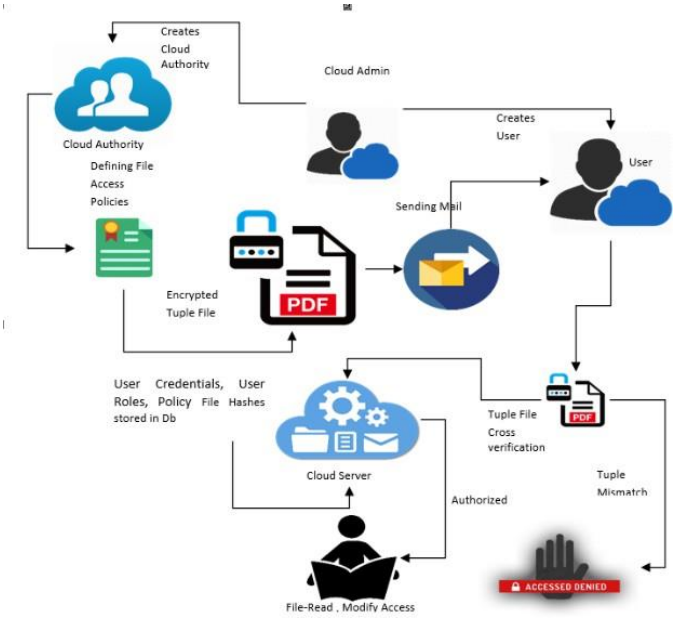


Fig.1.system architecture of DAC in cryptography

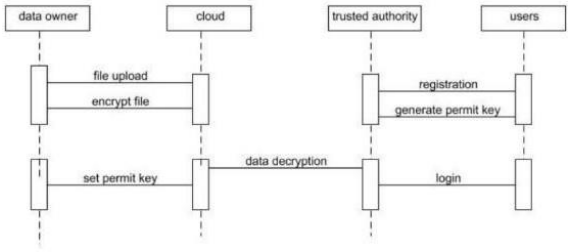


Fig.2. sequence diagram of DAC in cryptography

7 RESULT AND DISCUSSION

This is the engine that allows for the effective encryption enforcement of dynamic access management within untrusted cloud publishers. In cryptography, DAC realizes the dream of combining the three approaches. Allow cloud delegation to substitute privacy coverage information while using delegation-aware encryption mechanisms. This is commonly used to avoid fancy re-encryption in reports. Admin information covers the use of the tune able union encryption technique. Furthermore, the non-bonded encryption method is running behind schedule in order to avoid the overhead of reviewing reports. A theoretical examination and an overall performance evaluation reveal that, when compared to earlier methods, DAC in cryptography not only ensures the same protection as the honest but inquisitive variant, but also allows for revocation. It shows that it achieves more significant performance in terms of other schemes.

In [1], The architecture in is designed to provide security by monitoring and protecting virtual machines in real time. However, the rogue VM's attack behavior required further investigation. Malicious behaviors are examined in our article, and any visit by a malicious user to cloud-hosted data is recognized. This was accomplished through the use of white box traceability.

In [2] Security and privacy in remote cloud storage had been improved. To encrypt data on the cloud without downloading any data, an attribute-based d proxy re-encryption approach was presented. This study proposes a new revocation mechanism that allows the cloud to simply re-encrypt the file without decryption. For all communication overhead problems, DAC in cryptography is the best choice for revocation.

In 2019, Garrison presented a revocation approach with a security penalty because the revocation action will be postponed until the next user modifies the data.This has been addressed in this work through the use of Access Based Enumeration.Previous papers have always included a security penalty with revocation methods. However, this study advocates for effective revocation. The proposed approach delegated updating the encrypted files in permission revocations to the cloud.

8 CONCLUSIONS

We proposed DAC in cryptography, a system that allows dynamic access control in untrusted cloud providers using encryption. To achieve the purpose of DAC in cryptography, three strategies are used. To avoid costly re-encryption of file data on the admin side, we propose using a customizable onion encryption method. To avoid file read overhead, we developed a delayed de-onion encryption approach. In cryptography, DAC achieves orders of magnitude more blocking efficiency. It also ensures the same security features as prior models under an honest threat model.

References

[1] X. Wang, Y. Qi, and Z. Wang, “Design and implementation of SecPod: A framework for virtualization-based security systems,” IEEE Trans. Depend. Secure Comput., vol. 16, no. 1, pp. 44–57, Jan./ Feb. 2019.

[2] F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan, “Sieve: Cryptographically enforced access control for user data in untrusted clouds,” in

Proc. USENIX Symp. Netw. Syst. Des. Implementation, 2016, pp. 611–626.

[3] S. Muller and S. Katzenbeisser, “Hiding the policy in cryptographic access control,” in Proc. Int. Workshop Security Trust Manage., 2011, pp. 90–105.

[4] X. Jin, R. Krishnan, and R. S. Sandhu, “A unified attribute-based access control model covering DAC, MAC and RBAC,” in Proc. 26th Annu. IFIP WG 11.3 Conf. Data Appl. Security Privacy, 2012, pp. 41–55.

[5] W. C. Garrison III , A. Shull, S. Myers, and, A. J. Lee, “On the practicality of cryptographically enforcing dynamic access control policies in the cloud,” in Proc. IEEE Symp. Security Privacy, 2016, pp. 819–838.

[6] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, “Verifiable auditing for outsourced database in cloud computing,” IEEE Trans. Comput., vol. 64, no. 11, pp. 3293–3303, Nov. 2015.

[7] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, “Secure and efficient cloud data deduplication with randomized tag,” IEEE Trasactions Inform. Forensics Security, vol. 12, no. 3, pp. 532–543, Mar. 2017.

[8] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, “Towards achieving flexible and verifiable search for outsourced database in cloud computing,” Future Generation Comput. Syst., vol. 67, pp. 266–275, 2017.

[9] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, “New publicly verifiable databases with efficient updates,” IEEE Trans. Depend. Secure Comput., vol. 12, no. 5, pp. 546–556, Sep.-Oct. 2015.

[10] M. Maffei, G. Malavolta, M. Reinert, and D. Schroder, “Privacy and access control for outsourced personal records,” in Proc. IEEE Symp. Security Privacy, 2015, pp. 341–358.

[11] D. Boneh, K. Lewi, H. Montgomery, and A. Raghu Raghunathan, “Key homomorphic PRFs and their applications,” in Proc. Annu. Cryptology Conf., 2013, pp. 410–428.

[12] J. R. Lorch, B. Parno, J. W. Mickens, M. Raykova, and J. Schiffman, “Shroud: Ensuring private access to large-scale data in the data center,” in Proc. USENIX Conf. File Storage Technol., 2013, pp. 199–214.

[13] L. Ibraimi, “Cryptographically enforced distributed data access control,” Ph.D. dissertation, Electrical Engineering, Mathematics and Computer Science, Univ. Twente, Enschede, Netherlands, 2011.