Sainteknol Vol 23, No 1 (2025): June 2025

https://journal.unnes.ac.id/journals/sainteknol

Implementation of Hierarchy Attribute-Based Encryption (HABE) to Secure Information Stored in Cloud Storage

Arief Arfriandi[1*], Rahmat Gernowo[2], R. Rizal Isnanto[3]

[1,2,3]Doctoral Program of Information System, School of Postgraduate Studies, Universitas Diponegoro, Central Java, Indonesia

[*]Corresponding Author: ariefarfriandi@students.undip.ac.id

Abstract

The usage of cloud storage is currently growing, particularly for media used for information storage. Information saved in cloud storage makes it possible for the owner of the cloud storage to access the Information without the owner's permission. The privacy of access and management of access to the information kept in the cloud storage becomes a problem and security risk when using that cloud storage because the owner cannot ensure that the stored information is only accessed by authorized parties. Access control management is necessary for information owners to guarantee the safety of the information they hold. To regulate access to information kept in cloud storage, this study intends to apply the Hierarchy Attribute-Based Encryption (HABE) technique. A hierarchical attribute base is used to encrypt and decode information saved in cloud storage through the application of HABE, making hierarchical attributes the essential element in controlling access to the information. The hierarchical structure utilized to manage access control for information security will also be detailed in this research. This implementation procedure is completed before the information is stored in cloud storage. The mathematical computations used to secure text information during the encryption and decryption operations are described in the HABE implementation method and debate. By using Hierarchy Attribute-Based Encryption, security concerns with cloud storage can be resolved and flexible access control management in compliance with corporate standards is made possible.

Keywords: Information Security, Cloud Storage, HABE, ABE, Cryptography.

INTRODUCTION

How information is managed and stored has changed significantly as a result of the development of information technology. Users can store information online and access it from any location at any time by using cloud storage. Because of its great scalability, efficiency, and flexibility, cloud storage is a well-liked option for both consumers and businesses [1]. Behind these advantages, nevertheless, there are serious information security issues, namely the dangers of information theft, illegal access, and privacy violations [2] [3]. We use resources in a cloud environment according to their consumption. The primary services offered by this cloud environment are Platform As A Service (PaaS), which offers the compiled output from programming languages, Infrastructure as a Service (IaaS), which provides processing and storage resources, and Software as a Service (SaaS), which gives cloud users an interface. Private cloud, public cloud, community cloud, and hybrid cloud are those that are used in computing. A few of these cloud models are differentiated according to the cloud model's owner and consumers. An enterprise owns a private cloud, the general public or a large number of consumers utilize a public cloud, and a hybrid cloud combines a private and public cloud. Large amounts of data or information can now be saved in the cloud thanks to its existence, but cloud users are still hesitant to do so because of security and privacy concerns [2]. Data or information that may be viewed or handled remotely is stored in cloud storage, which is stored in a cloud environment [1]. Sensitive information stored in cloud storage can be accessed and shared with unauthorized persons by the cloud storage provider. Before being kept in cloud storage, data or information must be encrypted, and user rights and access must be limited to ensure security. Therefore, privacy and user access control are two factors that need to be taken into account when keeping data or information [4]. Asymmetric key encryption techniques are one way that cryptography can be used to secure data or information. Although asymmetric keys preserve privacy by using distinct keys for encryption and decryption, they have not yet solved access control in cloud storage. Hierarchical Attribute-Based Encryption (HABE), an attribute-based cryptographic technique, has been created to overcome this difficulty. HABE differs significantly from previous attribute-free cryptography techniques. The Attribute-Based Encryption (ABE) technique is extended by HABE, which uses a hierarchy of user attributes to offer more secure and adaptable access control. Users are assigned encryption keys according to a set of attributes when HABE is implemented, and access policies that refer to those qualities encrypt the information. Only users who fulfill particular attribute policies can access the data saved in cloud storage thanks to this method's more precise and extensive information access controls. There are several reasons why HABE is so important when discussing cloud storage. First, multi-level data management, which is frequently present in large businesses, is supported by this paradigm. How information is managed and stored has changed significantly as a result of the development of information technology. Users can store information online and access it from any location at any time by using cloud storage. Because of its great scalability, efficiency, and flexibility, cloud storage is a well-liked option for both consumers and businesses [1]. Behind these advantages, nevertheless, there are serious information security issues, namely the dangers of information theft, illegal access, and privacy violations [2] [3]. We use resources in a cloud environment according to their consumption. The primary services offered by this cloud environment are Platform As A Service (PaaS), which offers the compiled output from programming languages, Infrastructure as a Service (IaaS), which provides processing and storage resources, and Software as a Service (SaaS), which gives cloud users an interface. Private cloud, public cloud, community cloud, and hybrid cloud are those that are used in computing. A few of these cloud models are differentiated according to the cloud model's owner and consumers. An enterprise owns a private cloud, the general public or a large number of consumers utilize a public cloud, and a hybrid cloud combines a private and public cloud. Large amounts of data or information can now be saved in the cloud thanks to its existence, but cloud users are still hesitant to do so because of security and privacy concerns [2]. Data or information that may be viewed or handled remotely is stored in cloud storage, which is stored in a cloud environment [1]. Sensitive information stored in cloud storage can be accessed and shared with unauthorized persons by the cloud storage provider. Before being kept in cloud storage, data or information must be encrypted, and user rights and access must be limited to ensure security. Therefore, privacy and user access control are two factors that need to be taken into account when keeping data or information [4]. Asymmetric key encryption techniques are one way that cryptography can be used to secure data or information. Although asymmetric keys preserve privacy by using distinct keys for encryption and decryption, they have not yet solved access control in cloud storage. Hierarchical Attribute-Based Encryption (HABE), an attribute-based cryptographic technique, has been created to overcome this difficulty. HABE differs significantly from previous attribute-free cryptography techniques. The Attribute-Based Encryption (ABE) technique is extended by HABE, which uses a hierarchy of user attributes to offer more secure and adaptable access control. Users are assigned encryption keys according to a set of attributes when HABE is implemented, and access policies that refer to those qualities encrypt the information. Only users who fulfill particular attribute policies can access the information saved in cloud storage thanks to this method's more precise and extensive information access controls. There are several reasons why HABE is so important when discussing cloud storage. First, multi-level data management, which is frequently present in large businesses, is supported by this paradigm. HABE can lower the computational cost that typically arises in regular ABE by utilizing attribute hierarchy. Third, by reducing the possibility of information leaking to unauthorized parties, this strategy improves information security. This study examines the use of HABE to improve the security of information kept in cloud storage, emphasizing the technology's technical benefits, implementation difficulties, and pertinent case studies. We anticipate that HABE will facilitate the development of a more reliable and secure cloud storage information storage system. Numerous investigations have been carried out about the application of HABE. First proposed Hierarchical Attribute-Based Encryption (HABE) as an extension of ABE that enables hierarchical attribute management [5]. The study demonstrates that HABE works well for applications that need hierarchical data management, including multi-level permission organizations. To improve efficiency and flexibility in large systems, HABE divides critical authority into multiple domains. Only users with the appropriate attributes can access the information thanks to HABE's attribute-based granular access control feature. This paper by G. Wang and Wu (2010) combines the Hierarchical Identity-Based Encryption (HIBE) system with Ciphertext-Policy Attribute-Based Encryption (CP-ABE) to improve cloud storage services' access control and full delegation [6]. The research by Wang et al. (2016) suggests a good way to encrypt files in the cloud that uses hierarchical file attributes. To do this, layered access structures are combined into a single access structure that is then used to encrypt files that are organized in a hierarchy [7]. The security of the CP-HABE scheme was examined in the paper by Ali, and it was discovered that the system includes flaws in the key delegation mechanism that lets anyone with a single attribute access the encrypted information. One of the expanded mechanisms of Attribute-Based Encryption (ABE) [7] that integrates the idea of hierarchy in key management systems with attribute-based encryption is Hierarchical Attribute-Based Encryption (HABE), as depicted in Figure 1 of the ABE taxonomy. In contrast to conventional encryption techniques, ABE links encryption and decryption keys to certain characteristics, enabling more adaptable access control. In Figure 1, the ABE taxonomy is displayed.

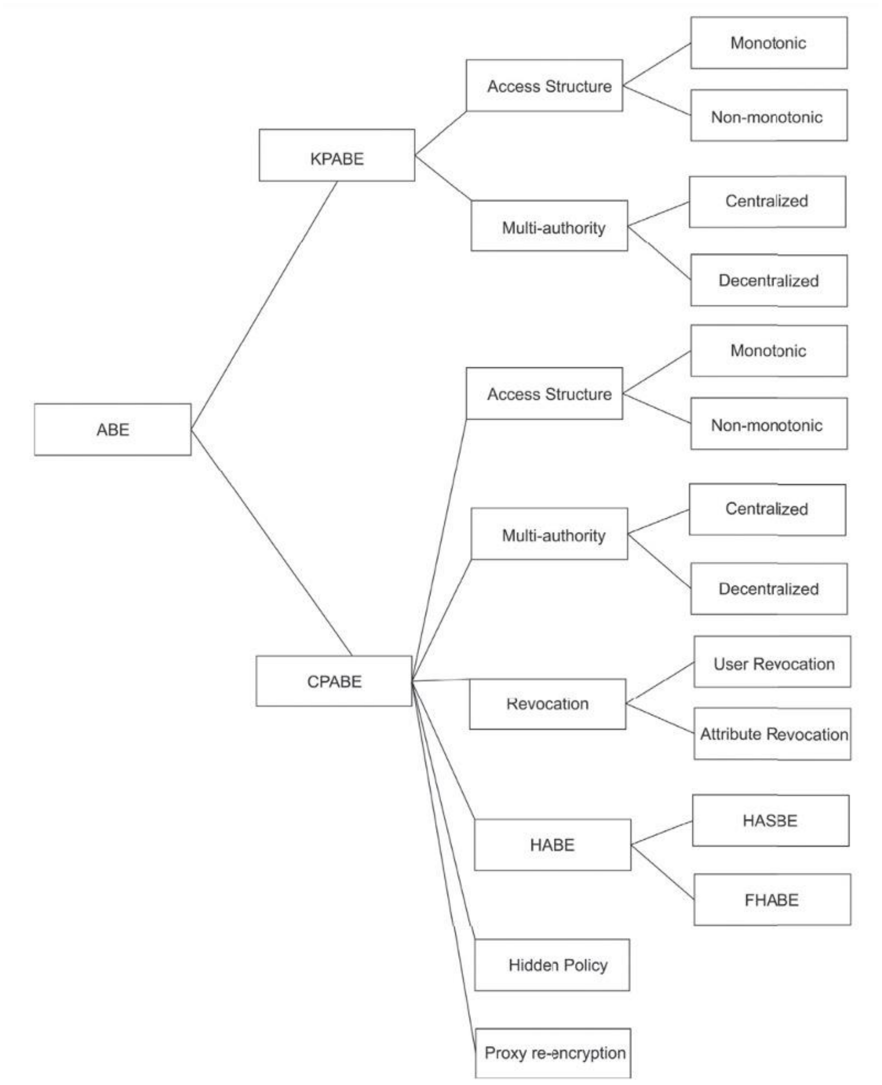


Figure 1. ABE Taxonomy [8].

The ideas of ABE and HIBE are combined in HABE [9]. The following are HABE's primary attributes:

a. A hierarchical structure is used for key authorities, allowing higher-level authorities to assign lower-level authorities the authority to generate keys.
b. Using attributes to ascertain a user's access privileges to encrypted information is known as attribute-based access control.

The following is the HABE work procedure [10].

a. Configuration: System settings and master keys are generated by the root authority. Lower-level authority are given these parameters.
b. Key Generation: Using their own characteristics and access regulations, lower-level authorities create keys. This procedure may be carried out all the way to the end-user level.
c. Encryption: Encryption-defined properties and access controls are used to protect information.
d. Decryption: The information can be decrypted by users who possess keys with the right characteristics.

The following are HABE's benefits [11].

a. Scalability: More effective and scalable key management is made possible by a hierarchical structure.
b. Fine-grained Access Control: More accurate and adaptable access control is made possible by attribute-based access controls.
c. Key Delegation: To increase efficiency and security, higher authorities might grant lower authorities the authority to generate keys.

Applications for HABE implementation and use are common in systems that need intricate and organized access control [12], like:

a. Distributed information Management System: For instance, cloud storage where multiple parties with varying access rights must be able to access information.
b. Network Security System: To protect information and communication in a network with several tiers of users and authorities.

METHODS

In this study, we use an experimental approach to implement HABE for securing information in cloud storage. In the implementation of HABE, an encryption process will be carried out on text information that will be stored in cloud storage. The testing parameter used is the time efficiency required during the encryption process.

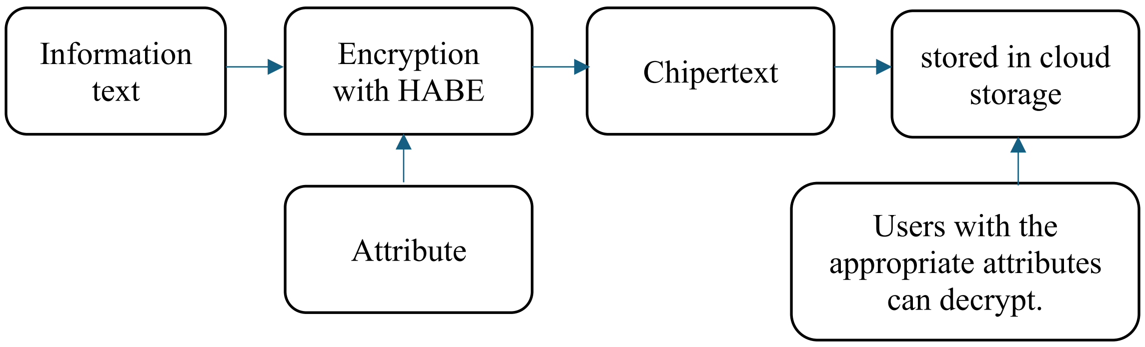


Figure 2. HABE Implementation Method for Information Security

Information and Attribute Used

The information used is plaintext. "Informasi Rahasia dari HR Manager".
Users consist of,
level01 = "`HR_Manager`"
level02 = "`IT Developer`"

the role or access policy used when encrypting information is,

`["role:manager", "department:HR"]`: This policy states that only users who have the attribute `role:manager` and `department:HR` who can access the encrypted information.

Thus, only the `HR_Manager` can decrypt the information because they meet both attributes required in the access policy. On the other hand, the `IT_Developer` cannot decrypt the information because they do not have the attributes that comply with the policy.

Architecture Hierarchy Attribute-Based Encryption (HABE)

Komponen arsitektur HABE

1. In addition to setting settings and overseeing the public system and master secret key, the Root Authority (RA) also grants and assigns authority to sub-authorities in the hierarchy.
2. Hierarchical Authorities (HA) function to generate attribute keys for users.
3. The owner of the data or information, known as the Data Owner (DO), encrypts the information in accordance with the access policy in place.
4. The user is given a key that matches their attributes, allowing them to decrypt information based on those attributes.

Workflow of Hierarchy Attribute-Based Encryption (HABE)

In HABE, the workflow is shown in Figure 3.

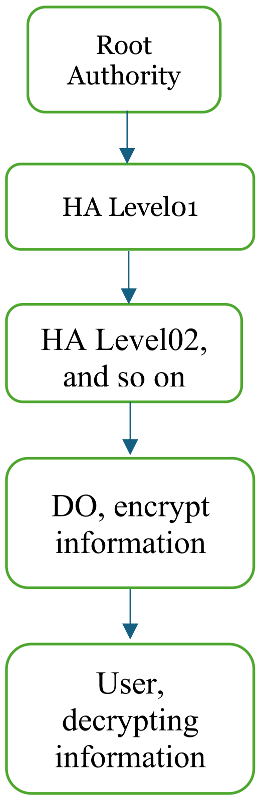


Figure 3. Workflow HABE

The following is the HABE work procedure [10].

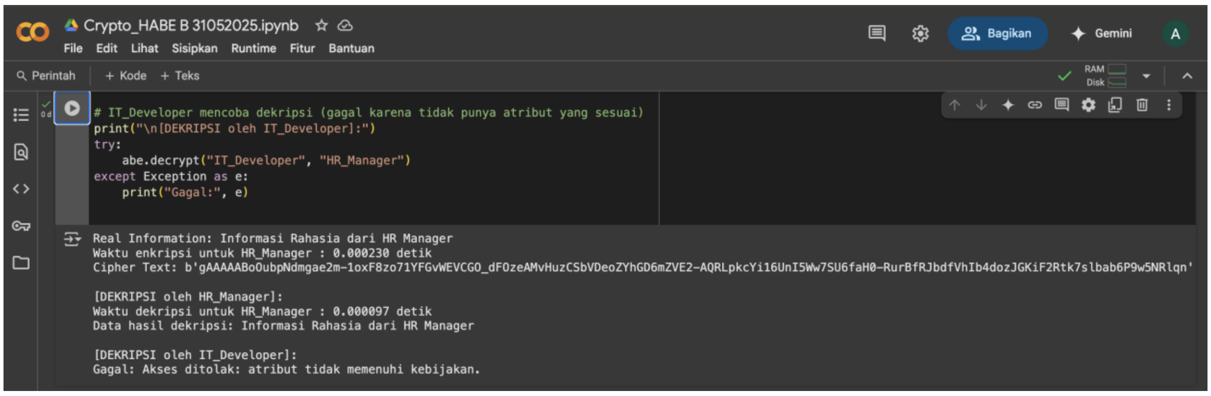
a. The setup process is performed by the RA, which creates the system parameters and the master secret key (MSK), and distributes the keys to the Level 1 HAs.
b. Key Generation: HA using their own characteristics and access regulations, lower-level authorities create keys. This procedure may be carried out all the way to the end-user level.
c. Distribution of attribute keys to users. Users get the attribute keys and associated with their attributes from HA.
d. DO encrypt information in accordance with the attribute access policy
e. The user decrypts the information with his attribute key, which is obtained from HA.

To determine the performance of HABE, the implementation results will be compared between the HABE and ABE algorithms using the same role. The results of the HABE encryption will be stored in cloud storage, making the stored information more secure.

Implementation And Discussion

The implementation of HABE is demonstrated using the Python programming language.

```
import time
from cryptography.fernet import Fernet


class HierarchicalABE:
    def __init__(self):
        self.keys = {}            # Menyimpan kunci pengguna
        self.attributes = {}      # Menyimpan atribut pengguna
        self.data_store = {}      # Menyimpan data terenkripsi
        self.policies = {}        # Menyimpan kebijakan akses data

    # Membuat kunci dan menetapkan atribut pengguna
    def generate_key(self, user, attributes):
        key = Fernet.generate_key()
        self.keys[user] = key
        self.attributes[user] = attributes
        return key

    # Cek apakah atribut pengguna memenuhi kebijakan
    def policy_check(self, user_attrs, required_attrs):
        return all(attr in user_attrs for attr in required_attrs)

    # Enkripsi dengan kebijakan akses (daftar atribut yang diperlukan)
    def encrypt(self, user, data, policy):
        if user not in self.keys:
            raise Exception("Kunci pengguna tidak ditemukan.")
        fernet = Fernet(self.keys[user])

        # Proses enkripsi
        start_time = time.time()
        encrypted_data = fernet.encrypt(data.encode())
        end_time = time.time()

        # Simpan data & kebijakan akses
        self.data_store[user] = encrypted_data
        self.policies[user] = policy

        print(f"Waktu enkripsi untuk {user} : {end_time - start_time:.6f} detik")
        return encrypted_data

    # Dekripsi hanya jika atribut user memenuhi kebijakan
    def decrypt(self, requester, owner):
        if requester not in self.keys:
            raise Exception("Kunci pengguna tidak ditemukan.")

        # Ambil data dan kebijakan
        encrypted_data = self.data_store.get(owner)
        policy = self.policies.get(owner)
        requester_attrs = self.attributes.get(requester, [])

        if not encrypted_data:
            raise Exception("Data tidak ditemukan.")
        if not self.policy_check(requester_attrs, policy):
            raise Exception("Akses ditolak: atribut tidak memenuhi kebijakan.")

        # Dekripsi
        fernet = Fernet(self.keys[owner])
        start_time = time.time()
        decrypted_data = fernet.decrypt(encrypted_data).decode()
        end_time = time.time()

        print(f"Waktu dekripsi untuk {requester} : {end_time - start_time:.6f} detik")
        return decrypted_data


# Inisialisasi objek HABE
abe = HierarchicalABE()

# Buat pengguna beserta atributnya
abe.generate_key("HR_Manager", ["role:manager", "department:HR"])
abe.generate_key("IT_Developer", ["role:developer", "department:IT"])

# Enkripsi informasi rahasia dengan kebijakan akses
plain_text = "Informasi Rahasia dari HR Manager"
policy = ["role:manager", "department:HR"]  # hanya bisa diakses oleh yang memenuhi dua atribut
ini
print("Real Information:", plain_text)

encrypted_data = abe.encrypt("HR_Manager", plain_text, policy)
print("Cipher Text:", encrypted_data)

# HR_Manager mencoba dekripsi (berhasil)
print("\n[DEKRIPSI oleh HR_Manager]:")
try:
    decrypted_data = abe.decrypt("HR_Manager", "HR_Manager")
    print("Data hasil dekripsi:", decrypted_data)
except Exception as e:
    print(e)

# IT_Developer mencoba dekripsi (gagal karena tidak punya atribut yang sesuai)
print("\n[DEKRIPSI oleh IT_Developer]:")
try:
    abe.decrypt("IT_Developer", "HR_Manager")
except Exception as e:
    print("Gagal:", e)
```



```
# IT_Developer mencoba dekripsi (gagal karena tidak punya atribut yang sesuai)
print("\n[DEKRIPSI oleh IT_Developer]:")
try:
    abe.decrypt("IT_Developer", "HR_Manager")
except Exception as e:
    print("Gagal:", e)
```

```
Real Information: Informasi Rahasia dari HR Manager
Waktu enkripsi untuk HR_Manager : 0.000230 detik
Cipher Text: b'gAAAAABoOubpNdmgae2m-1oxF8zo71YFGvWEVCGO_dFOzeAMvHuzCSbVDeoZYhGD6mZVE2-AQRLpkcYi16UnI5Ww7SU6faH0-RurBfRJbdfVhIb4dozJGKiF2Rtk7slbab6P9w5NRlqn'

[DEKRIPSI oleh HR_Manager]:
Waktu dekripsi untuk HR_Manager : 0.000097 detik
Data hasil dekripsi: Informasi Rahasia dari HR Manager

[DEKRIPSI oleh IT_Developer]:
Gagal: Akses ditolak: atribut tidak memenuhi kebijakan.
```

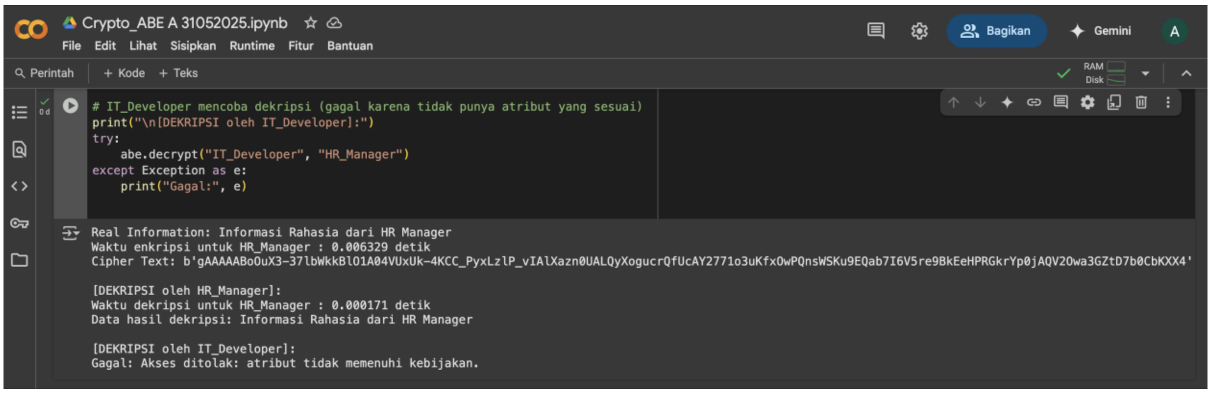
Figure 4. HABE Encryption and Decryption Results

```
Real Information: Informasi Rahasia dari HR Manager
Waktu enkripsi untuk HR_Manager : 0.000230 detik
Cipher     Text:     b'gAAAAABoOubpNdmgae2m-1oxF8zo71YFGvWEVCGO_dFOzeAMvHuzCSbVDeoZYhGD6mZVE2-
AQRLpkcYi16UnI5Ww7SU6faH0-RurBfRJbdfVhIb4dozJGKiF2Rtk7slbab6P9w5NRlqn'

[DEKRIPSI oleh HR_Manager]:
Waktu dekripsi untuk HR_Manager : 0.000097 detik
Data hasil dekripsi: Informasi Rahasia dari HR Manager

[DEKRIPSI oleh IT_Developer]:
Gagal: Akses ditolak: atribut tidak memenuhi kebijakan.
```

Using the same role, the implementation using ABE is shown using the Python programming language below.

```
import time
from cryptography.fernet import Fernet

class AttributeBasedEncryption:
    def __init__(self):
        self.keys = {}              # Menyimpan kunci pengguna
        self.attributes = {}        # Menyimpan atribut pengguna
        self.data_store = {}        # Menyimpan data terenkripsi
        self.policies = {}          # Menyimpan kebijakan akses data

    # Membuat kunci dan menetapkan atribut pengguna
    def generate_key(self, user, attributes):
        key = Fernet.generate_key()
        self.keys[user] = key
        self.attributes[user] = attributes
        return key

    # Cek apakah atribut pengguna memenuhi kebijakan
    def policy_check(self, user_attrs, required_attrs):
        return all(attr in user_attrs for attr in required_attrs)
    # Enkripsi dengan kebijakan akses (daftar atribut yang diperlukan)
    def encrypt(self, user, data, policy):
        if user not in self.keys:
            raise Exception("Kunci pengguna tidak ditemukan.")
        fernet = Fernet(self.keys[user])

        # Proses enkripsi
        start_time = time.time()
        encrypted_data = fernet.encrypt(data.encode())
        end_time = time.time()

        # Simpan data & kebijakan akses
        self.data_store[user] = encrypted_data
        self.policies[user] = policy

        print(f"Waktu enkripsi untuk {user} : {end_time - start_time:.6f} detik")
        return encrypted_data

    # Dekripsi hanya jika atribut user memenuhi kebijakan
    def decrypt(self, requester, owner):
        if requester not in self.keys:
            raise Exception("Kunci pengguna tidak ditemukan.")

        # Ambil data dan kebijakan
        encrypted_data = self.data_store.get(owner)
        policy = self.policies.get(owner)
        requester_attrs = self.attributes.get(requester, [])

        if not encrypted_data:
            raise Exception("Data tidak ditemukan.")

        if not self.policy_check(requester_attrs, policy):
            raise Exception("Akses ditolak: atribut tidak memenuhi kebijakan.")

        # Dekripsi
        fernet = Fernet(self.keys[owner])
        start_time = time.time()
        decrypted_data = fernet.decrypt(encrypted_data).decode()
        end_time = time.time()

        print(f"Waktu dekripsi untuk {requester} : {end_time - start_time:.6f} detik")
        return decrypted_data


# Inisialisasi objek ABE
abe = AttributeBasedEncryption()
```

```
# Buat pengguna beserta atributnya
abe.generate_key("HR_Manager", ["role:manager", "department:HR"])
abe.generate_key("IT_Developer", ["role:developer", "department:IT"])

# Enkripsi informasi rahasia dengan kebijakan akses
plain_text = "Informasi Rahasia dari HR Manager"
policy = ["role:manager", "department:HR"]  # hanya bisa diakses oleh yang memenuhi dua atribut
ini
print("Real Information:", plain_text)

encrypted_data = abe.encrypt("HR_Manager", plain_text, policy)
print("Cipher Text:", encrypted_data)

# HR_Manager mencoba dekripsi (berhasil)
print("\n[DEKRIPSI oleh HR_Manager]:")
try:
    decrypted_data = abe.decrypt("HR_Manager", "HR_Manager")
    print("Data hasil dekripsi:", decrypted_data)
except Exception as e:
    print(e)

# IT_Developer mencoba dekripsi (gagal karena tidak punya atribut yang sesuai)
print("\n[DEKRIPSI oleh IT_Developer]:")
try:
    abe.decrypt("IT_Developer", "HR_Manager")
except Exception as e:
    print("Gagal:", e)
```



```
# IT_Developer mencoba dekripsi (gagal karena tidak punya atribut yang sesuai)
print("\n[DEKRIPSI oleh IT_Developer]:")
try:
    abe.decrypt("IT_Developer", "HR_Manager")
except Exception as e:
    print("Gagal:", e)
```

```
Real Information: Informasi Rahasia dari HR Manager
Waktu enkripsi untuk HR_Manager : 0.006329 detik
Cipher Text: b'gAAAAABoOuX3-37lbWkkBlO1A04VUxUk-4KCC_PyxLzlP_vIAlXazn0UALQyXogucrQfUcAY2771o3uKfxOwPQnsWSKu9EQab7I6V5re9BkEeHPRGkrYp0jAQV2Owa3GZtD7b0CbKXX4'

[DEKRIPSI oleh HR_Manager]:
Waktu dekripsi untuk HR_Manager : 0.000171 detik
Data hasil dekripsi: Informasi Rahasia dari HR Manager

[DEKRIPSI oleh IT_Developer]:
Gagal: Akses ditolak: atribut tidak memenuhi kebijakan.
```

Figure 5. The Results of ABE Encryption and Decryption

```
Real Information: Informasi Rahasia dari HR Manager
Waktu enkripsi untuk HR_Manager : 0.006329 detik
Cipher                    Text:                    b'gAAAAABoOuX3-37lbWkkBlO1A04VUxUk-
4KCC_PyxLzlP_vIAlXazn0UALQyXogucrQfUcAY2771o3uKfxOwPQnsWSKu9EQab7I6V5re9BkEeHPRGkrYp0jAQV2Owa3
GZtD7b0CbKXX4'

[DEKRIPSI oleh HR_Manager]:
```

```
Waktu dekripsi untuk HR_Manager : 0.000171 detik
Data hasil dekripsi: Informasi Rahasia dari HR Manager

[DEKRIPSI oleh IT_Developer]:
Gagal: Akses ditolak: atribut tidak memenuhi kebijakan.
```

The encryption process is carried out by the data owner and is only permitted after the user possesses a valid key. Access policies are defined as a list of attributes required to access the data, `["role:manager", "department:HR"]`. The information is then encrypted using the user's key, and the policy is stored along with the encrypted data in cloud storage.

Plaintext M = "Informasi Rahasia dari HR Manager"

Access Policy:
$P(x)$ = "role:manager" ∧ "department:HR"

User attributes:
- HR_Manager: A={role:manager,department:HR}
- IT_Developer: A={role:developer,department:IT}

The steps for implementing HABE are as follows,

1. System setup by RA, defining public parameters and master key

- Random prime numbers: p
- Bilinear Group: G_1, G_T dengan orde p
- Pairing: $e: G_1 \times G_1 \rightarrow G_T$
- Generator: $g \in G_1$
- Choose: $\alpha, y \in Zp$ secara acak
- H: fungsi hash H: Attribute→G_1

Public Key:
$$PK = (g, h = g^y, f = g^{1/\alpha}, e(g,g)^\alpha)$$

Master Secret Key:
$$MSK = (\alpha, y)$$

2. Key Generation for Users (HR_Manager)

User has the attribute:
A = {a_1 = role:manager, a_2 = department:HR}
Choose a random number $r \in Zp$ and for each attribute a_i, choose $ri \in Zp$

For each $a_I \in A$, count:

$$D = g^{(\alpha+r)/y} \in G$$

$D_i = g^r \cdot H(a_i)^{r_i} \in G_1$
$D_i' = g^{r_i} \in G_1$

3. Message Encryption M

- Receive the message $M \in GT$
- Choose a random number $s \in Zp$
- Create an access tree structure based on $P = a_1 \wedge a_2$
- Build a polynomial qx at each node x from tree
  Root: $q_{root}(0) = s$
- For each leaf (atribut a), count:

$$C_a = g^{q_a(0)} \in G_1$$
$$C'_a = H(a)^{q_a(0)} \in G_1$$

Ciphertext Result:

$$CT = (P, C = M \cdot e(g,g)^{\alpha s}, C' = g^s, \{C_a, C'_a\}_{a \in atribut\ pada\ P})$$

Table 1. Performance of HABE and ABE

Variabel	HABE (s)	ABE (s)
Encryption	0.000230	0.006329
Decryption	0.000097	0.000171

CONSLUSION

The application of Hierarchical Attribute-Based Encryption (HABE) demonstrates more time efficiency than traditional Attribute-Based Encryption (ABE), according to the findings of performance testing of the encryption and decryption algorithms. HABE's encryption procedure takes 0.000230 seconds, which is significantly less time than ABE's 0.006329 seconds. In a similar vein, HABE outperformed ABE in the decryption process, with a time of 0.000097 seconds as opposed to 0.000171 seconds.

These findings show that HABE performs exceptionally well in terms of cryptographic processing time efficiency in addition to maintaining a hierarchical attribute-based access control mechanism. As a result, HABE is ideal for deployment in systems that need to prioritize performance while managing complex data access, including large-scale data management systems or cloud environments based on organizations.

REFERENCES

[1] J. Wu, L. Ping, X. Ge, W. Ya, and J. Fu, "Cloud storage as the infrastructure of Cloud Computing," in *Proceedings - 2010 International Conference on Intelligent Computing and Cognitive Informatics, ICICCI 2010*, 2010, pp. 380–383. doi: 10.1109/ICICCI.2010.119.

[2] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Cloud Computing Security and Privacy Challenges in Cloud Computing Environments," 2010. [Online]. Available: http://csrc.nist.gov

[3] S. Paudel, "Data Breach a Cyber Security Issue in Cloud Data Breach a Cybersecurity Issue in Cloud." [Online]. Available: https://www.researchgate.net/publication/335243297

[4] A. R. Khan, "Access Control In Cloud Computing Environment," vol. 7, no. 5, 2012, [Online]. Available: www.arpnjournals.com

[5] J. Li, Q. Wang, C. Wang, and K. Ren, "Enhancing Attribute-Based Encryption with Attribute Hierarchy."

[6] G. Wang and J. Wu, "Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services," Chicago, Illinois, USA: Association for Computing Machinery, Oct. 2010, p. 763. doi: 978-1-4503-0244-9/10/10.

[7] G. Wang, Q. Liu, J. Wu, and M. Guo, “Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers,” *Computers and Security*, vol. 30, no. 5, pp. 320–331, Jul. 2011, doi: 10.1016/j.cose.2011.05.006.

[8] P. Kumar Premkamal, S. Kumar Pasupuleti, and A. P.J.A., “Attribute based encryption in cloud computing: A survey, gap analysis, and future directions,” *Journal of Network and Computer Applications*, vol. 108, pp. 37–52, Apr. 2018, doi: 10.1016/j.jnca.2018.02.009.

[9] P. J. Sun, “Security and privacy protection in cloud computing: Discussions and challenges,” *Journal of Network and Computer Applications*, vol. 160, Jun. 2020, doi: 10.1016/j.jnca.2020.102642.

[10] M. Kiruthika and R. Mohanabharathi, “A Secured File Store in Cloud Environment Using Hierarchy Attribute-Based Encryption,” *International Journal of Advanced Engineering Research and Science*, vol. 3, no. 11, pp. 110–114, 2016, doi: 10.22161/ijaers/3.11.19.

[11] M. Ali, J. Mohajeri, M. R. Sadeghi, and X. Liu, “A fully distributed hierarchical attribute-based encryption scheme,” *Theoretical Computer Science*, vol. 815, pp. 25–46, May 2020, doi: 10.1016/j.tcs.2020.02.030.

[12] M. Ali, J. Mohajeri, and M.-R. Sadeghi, “On the security of the hierarchical attribute based encryption scheme proposed by Wang et al,” Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.05864