**Vietnam National University Ho Chi Minh City**

**University of Information Technology**

Instructor: Ph.D Nguyễn Ngọc Tự

Students:

Lê Quốc Khánh – 21520978  
Hoàng Trí Tường – 21521654

Trần Trọng Hiếu – 21520859

**ENCRYPTION, ACCESS CONTROL, QUERY IN CLOUD-NATIVE DBMS**

University of Information Technology, 1/2024

**Table of Contents**

[I. Applications Scenario 3](#_Toc155549366)

[1. Introduction 3](#_Toc155549367)

[2. Gaps in Security 3](#_Toc155549368)

[3. Motivations 4](#_Toc155549369)

[4. Desired Functional and Security Features 5](#_Toc155549370)

[5. Related Stakeholders 5](#_Toc155549371)

[II. Used Algorithms Overview 5](#_Toc155549372)

[1. Curved 5](#_Toc155549373)

[2. AES256 7](#_Toc155549374)

[3. Ciphertext-Policy ABE 7](#_Toc155549375)

[III. Solutions 7](#_Toc155549376)

[1. Solution Architecture 7](#_Toc155549377)

[2. Functional Features 8](#_Toc155549378)

[3. Security Features 9](#_Toc155549379)

[IV. Implementation & Testing 9](#_Toc155549380)

[V. Demonstration 10](#_Toc155549381)

# Applications Scenario

## Introduction

The integration of cloud-native database management systems (DBMS) has revolutionized data storage and management. However, this adoption brings forth significant concerns regarding data security, specifically in areas of data encryption, access control, and query security. This report outlines identified gaps, motivations for securing cloud-native DBMS, proposed functional and security features, stakeholders involved, specific algorithms, solutions, implementation details, and testing strategies.

## Gaps in Security

The vulnerabilities in cloud-native DBMS include shortcomings in data encryption methods, inadequacies in access control mechanisms, and weaknesses in query security. These vulnerabilities have been substantiated by real-world breaches and data exposures, showcasing the critical need for stronger security measures.

* Data Encryption:
  + **Weak Encryption Protocols:** Usage of outdated or weak encryption algorithms can expose data to potential breaches. It's crucial to use strong encryption standards like AES (Advanced Encryption Standard) for data at rest and in transit.
  + **Misconfigured Encryption:** Improperly configured encryption settings or keys can lead to vulnerabilities. Secure key management and regular rotation of encryption keys are essential.
* Access Control:
  + **Insufficient Authentication Measures:** Weak authentication methods or compromised credentials can lead to unauthorized access.
  + **Dependency on Third-Party Services:** Cloud-native systems often rely on various third-party services or APIs. If any of these services are compromised, it could impact the overall security of the implementation.
* Query:
  + **Injection Attacks:** SQL injection vulnerabilities can allow attackers to manipulate queries, leading to data breaches or corruption.
  + **Excessive Privileges:** Queries might unintentionally have excessive privileges, allowing unauthorized access or unintended data exposure. Implementing the principle of least privilege can mitigate this risk.

Here is a real-life example of a cloud-native data breach, how it evolved and how it possibly could have been avoided.[[1]](#footnote-1)

Target Profile: The company is a photo-sharing social media application, with over 20 million users. It stores over 1PB of user data within Amazon Web Services (AWS), and in 2018, it was the victim of a massive data breach that exposed nearly 20 million user records. This is how it happened.

**Compromising a legitimate user:** The first step in a data breach is that an attacker compromises the credentials of a legitimate user. In this incident, an attacker used a spear-phishing attack to obtain an administrative user’s credentials to the company’s environment.

**Fortifying access:** After compromising a legitimate user, a hacker frequently takes steps to fortify access to the environment, independent of the compromised user. In this case, the attacker connected to the company’s cloud environment through an IP address registered in a foreign country and created API access keys with full administrative access.

**Reconnaissance:** Once inside, an attacker then needs to map out what permissions are granted and what actions this role allows.

**Exploitation:** Once the available permissions in the account have been determined, the attacker can proceed to exploit them. Among other activities, the attacker duplicated the master user database and exposed it to the outside world with public permissions.

**Exfiltration:** Finally, with customer information at hand, the attacker copied the data outside of the network, gaining access to over 20 million user records that contained personal user information.

## Motivations

Cloud-Native database management systems (DBMS) plays a crucial role in today data storage and processing in distributed environment. As several organizations increasingly migrate their data to the cloud. The sensitivity of data stored, the shared environment of the cloud, and regulatory compliance requirements, securing cloud-native DBMS becomes imperative. The potential risks posed by unauthorized access, data breaches, and regulatory non-compliance underscore the necessity of robust security measures.

**Confidentiality**: Encryption ensures that data remains confidential, even if unauthorized parties gain access to the database. Encrypting data at rest and in transit prevents sensitive information from being compromised

**Access Control**: Implementing robust access controls ensures that only authorized users have the necessary permissions to view, modify, or delete specific data. Fine-grained access controls based on user attributed enhance security by limiting access to sensitive information.  
**ABE-based Query Processing**: Implementing ABE-based query processing allows for secure and efficient querying of encrypted data. This methodology enables users to run queries on encrypted data without requiring full decryption, maintaining data security while allowing for efficient data retrieval and analysis.

## Desired Functional and Security Features

* Data Encryption: Strong data encryption to protect data at rest and in transit.
* Access Control: Fine-grained access control to restrict data access based on user roles
* Secure Query Processing: Secure and efficient querying of encrypted data using ABE.

## Related Stakeholders

Key stakeholders involved in securing cloud-native DBMS include database administrators (owner), cloud service providers, and end-users(member). Each stakeholder plays a critical role in ensuring the security and integrity of the system.

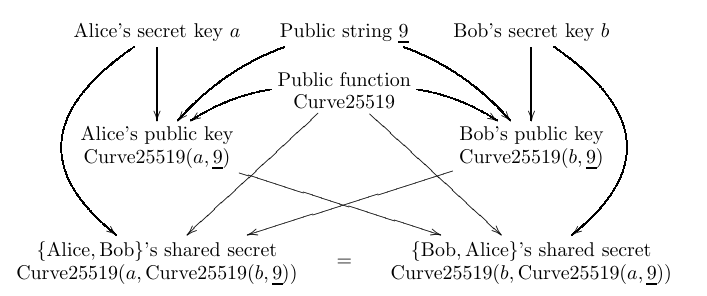
# Used Algorithms Overview

Here is an overview of out used algorithms: Curved for key agreement, AES256 for encryption and ABE for fine-grained access control).

## Curved [[2]](#footnote-2)

Curve is a state of the art elliptic-curve Diffie-Hellman function suitable for a wide variety of cryptographic applications and it is one of the fastest curve in ECC

Each Curve25519 user has a 32-byte secret key and a 32-byte public key. Each set of two Curve25519 users has a 32-byte shared secret used to authenticate and encrypt messages between the two users.



*Picture 1: Public Key Exchange with Curve25519*

The following picture shows the data flow from secret keys through public keys to a shared secret

* Alice and Bob generate their own secret keys(a and b). These keys are never revealed to anyone else.
* Alice and Bob also generate public keys (Curve25519(a, 9) and Curve25519(b, 9)). These keys are derived from their secret keys and a public function Curve25519.
* Alice sends her public key to Bob, and Bob sends his public key to Alice.
* Alice calculates a shared secret key by applying the Curve25519 function to her secret key (a) and Bob’s public key (Curve25519(b, 9)).
* Bob calculates the same shared secret key by applying the Curve25519 function to his secret key (b) and Alice's public key(Curve25519(a, 9)).

A hash of the shared secret Curve25519(a, Curve25519(b, 9)) is used as the key for a secret-key authentication system (to authenticate messages), or as the key for a secret-key authenticated-encryption system (to simultaneously encrypt and authenticate messages).

## AES – 256[[3]](#footnote-3)

The **Advanced Encryption Standard** (**AES**), also known by its original name **Rijndael,** is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST)

AES Basic Structure:

* Block cipher
* Block size 16-bytes
* Three key length: 128, 192, 256 bits
* AES256 has 14 rounds

AES-256 is considered to be quantum resistant, as it has similar quantum resistance to AES-128's resistance against traditional, non-quantum, attacks. AES-192 and AES-128 are not considered quantum resistant due to their smaller key sizes. AES-192 has a strength of 96-bits against quantum attacks and AES-128 has 64-bits of strength against quantum attacks, making them both insecure.

## Ciphertext-Policy ABE

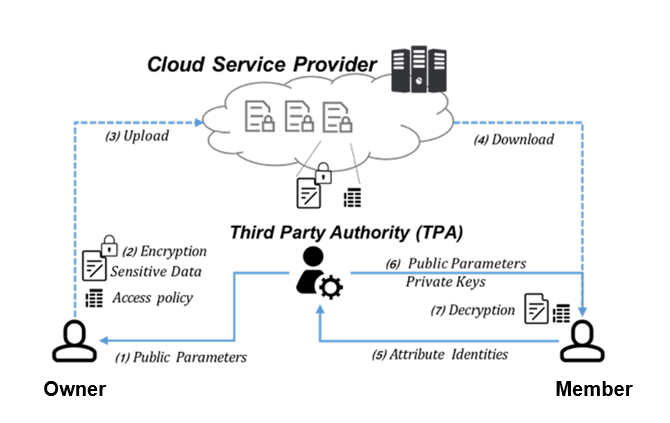
**Attribute-based encryption** is a generalization of public-key encryption which enables fine grained access control of encrypted data using authorization policies. The secret key of a user and the ciphertext are dependent upon attributes. In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext.

Instead of secret keys for policies and ciphertexts encrypted with sets of attributes in Key-Policy ABE. Ciphertext-Policy ABE uses access trees to encrypt data and users' secret keys are generated over a set of attributes.

# Solutions

## Solution Architecture

An owner of a doctor group encrypts his data with a specified access policy and then sends to the CSP. With a provided link, anyone in the hospital can download the outsourced encrypted data. However, only a member of a doctor groups who has attributes that satisfy the access policy can access(decrypt) the encrypted data



*Picture 2: Solution Architecture*

## Functional Features

* **Data Encryption:**
  + **AES – 256 Encryption:** Utilize the AES256 encryption algorithm for securing data at rest and in transit. This robust encryption standard ensures high-level security against unauthorized access.
  + **Secure Encryption/Decryption Processes**: Implement efficient and secure processes for encrypting and decrypting data. Ensure these processes maintain integrity and confidentiality during data transmission and storage.
* **Access Control:**
  + **Attribute-Based Encryption (ABE):** Implement ABE for fine-grained access control.
  + **Fine-grained Access Control based on User Attributes:** Leverage Curved for key agreement and establish fine-grained access control mechanisms based on user attributes. Users' specific attributes define their access privileges, enabling precise control over data accessibility.
  + **Enforcement of Data Access Policies:** Enforce access policies based on user attributes. These policies dictate who can access what data, ensuring adherence to security protocols and regulatory requirements.
* **Secure Query Processing:**
  + **ABE-based Query Processing:** Implement ABE-based query processing to enable secure and efficient querying of encrypted data. This feature allows authorized users to perform queries on encrypted data without compromising security.

## Security Features

* **Authentication and Key Agreement:**
  + **Curved Key Agreement:** Employ Curved for secure key agreement between users and the system. This ensures strong, authenticated communication channels and prevents unauthorized access.
  + **Strong User Authentication:** Implement robust user authentication mechanisms, utilizing Curved , to verify user identities and prevent unauthorized access attempts.
* **Access Control:**
  + **Fine-grained Access Control using ABE:** Leverage ABE along with Curved for key agreement to establish granular access controls based on user attributes. This approach ensures that access to sensitive data is strictly controlled according to predefined attribute-based policies.
  + **Enforcement of Access Policies:** Enforce access policies consistently throughout the system, ensuring that only authorized users with requisite attributes can access specific data.

# Implementation & Testing

Using Python language version 3.7.x for the demonstration. There are many cryptographic libraries in Python, we are using tabulate and pycryptodome which are the latest versions. As we mentioned before, CPABE is used and Charm-crypto library is highly recommended for the project. Charm-crypto cannot be installed by using pip command like other libraries. Before installing this library, make sure the user’s computer has dependencies such as GMP, PBC and OpenSSL library. After that, using “git clone” command and compile Charm-crypto library.

We use SQLite to store database of the hospital. There are some advantages to show SQLite is suitable for this project. Firstly, it is lightweight, so it does not require much time or effort to setup or maintain. Secondly, it is open source and free to use. Lastly, reading and writing are very fast for SQLite database. It is almost 35% faster than File system.

# Demonstration

When a user starts a connection to Server, there is a key agreement between Client and Server using ECDH method and Curve25519 algorithm. As a result, Client and Server both have the same shared – secret key.

We design an encrypted file to upload to Storage based on CP – ABE following this structure:

|  |
| --- |
| Session key size (8 bytes) |
| IV (16 bytes) |
| Session key (in bytes) |
| Encrypted file’s content (in bytes) |

At first, we randomly generate IV and session key (from GT). Then, we encrypt this session key using scheme named ***ac17*** of CP – ABE. Next, we do some tricks to convert it into bytes, then hash it using ***SHA – 256*** to get a hashed value. This value seems to be a key and it is used to encrypt file content with ***AES – 256 mode CFB***.

If a user wants to download an encrypted file from Storage, firstly, Server uses the shared – secret key before to encrypt the CP – ABE secret key under user’s attributes using ***AES – 256 mode GCM*** then returns to user with CP – ABE public key and encrypted file from Storage. When received, user uses the shared – secret key to decrypt the CP – ABE secret key. This key is used to decrypt the session key attached in the encrypted file. The session key is subsequently hashed using ***SHA–256*** algorithm to get a hashed value, and this value is used as a key to decrypt the encrypted file’s content using ***AES – 256 mode CFB***. Finally, we get a plain file.

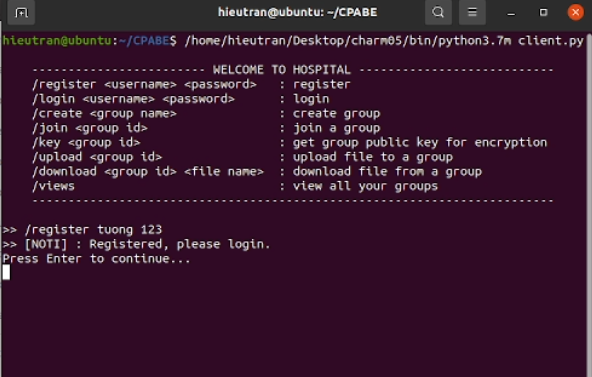
* **Functions**

## Registration

Users must follow this command:

/register <username> <password>

After that, Server notices that registration be successful and users must login to start work session.



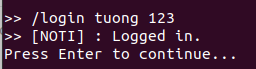
*Picture 3: Users register successfully*

## Login

Users must follow this command:

/login <username> <password>

When users press Enter, Server checks user’s information stored in table DOCTORS in database. If there is a match, users’ login is successful; otherwise, users receive a notification “This account does not exist” from Server.



*Picture 4: Users sign in successfully*



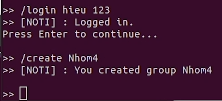
*Picture 5: User’s login failed*

## Creating group

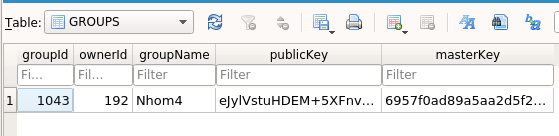
Users must follow this command:

/create <group name>

After that, group’s information (including group ID, owner ID, group name, public key and master key) is stored in table GROUPS in database. Moreover, the user who creates group is the owner of that group.



*Picture 6: User created group successfully*



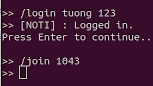
*Picture 7: Group’s information in table GROUPS*

## Joining group

Users must follow this command:

/join <group ID>

Users have known group’s ID before and request to join this group.



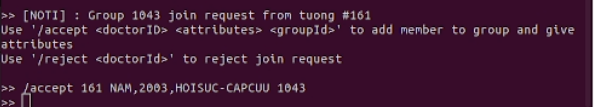
*Picture 8: Users join a group through its ID*

## Accepting/Rejecting request

Users must follow this command to accept request:

/accept <doctor ID> <attributes> <group ID>

When a user requests to join one group, this request is forwarded to the group’s owner. The owner can accept this request then assign attributes to this member following format <Gender>,<Year of Birth>,<Department> or reject using command /reject <doctor ID>.



*Picture 9: Group’s owner accepts member’s request*



*Picture 10: Member receives reply from group’s owner*

## Getting group’s public key

Users must follow this command:

/key <group ID>

This step is important because users must obtain group’s public key to encrypt files before uploading them to their group.



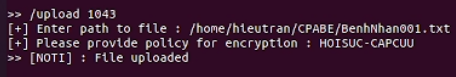
*Picture 11: Users receive their group’s public key successfully*

## Uploading files

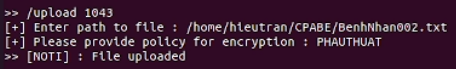
Users must follow this command:

/upload <group ID>

Users choose which group that their encrypted files with obtained public key belong to. They have to type in file path/file name and access policy string for each file. Subsequently, these files are stored in Storage.



*Picture 12: In this case, users must be in Emergency Resuscitation Department to access this file*



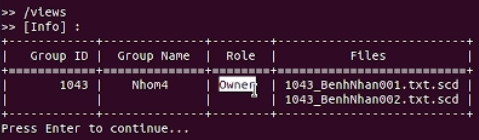
*Picture 13: In this case, users must be in Surgery Department to access this file*

## Viewing group

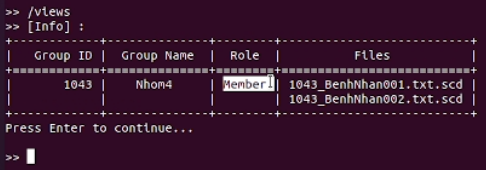
Users must follow this command:

/views

Users can view their groups’ information including Group ID, Group Name, Role and Files.



*Picture 14: Role is Owner*



*Picture 15: Role is Member*

## Downloading files

Users must follow this command:

/download <group ID> <file name>

Users can only download files under their group’s management. Furthermore, users’ attributes must satisfy the policy of the file they want to download.

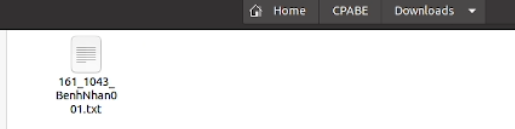
In this case, the account “tuong” has an attribute “HOISUC-CAPCUU”, so this user can only download the file named 1043\_BenhNhan001.txt.scd except 1043\_BenhNhan002.txt.scd because the file 1043\_BenhNhan002.txt.scd can only be accessed by whom in Surgery Department.



*Picture 16: Failed to download 1043\_BenhNhan002.txt.scd*



*Picture 17: But it is fine with 1043\_BenhNhan001.txt.scd*



*Picture 18: The decrypted file is stored in Downloads folder*

1. [Radware-Anatomy-Cloud-Data-Breach-Guide-Final-2020.pdf.aspx](https://www.radware.com/getattachment/d9fa9662-d9dc-45eb-818b-4aed779bd458/Radware-Anatomy-Cloud-Data-Breach-Guide-Final-2020.pdf.aspx?fbclid=IwAR2G0x5rshkzCLxcrFCD7iAHRObYazFz69q5S3bpEs4IBvO1bbmfbwnIrpk) [↑](#footnote-ref-1)
2. [curve25519-20060209.pdf (yp.to)](https://cr.yp.to/ecdh/curve25519-20060209.pdf) [↑](#footnote-ref-2)
3. [Advanced Encryption Standard - Wikipedia](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) [↑](#footnote-ref-3)