|  |  |  |
| --- | --- | --- |
| **Tech Saksham**  Final Project Report  **Cloud Computing** |  |  |

**“Protection based cloud service access control”**

**“Sree Chaitanya institute of technological sciences”**

|  |  |
| --- | --- |
| **ROLL NO** | **NAME** |
| 19TR1A0421 | Santhoshi Rupa Chevula |
| 19TR1A0427 | Shivani Dasari |
| 19TR1A0425 | Shivani Chukka |

|  |  |
| --- | --- |
|  |  |
|  | Mr. S.S. Ahmed Ali |
|  | Master Trainer |

**ABSTRACT**

With the rapid development of the computer technology, cloud-based services have become a hot topic. Cloud based services not only provide users with convenience, but also bring many security issues. Therefore, the study of access control scheme to protect users' privacy in cloud environment is of great significance. In this paper, we present an access control system with privilege separation based on privacy protection (PS-ACS). In the PS-ACS scheme, we divide the users into personal domain (PSD) and public domain (PUD) logically. In the PSD, we set read and write access permissions for users respectively. The Key-Aggregate Encryption (KAE) is exploited to implement the read access permission which improves the access efficiency. A high degree of patient privacy is guaranteed simultaneously by exploiting an Improved Attribute-based Signature (IABS) which can determine the users’ write access. For the users of PUD, a hierarchical attribute-based encryption (HABE) is applied to avoid the issues of single point of failure and complicated key distribution. Function and performance testing result shows that the PS-ACS scheme can achieve privacy protection in cloud based services.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Table of Contents** | **Page No.** |
| 1 | Chapter 1: Introduction | 4 |
| 2 | Chapter 2: Services and Tools Required | 10 |
| 3 | Chapter 3: Project Architecture | 11 |
| 4 | Chapter 4: Architecture Blocks Detail Working | 19 |
| 5 | Conclusion | 20 |
| 6 | References | 21 |
| 7 | Code | 25 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview**

Protection-based cloud service access control is a security mechanism that enables cloud service providers to protect their resources and data from unauthorized access and ensure that only authorized users can access their services. It involves the implementation of access control policies that govern the access to cloud services and resources.

There are several techniques that can be used to implement protection-based cloud service access control, including:

**Role-based access control (RBAC):** This technique involves assigning roles to users and granting permissions based on their roles. For example, a user with the role of "administrator" may have full access to all resources, while a user with the role of "guest" may have limited access.

**Attribute-based access control (ABAC):** This technique involves defining access control policies based on attributes of the user, such as their job title or department, and the resource they are trying to access. For example, a user with the job title "HR Manager" may have access to employee records, while a user with the job title "Marketing Manager" may have access to customer data.

**Mandatory access control (MAC):** This technique involves assigning security labels to resources and users and enforcing access control policies based on those labels. For example, a user with a "top secret" security clearance may only be able to access resources labeled as "top secret."

**Discretionary access control (DAC):** This technique involves allowing the resource owner to determine who has access to their resources. For example, a user who creates a document may be able to grant or revoke access to that document for other users.

**1.2 Feature**

The following are some of the key features of protection-based cloud service access control:

**Authentication:** Protection-based cloud service access control requires authentication of users before they can access cloud services or resources. Authentication involves verifying the identity of a user using credentials such as usernames, passwords, or biometric information.

**Authorization:** Once a user is authenticated, protection-based cloud service access control determines whether the user is authorized to access a particular resource or service. Authorization involves granting or denying access to resources based on the user's identity, role, attributes, or other factors.

**Role-based access control (RBAC):** RBAC is a key feature of protection-based cloud service access control. It involves assigning roles to users and granting permissions based on those roles. This simplifies access control management and allows for more granular access control.

**Attribute-based access control (ABAC):** ABAC is another key feature of protection-based cloud service access control. It involves defining access control policies based on attributes of the user, such as their job title, location, or department, and the resource they are trying to access. This provides more fine-grained access control than RBAC.

**Audit trail:** Protection-based cloud service access control often includes an audit trail that records access attempts, including successful and unsuccessful attempts. This helps to identify security breaches and provides a record of who has accessed a resource or service.

In summary, protection-based cloud service access control includes authentication, authorization, RBAC, ABAC, MAC, DAC, and audit trails. These features work together to provide a secure and flexible access control mechanism for cloud resources and services.

**1.3 Advantages**

The advantages of protection-based cloud service access control include:

**Increased security:** Protection-based cloud service access control ensures that only authorized users have access to cloud resources and services. This reduces the risk of unauthorized access and data breaches.

**Flexibility:** Protection-based cloud service access control offers flexibility in managing access to cloud resources and services. It allows for granular access control based on user roles, attributes, and other factors.

**Simplified management:** Protection-based cloud service access control simplifies access management by allowing for centralized control of access policies. This reduces the need for manual access control management and improves compliance with security policies.

**Improved compliance:** Protection-based cloud service access control helps organizations to comply with regulatory and industry standards. It provides an audit trail of access attempts, which can be used to demonstrate compliance with security policies.

**Cost-effective:** Protection-based cloud service access control can be cost-effective compared to traditional on-premises access control systems. It reduces the need for hardware and software installations and maintenance.

**Better resource utilization:** Protection-based cloud service access control helps to prevent unauthorized access to resources, which can lead to better resource utilization. It reduces the risk of resource wastage and improves the overall efficiency of cloud services.

**1.4 Future Work**

The future work of protection-based cloud service access control is focused on addressing the evolving security threats in cloud environments and improving the user experience. Some of the potential areas of future work include:

**Advanced threat detection:** Protection-based cloud service access control needs to evolve to detect and prevent advanced threats such as zero-day exploits, malware, and insider threats. This requires incorporating machine learning and artificial intelligence (AI) into access control systems to detect abnormal user behavior and suspicious activity.

**Integration with identity and access management (IAM):** Protection-based cloud service access control needs to integrate more seamlessly with IAM systems to provide a holistic approach to access control management. This includes integrating with identity providers such as Active Directory and LDAP to simplify user provisioning and deprovisioning.

**Cloud-native access control:** Cloud-native access control solutions that are purpose-built for cloud environments are becoming increasingly important. These solutions need to be lightweight, scalable, and provide seamless integration with cloud services such as AWS, Azure, and Google Cloud Platform.

**Better user experience:** Protection-based cloud service access control needs to provide a better user experience that balances security with usability. This includes improving the user interface, providing more granular access control options, and reducing the number of authentication prompts required.

**Compliance with new regulations:** Protection-based cloud service access control needs to comply with new regulations such as GDPR, CCPA, and HIPAA. This requires incorporating privacy controls into access control systems and providing robust audit trails that can be used to demonstrate compliance.

**CHAPTER 2**

**SERVICES AND TOOLS REQUIRED**

**2.1 Services Used**

Protection-based cloud service access control involves using various services and technologies to ensure that only authorized users have access to cloud resources and services. Some of the services used for protection-based cloud service access control include:

* **Identity and Access Management (IAM):** IAM services are used to manage user identities and their access to cloud resources and services. IAM services provide capabilities such as user authentication, user provisioning and deprovisioning, and access control policies.
* **Cloud Access Security Brokers (CASB):** CASB services are used to monitor and control access to cloud services and resources. CASB services provide capabilities such as access control, policy enforcement, and threat detection.
* **Multi-Factor Authentication (MFA):** MFA services are used to enhance user authentication by requiring additional factors such as SMS-based verification codes, biometric authentication, or hardware tokens.
* **Security Information and Event Management (SIEM):** SIEM services are used to monitor and analyze security events in real-time. SIEM services provide capabilities such as log collection, correlation, and reporting.
* **Privileged Access Management (PAM):** PAM services are used to manage and monitor access to privileged accounts. PAM services provide capabilities such as password management, session recording, and activity monitoring.
* **Encryption services:** Encryption services are used to protect data in transit and at rest. Encryption services provide capabilities such as key management, data encryption, and decryption.
* **Firewall services:** Firewall services are used to protect cloud resources and services from unauthorized access. Firewall services provide capabilities such as network filtering, traffic monitoring, and policy enforcement.

**2.1.1 Liberty Profile**

Protection-based cloud service access control is a security mechanism that is used to control access to cloud-based services by enforcing policies based on the protection level of the data being accessed. This approach ensures that only authorized users are granted access to sensitive data, and that the data is protected from unauthorized access, disclosure, or modification.

The Liberty profile provides a number of features and tools that can be used to implement protection-based cloud service access control. Some of these include:

* **Role-based access control:** The Liberty profile includes support for role-based access control (RBAC), which allows you to define roles for users and then assign permissions based on those roles. This can be used to enforce access policies based on the protection level of the data being accessed.
* **Security APIs:** The Liberty profile provides a set of security APIs that can be used to implement custom security policies and access control mechanisms. These APIs can be used to enforce protection-based access control policies based on the sensitivity of the data being accessed.
* **Authentication and authorization:** The Liberty profile includes support for a wide range of authentication and authorization mechanisms, including OAuth, OpenID Connect, and SAML. These mechanisms can be used to enforce protection-based access control policies based on the identity and role of the user requesting access to the data.
* **Transport layer security:** The Liberty profile includes support for transport layer security (TLS), which can be used to secure communications between cloud services and their users. This can help to prevent unauthorized access to sensitive data by encrypting data in transit.

**2.2 Tools and Softwares used**

**HARDWARE REQUIREMENTS:**

* System : Pentium Dual Core.
* Hard Disk : 120 GB.
* Monitor : 15’’ LED
* Input Devices : Keyboard, Mouse
* Ram : 1 GB

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows 7.
* Coding Language : JAVA/J2EE
* Tool : Netbeans 7.2.1
* Database : MYSQL

**2.2.1 NodeJS**

Node.js is a popular open-source server-side runtime environment for JavaScript that can be used to build scalable and high-performance web applications. While Node.js itself is not specifically designed for protection-based cloud service access control, it can be used to implement various security mechanisms and access control policies in a cloud environment.

Node.js can be used to implement authentication and authorization mechanisms, such as OAuth and JSON Web Tokens (JWTs). These mechanisms can be used to control access to cloud resources based on the identity of the user and their role.

Node.js provides built-in cryptographic modules that can be used to encrypt and decrypt data in transit and at rest. This can help to protect sensitive data from unauthorized access.

Node.js can be used to create an API gateway that provides a unified interface for accessing cloud resources. This can help to enforce access control policies and rate limits for API requests.

**Middleware:** Node.js middleware can be used to implement custom security policies and access control mechanisms. Middleware can be used to intercept incoming requests and enforce policies based on the sensitivity of the data being accessed.

**Security Libraries:** Node.js has a rich ecosystem of security libraries that can be used to implement various security mechanisms, such as SSL/TLS, input validation, and SQL injection prevention.

Overall, Node.js can be a useful tool for implementing protection-based cloud service access control. By leveraging the various security features and tools available in Node.js, developers can create secure and scalable cloud applications that protect sensitive data from unauthorized access.

**2.2.2 HTML**

HTML (Hypertext Markup Language) is a markup language used to create web pages and web applications. It is a standard markup language that is used to structure and format content on the web. HTML is the foundation of all web development, and it works alongside CSS (Cascading Style Sheets) and JavaScript to create interactive and responsive web pages.

HTML is used to define the structure and content of web pages, using a series of HTML tags that are enclosed in angle brackets. HTML tags are used to specify headings, paragraphs, links, images, lists, tables, forms, and other elements that make up a web page.

Here are some examples of common HTML tags:

<html>: This tag indicates the start of an HTML document.

<head>: This tag contains information about the web page, such as the title and meta data.

<body>: This tag contains the main content of the web page.

<h1> to <h6>: These tags are used to define headings of various sizes.

<p>: This tag is used to define paragraphs of text.

<a>: This tag is used to define links to other web pages.

<img>: This tag is used to define images that are displayed on the web page.

<ul> and <ol>: These tags are used to define unordered and ordered lists, respectively.

<table>: This tag is used to define tables that display data.

<form>: This tag is used to define forms that collect user input.

HTML is a powerful and flexible language that can be used to create a wide range of web applications, from simple static websites to complex web-based applications. It is widely supported by web browsers and is constantly evolving to keep up with the latest trends in web development.

**2.2.3 Cloud Foundry**

Cloud Foundry is an open-source cloud application platform that is designed to simplify the deployment and management of cloud-native applications. It is a platform-as-a-service (PaaS) solution that provides developers with a flexible and scalable environment for building, deploying, and running cloud applications.

Cloud Foundry supports multiple programming languages, including Java, Ruby, Node.js, Go, and Python, and it provides a range of services and tools for managing applications in the cloud. Some of the key features of Cloud Foundry include:

1. **Multi-Cloud Support:** Cloud Foundry can be deployed on multiple cloud platforms, including AWS, Azure, Google Cloud, and VMware vSphere, providing developers with the flexibility to choose the cloud platform that best meets their needs.
2. **Application Deployment**: Cloud Foundry provides a simple and automated process for deploying applications to the cloud, with support for continuous delivery and integration.
3. **Application Scaling:** Cloud Foundry enables developers to easily scale their applications up or down to meet changing demands, without requiring any additional configuration.
4. **Service Broker Integration:** Cloud Foundry provides integration with service brokers, enabling developers to easily add third-party services to their applications, such as databases, messaging systems, and caching services.
5. **Logging and Metrics:** Cloud Foundry provides built-in logging and metrics capabilities, enabling developers to monitor and analyze the performance of their applications in the cloud.

Cloud Foundry is a popular choice for developers who are looking for a flexible and scalable platform for building cloud-native applications. Its open-source nature and wide range of features make it a powerful tool for organizations of all sizes, from startups to large enterprises.

**CHAPTER 3**

**PROJECT ARCHITECTURE**

**3.1 Architecture**

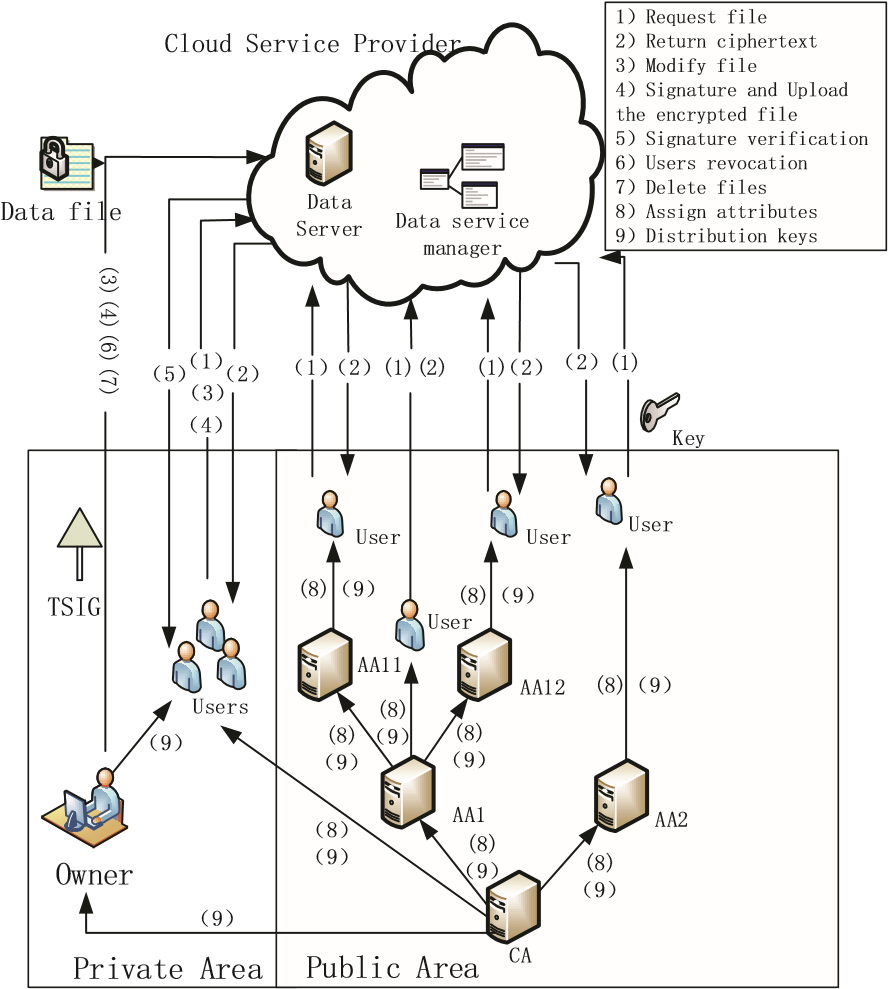


FIG: System Architecture

**CHAPTER 4**

**ARCHITECTURE BLOCKS DETAIL WORKING**

**4.1 Blocks**

As shown in above fig, our system model consists of Data owner, users in PSD, and users in PUD, root authority CA, regional authority AA and cloud service provider, which are defined as follows.

1. The cloud service provider consists of two parts: data storage server and data service management. Data storage server is responsible for storing confidential data files, and data service management is in charge of controlling external users’ access to secret data and returning the corresponding ciphertext.
2. In the actual cloud environment, CA manages multiple AA, and AA each manages attributes in their own field. The attributes owned by the user are issued by different authority.
3. Personal domain (PSD), in which users have special privileges, such as family, personal assistant, close friends and partners. This domain has a small number of users and small scale attributes, and the data owner knows the user's identity, which is easy to manage.
4. Public domain (PUD), which owns a huge number ofusers with unknown identity and a lot of attributes owned by the user.
5. Data Owner, based on the characteristics of users in public and personal domain to develop different access control strategy, encrypt uploaded files using the corresponding encryption method and then send to the cloud server.

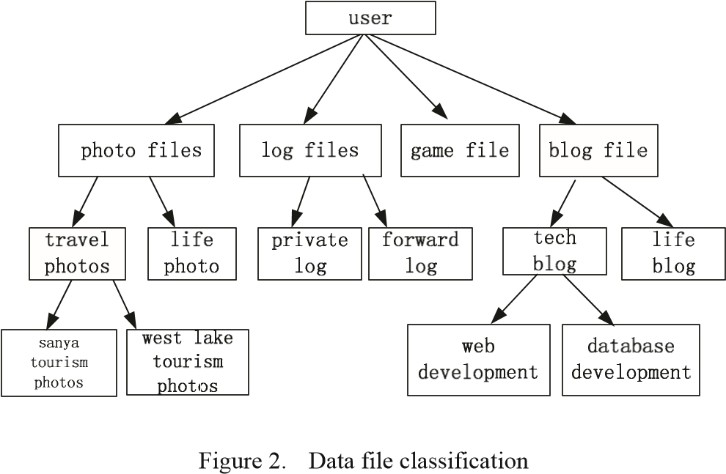
**ACCESS CONTROL SCHEME IN PSD**

## **Read Access Control**

The PSD has a small number of users, and their identities are known to the owner. In general, the data owner only wants the users to access or modify parts of data files, and different users can access and modify different parts of the data. For example, the blogger can allow his friend to browse part of his private photos; enterprises can also authorize employees to access or modify part of sensitive data. This requires the data owner to grant users read or write access permission to some data. In Chen’s MAH-ABE scheme, the CP-ABE is used to achieve the read access permission, but there are some defects to be considered. Firstly, since in the PSD, the users are all have a close relationship with the owner and the number is small, there is no need to use the CP-ABE which is applicable to the scenario which has a lot of users, and their identities are unknown to the owner, while the KAE scheme is set for the small users with certain identities. Besides, the distribution and management of keys and attributes, encryption and decryption process of CP-ABE are much more complex compared with the KAE scheme. Therefore, the KAE is exploited to implement the read access permission which improves the access efficiency.

Based on the above analysis, the paper uses the Aggregate Key Encryption scheme to encrypt the data files to realize different read access control. The specific application process of the KAE algorithm is as follows.

1. System setup and file encryption. The system first runs*Setup* of KAE to establish the public system parameter and master key. Each owner classified the file by its data attribute, such as “photo files”, “blog files” and “game files”. Fig.2 shows the way to classify the files. Choose and label the filesThen the owner’s client application runs Encrypt of KAE using the public key and the number of classification file to encrypt the PHR files and sends them to the cloud.
2. Access and key distribution. When the user send accessrequest to the cloud server, and his file index number is *i*, then the cloud server returns the corresponding encrypted classification file to the user. The owner authorized users access permission with the file index number denoted by *j* and sent the collection *S* of all the index number *j* to CA, CA generate an aggregate decryption key for a set of ciphertext classes via *Extract* of KAE and sent it to the corresponding user, Finally, any user with an aggregate key can decrypt any ciphertext whose class is contained in the aggregate key via *Decrypt* of KAE.



## **Write Access Control**

As Chen’s MAH-ABE scheme does not refer to the write access control, and in the PSD some cases exist, for example, the owner needs his friends to modify his file after he read it. So we proposed the write access permission in the PSD. For the user, the public key and file class label are all known, he can implement the algorithm to encrypt the files after he modified, and then upload them to the cloud. But whether the cloud server saves the modified file is decided by the write access control policy. On the one hand, in the complex cloud environment, if a user’s modification operations are very frequent, maybe he is very important to the user, so that the user may be stricken from outside attacks. On the other hand, in the data sharing scheme, the separate access of read and write to the file is extremely important. In PSD, not all users who have read permissions also have write permissions to the files. Whether the user has write permissions to the file is decided by the data owner. Therefore, this paper selects the improved attribute-based signature (IABS) to determine the user's write permission.

The main structure of the scheme includes five parts: an authentication center (CA), the data owner, users, mediator and cloud servers. The CA is responsible for generating master key which is sent to the owner and system parameters which are shared for all users. The mediator holds part components of the signature keys and is responsible for the validity check of attributes and users. The data owner produces the signature tree and sends it directly to the cloud server. The user encrypts the modified files and signs them using the attribute-based signature, then uploads them to the cloud server. The cloud server verifies the attribute-based signature, if the authentication is successful, the user has permission to modify files and the cloud server stores the file. Own to the limited space we will omit the specific description of the IABS scheme in PSD.

**ACCESS CONTROL SCHEME IN PUD**

Before introducing our proposed secure authentication protocol, we first make a statement for the notations used in the later, all of them are listed in Table I.

|  |  |
| --- | --- |
| Notation | Description |
| PUD | Public Domain |
| PRD | Private Domain |
| CP-ABE | Ciphertext-policy Attribute-Based Encryption |
| MA-ABE | Multi-authority Attribute-based Encryption |
| HABE | Hierarchical Attribute Encryption |
| CK | Encryption Key |
| K | Key Space |
| PK | Public Key |
| SK | Secret Key |
| KAE | Key-Aggregate Encryption |
| CA | Authorization Center |

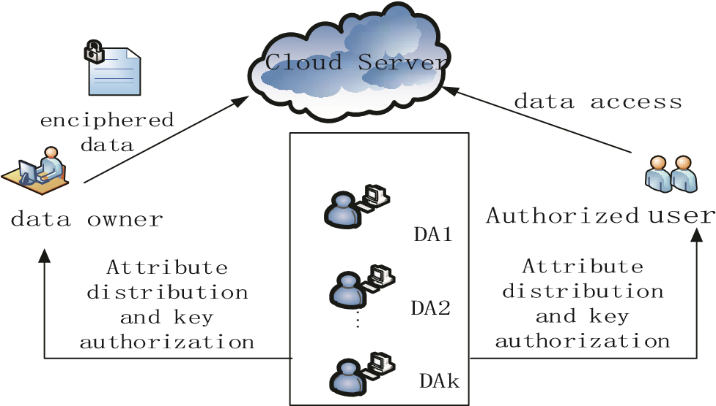
Table I

## **Scheme Design**

The PUD is characterized by a huge number of users, a lot of attributes owned by the user, complexity management, and indefinite users’ identity. In view of the above characteristics, the user can only have the read access permission. Although the attribute-based encryption scheme (CP-ABE) can achieve access control, it cannot meet the needs of complex cloud environment. In traditional CP-ABE scheme, there is only one authorized agency responsible for the management of attributes and distribution of keys. The authority may be a university registrar's office, the company's HR department or government educational organizations and so on. The data owner defines access policies and encrypts the data files in accordance with this policy. Each user is distributed a key related to his attribute. As long as the user's attributes meet the access policy he can decrypt the file. However, if there is only one authority in the system and all public and private keys are issued by the authority. Two problems will appear in the practical application:

1. In the practical cloud environment, there are a lot of authorities and each authority in their own field manages part of users’ attributes. The attributes owned by the user are issued from different authorities.
2. For example, a data owner may want to share his medical data with a user who owns the doctor attribute issued by medical institutions and the medical researcher attribute by the clinic practice management. Therefore, exploiting multi authority is more realistic in the practical scenarios.
3. If there is only one authority, all the distribution of the keys are handed over by one trusted authority. The frequent interaction between the user and trust authority will not only bring bottlenecks for the system load capacity, but also increase the potential security risks. Therefore, multi authority ABE (MA-ABE) is used in this paper.

Users in PUD do not need to interact directly with the data owner, and the attributes of the user are called role attributes. Firstly the data owner uploads the attribute-based encrypted data files to the cloud server. Then after authorized, the data owner receives the corresponding decryption key and sends a data file access request directly from the cloud server. Finally, after the cloud server returns the ciphertext, users can use their own decryption key to decrypt the ciphertext. The framework of this area is shown in Fig.3.



**Figure. Access control framework of PUD**

## **Access Control Process**

Based on the above analysis, we use a hierarchical attribute encryption scheme (HABE) to implement access control in

PUD.

1. Files creation: The creating of files is completed by the data owner. In general, in order to protect the privacy of the data file, the data owner firstly encrypts data file, and then stores it in the cloud.
2. To reduce the ciphertext size and complexity, the data owner combines the symmetric encryption scheme with public key encryption scheme, namely that each file is firstly encrypted with symmetric encryption key called CK, then CK is encrypted with the HABE program. Before the data file uploaded to the cloud, the process of creating a data file is as follows:
3. Select a unique *ID* for the data file.
4. Choose a random symmetric encryption key*CK*m*R K* . *K* means key space, and encrypt the data file with *CK*.
5. Define access tree T, use the algorithm *H A B E Encrypt*. *P K e* , *C K T*, to encrypt *CK* and return the *CT*.
6. The data owner computes the *CT* by hash operations and signs *h*(*CT*) to get the signature *SG* , on the one hand to ensure the integrity of the data, on the other hand to facilitate the cloud and user to authenticate the identity of the data owner.
7. Data access: If the user wants to access a data file, he should get the file from the cloud server and decrypt the encrypted data file, which corresponds to the decryption process. There are two stages: firstly use the algorithm *HABE*  *Encrypt PK e* ,*CK T*, to decrypt the symmetric encryption key *CK*, then use the key *CK* to decrypt the data file.
8. Files deletion: If the data owner wants to delete a file, he can send the file *ID* and his signature *SG* to the cloud server, then the cloud servers delete the files after verifying the signature of the data owner.
9. Attribute revocation: The authority assigns attributes to each user and attaches the set of attributes with an expiration time *T* . The attributes of access control tree contain a time attribute *T*c, if *T T*! c and the attributes match, then this file can be access to. So the data owner can restrict users’ access permissions by changing the time attributes.
10. Users’ attributes Revocation: The *DA* calculates the minimum set of attributes *A*min that allows users’ access revocation, and *Anew* *A A*min , making *T A* min returns null. Set a new expiration time to each attribute set, generate new private key components and return it to the client.

**SYSTEM SIMULATION AND PERFORMANCE**

**ANALYSIS**

## **Security Analysis**

In PSD, the user can only decrypt the files corresponding to the received aggregate keys and do not have access to other files, so that the data owner controls the users’ access permissions. When the data file is modified, although CA is trusted, also the system parameters and revocation instructions are generated by the CA. The signature policy is formulated by the data owner and sent directly to the cloud server. The CA does not know the signature policy. Assuming that CA cannot give itself authorization, as long as the attributes of CA cannot meet the access policy, it is not valid to modify the fileIn the process of the users’ signature, the signature key is only related to the users’ attributes, so the user's identity is safe. On the whole, the IABS scheme can protect users’ identity privacy.

In PUD, this paper employs the HABE scheme for the large number of users with uncertain identity in this region. For the trusted CA, it can only issue the private key and the corresponding attribute structure to the authority in the first level not to the users, so that the CA does not directly control the user's private key, thus reducing the trust in CA. In addition, the user's private keys are managed by multiple authorized agencies, which can avoid users’ privacy leakage.

## **Simulation Analysis**

In our KAE scheme in the PSD, the system parameters are generated by the trusted authority, which is not within our consideration. Furthermore, the˄ ˅*e g g*ˆ 1, *n* can be calculated in the system setup. In addition, the aggregate key only needs one pairing operation, and to calculate a pairing operation is very fast, the specific comparison can be seen in Fig.4.

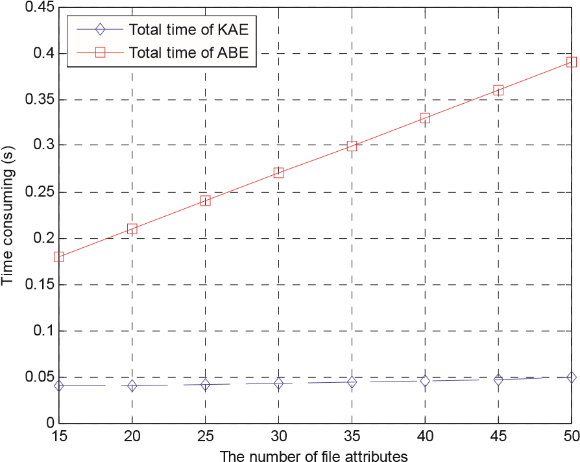
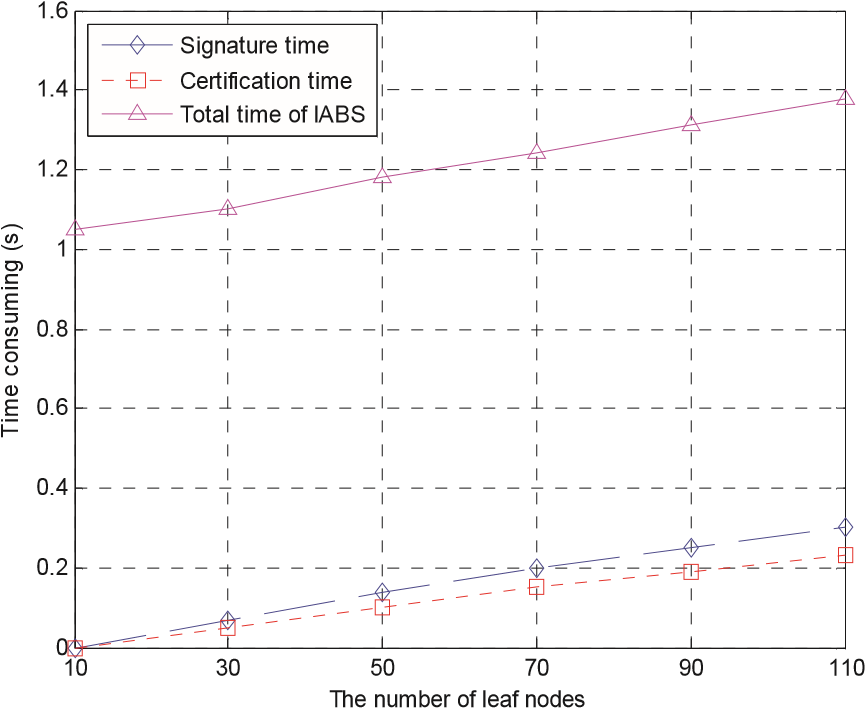


Figure 4. Total time of KAE and ABE

In Fig.4, the attribute-based encryption algorithm of the MAH-ABE scheme spent much more time than the KAE

algorithm used in our scheme. If the attribute revocation occurs, the ABE algorithm will be more time-consuming. More importantly, the growth rate of time spent with the number of file attributes is much higher than KAE algorithm. The simulation results show the high efficiency of our scheme.

In Fig.5, the user only needs a very short time to sign the modified files. While, the authentication time only makes up a small part, so the process of signature and authentication consume a very small time. Therefore, from the client's perspective, the program is efficient.



**Figure. The signature and authentication time of IABS**

**CONCLUSION**

We propose access control system (PS-ACS), which is privilege separation based on privacy protection. Through the analysis of cloud environment and the characteristics of the user, we divide the users into personal domain (PSD) and public domain (PUD) logically. In the PSD, the KAE algorithm is applied to implement users read access permissions and greatly improved efficiency. The IABS scheme is employed to achieve the write permissions and the separation of read and write permissions to protect the privacy of the user's identity. In the PUD, we use the HABE scheme to avoid the issues of single point of failure and to achieve data sharing. Furthermore, the paper analyzes the scheme from security and efficiency, and the simulation results are given. By comparing with the MAH-ABE scheme, the proposed scheme shows the feasibility and superiority to protect the privacy of data in cloud-based services.

**REFERENCES**

1. S. Yu, C. Wang, K. Ren, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” Proc. IEEE INFOCOM, pp. 1-9, 2010.
2. J. Bethencourt, A. Sahai, B. Waters, “Ciphertext-policy attribute-based encryption,” Proc. Security and Privacy, pp. 321-334, 2007.
3. J. Hur, D.K. Noh, “Attribute-based access control with efficient revocation in data outsourcing systems,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 7 pp. 1214-1221, 2011.
4. A. Lewko, B. Waters, “Decentralizing attribute-Based encryption,” Proc. Advances in Cryptology-EUROCRYPT, pp. 568-588, 2011.
5. M. Li, S. Yu, Y. Zheng, “Scalable and secure sharing of personal health records in cloud computing using attribute-Based Encryption,” IEEE Transactions on Parallel and Distributed System, vol. 24, no. 1, pp. 131143, 2013.
6. C.K. Chu, S.S.M. Chow, W.G. Tzeng, “Key-aggregate cryptosystem for scalable data sharing in cloud storage,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, pp.468-477, 2014.
7. J. Li, K. Kim, “Hidden attribute-based signatures without anonymity revocation,” Information Sciences, vol. 180, no. 9, pp. 1681-1689, 2010.
8. H.K. Maji, M. Prabhakaran, M. Rosulek, “Attribute-Based Signatures,” Proc. Topics in Cryptology - CT-RSA, pp. 376-392, 2011.
9. S. Kumar, S. Agrawal, S. Balaraman, “Attribute based signatures for bounded multi-level threshold circuits,” Proc. Public Key Infrastructures, Services and Applications, pp. 141-154, 2011.

**CODE**

