Secure Data Sharing in a Cyber-Physical Cloud Environment

Jun-Wen Chan
Multimedia University, Malaysia

Swee-Huay Heng
Multimedia University, Malaysia

Syh-Yuan Tan
Multimedia University, Malaysia

Abstract: Cloud computing plays a significant role in digital workplaces and has become an integral part of everyday life. However, the security issues associated with cloud computing remain a major concern that hinders the wider adoption of this technology. In a cyber-physical cloud environment, achieving secure and efficient file storage remains a tough goal. This is particularly the case owing to the wide variety of devices that are being utilised to access the various services and data. Thus, the employment of a secure data sharing protocol is one of the essential techniques to better protect the shared data. A formal agreement between entities or organisations in exchanging personal or business data is called a data sharing protocol. A secure data sharing protocol ensures that data is encrypted and secured when being transferred. There are many different encryption algorithms and protocols in use to devise various secure and efficient data sharing protocols. This paper first reviews the state of the art of some existing data sharing protocols and subsequently implements a secure and efficient data sharing protocol as an application in a cyber-physical cloud environment. The performance analysis conducted on the developed secure data sharing protocol application shows positive results.

Keywords: cloud computing, data sharing protocol, cyber-physical environment

Introduction

The term 'cyber-physical cloud systems' (CPCS) refers to a technology that has a wide range of applications, some of which include healthcare, smart electricity grids, smart cities, battlegrounds, and the military. Client devices such as those running on Android or iOS, or devices with restricted resource availability such as sensors, are being deployed in these types of systems to gain access to services (Deng *et al.*, 2014). In the context of industry, CPCS is

http://doi.org/10.18080/jtde.v11n4.772

used for collecting, processing, analysing and interpreting large data. The utilisation of Internet of Things technologies in conjunction with real-time analysis of vast amounts of data allows for large-scale monitoring opportunities, thanks to the prevalence of multifaceted devices. This generates novel insights that can enhance decision-making processes and provide an advantageous edge in business (Cheng *et al.*, 2018). However, compared to typical personal computers, client devices sometimes have much less processing capabilities. Recent years have witnessed a rapid and vast adoption of mobile cloud computing as compared to the conventional cloud computing since smart mobile devices such as Android and iOS smartphones have become more prevalent (Salvi, 2019).

Nevertheless, there are still significant concerns regarding security issues such as reliability and privacy in physical devices and in untrusted cloud environments, despite the widespread acceptance of cloud computing in its varied forms. Safe and reliable information exchange between users of a hybrid cloud (cyber and physical) is fortunately made possible through a cryptographic approach known as data sharing protocol (Salvi, 2019). However, a protocol is developed with complex algorithms and, as mobile devices' processors are not very powerful, this will take up a lot of computation time. Therefore, lightweight operation is essential for efficiency and faster computation in mobile devices.

This research reviews the existing data sharing protocols in a cyber-physical cloud environment. Leveraging the strengths of the state-of-the-art protocol, the application will be developed by incorporating the chosen data sharing protocol. The application ensures heightened security and efficiency and promises fast computation times in a cyber-physical cloud environment. The performance analysis attests that the data sharing protocol facilitates a rapid and reliable approach. Thus, this research paves the way for future development in secure and efficient data sharing solutions.

Table 1 contains the list of abbreviations used throughout our discussion.

Table 1. List of abbreviations

LIST OF ABBREVIATIONS/SYMBOLS	
AES	Advanced encryption standard
CC	Cloud controller
CS	Cloud server
DC	Data consumer
DCs	Data consumers
dDHPEKS	Designated-tester decryptable hierarchical public key encryption with keyword search
DH	Data holder
DO	Data owner
DS	Data sharer
HPEKS	Hierarchical public key encryption with keyword search
IBADS	Identity-based authenticated data sharing

http://doi.org/10.18080/jtde.v11n4.772

LIST OF ABBREVIATIONS/SYMBOLS	
IBE	Identity-based encryption
KB	Kilobyte
MC_c	Client
OO-ABPRE	Online/offline attribute-based proxy re-encryption
PEKS	Public key encryption with keyword search
PKG	Private key generator
PKTree	Public key tree
S_{CC}	ID of cloud controller
SSGK	Secret sharing group key management

Current Data Sharing Protocols

Shao *et al.* (2015) proposed a fine-grained data sharing protocol employed in cloud computing through utilising the transformed key approach and online/offline attribute-based proxy re-encryption (OO-ABPRE). The proposed approach protects user data by allowing fine-grained access control, convenient sharing and is also cost-effective. The protocol uses a proxy that is only partially trusted and is equipped with a re-encryption key. As a result, the proxy is able to transform a plaintext that was originally encrypted with one public key into a plaintext that was encrypted with another public key. In other words, this enables encryption to be performed by using the public key of the data holder (DH) or someone else. When the DH decides to engage in sharing the data, all he has to do is produce the relevant re-encryption key. The cloud server (CS) will then be able to use this re-encryption key to transform the ciphertext into a form that the data sharer (DS) will be able to decrypt by using his corresponding private key.

In order to safeguard the information kept in cloud environments, an identity-based authenticated data sharing (IBADS) protocol was designed by Karati *et al.* (2018). This protocol ensures user anonymity which enables the identities of the client and user to be concealed from the attackers, even in the event that they intercepted the public channel over which the message was transmitted. This end-to-end connection will be encrypted using a technique that relies on a small public parameter and bilinear pairing once the authentication of physical devices has been completed. The protocol is distinguished by its deployment of identity-based encryption (IBE) to facilitate the safe exchange of data between the clients and the clients' geographically scattered physical devices.

Han *et al.* (2019) proposed a secret sharing group key management protocol (SSGK) for resolving the issue of increasing security and privacy risks in cloud storage. It employs symmetric encryption algorithms to encrypt the shared data, ensuring its usability by authorised users. The data owner (DO) distributes decryption keys to authorised sharers. Decryption keys control permissions for accessing shared data. Only legitimate participants

http://doi.org/10.18080/jtde.v11n4.772

can decrypt the key associated with the interactive message due to asymmetric encryption algorithms. If unauthorised users obtain access to shared data, a secret sharing scheme assigns the key to legitimate participants.

Lu *et al.* (2020) proposed a data sharing scheme to ensure sensitive data will be shared in a secure and authorised manner. To avoid inaccurate computations, this protocol provides an efficient full validation before users share the data. Using this protocol, sensitive data can be protected during the sharing procedure and authorisation for access can be controlled by the data requester.

Li *et al.* (2022) proposed a protocol for enterprise users that supports hierarchical keyword searching to facilitate secure data sharing in a cloud environment. There are two types of public key encryption with keyword search (PEKS) employed in the protocol, namely, hierarchical public key encryption with keyword search (HPEKS) and designated-tester decryptable hierarchical public key encryption with keyword search (dDHPEKS) which is more advanced than HPEKS. This protocol addresses the challenges of secure data sharing in an enterprise environment. In this protocol, advocated encrypted data should be hierarchically accessible and searchable. To manage the hierarchical structure of users in an enterprise, public key tree (PKTree) was developed. The PKTree algorithm pairs elements from two cryptographic groups to construct complex cryptographic protocols. Users can find the ciphertext encrypted with their public key using HPEKS. Users with lower access permissions can search for ciphertexts sent to a hierarchical group of users. Using this feature in an enterprise setting is particularly beneficial in cases where higher-level employees are required to monitor the data of lower-level employees. dDHPEKS combines symmetric and public key encryption in an advanced version of HPEKS. In addition to preventing outside and offline keyword-guessing attacks, it provides keyword search and decryption functionality. Sharing encrypted data with others does not require knowledge of the enterprise's internal hierarchy due to transparency in the scheme (Li *et al.*, 2022).

Comparison analysis among data sharing protocols

In general, the primary goals of employing the respective data sharing protocols (Shao *et al.*, 2015; Karati *et al.*, 2018; Han *et al.*, 2019; Lu *et al.*, 2020; Li *et al.*, 2022) are to ensure data is encrypted and securely shared, especially in an untrusted cloud environment. Table 2 presents a comparison analysis of the reviewed data sharing protocols.

http://doi.org/10.18080/jtde.v11n4.772

Table 2. Comparison analysis among data sharing protocols

Characteristic	Protocol				
	Shao *et al.* (2015)	Karati *et al.* (2018)	Han *et al.* (2019)	Lu *et al.* (2020)	Li *et al.* (2022)
Underlying cryptographic scheme(s)	Linear secret sharing and access policy schemes Hash function OO-ABPRE	One-way hash function IBE	Symmetric and asymmetric encryption Secret sharing scheme	Linear secret sharing schemes XOR-homomorphic function Algebraic signature	Public key encryption with keyword search
Encryption and decryption method	CS uses the re-encryption key from DH to re-encrypt data and uses DS's transform key to return the transformed ciphertext; DS decrypts using the private key	Encrypt using the public ID as public key; decrypt using the private key of the corresponding ID	Data shared is encrypted with symmetric encryption Asymmetric encryption is used to encrypt the interactive message	Encrypt data using private key generated by the DO; decrypt using private key requested from key generator centre	Encrypted with public key and allows keyword search without decryption
Advantage	Policy access could be inserted to ciphertext Only users with attribute sets that conform to the access policy can decrypt	Provide secure end-to-end communication Designed to be lightweight and has good performance runtime execution	Control permission for shared data Ensure that the key required for decrypting the shared data is not visible to unauthorised users	Authorised data requester can correctly recover the shared data	Sharing encrypted data with an enterprise does not require the sender to know the enterprise's internal hierarchy
Disadvantage	When generating the re-encryption key, DH's device needs to be charged	Complex protocol	Not obvious	Costing is more expensive using cloud storage server and cloud-managed server	dDHPEKS scheme is slower than HPEKS scheme

System Design of Data Sharing Protocol Application

Based on the review and comparison analysis among some existing data sharing protocols as summarised above, the IBADS protocol (Karati *et al.*, 2018) is chosen for implementation because it is a lightweight protocol, meaning it can perform lightweight encryption on a mobile device. The user's mobile device does not spend much time performing encryption, thus speeding up this process. The context diagram depicted in Figure 1 shows the overview of the entity structure of our proposed secure data sharing protocol application. According to the diagram, the external entities are identified as DO, data consumer (DC), cloud controller (CC) and private key generator (PKG). They interact following the processes with the data flow. The DO encrypts and uploads the file to the cloud via the system. The DC could receive the encrypted file and decrypt it while receiving the private key sent from the PKG.

http://doi.org/10.18080/jtde.v11n4.772

Figure 2 shows the use case diagram. Both DO and DC need to register an account with their phone numbers and emails. After that, they need to do mutual authentication with the CC to verify the authenticity of each other. Once mutual authentication has been completed, the DO needs to encrypt the file with the parameters and then upload the file to the cloud. The DC can search files uploaded by the DO by searching the relevant keyword in the system. Following that, the DC can request to download a file from the CC and the CC sends the encrypted file to the DC. Upon receiving the encrypted file, the DC requests the private key from the PKG in order to decrypt the encrypted file. The private key is then used to decrypt the file by the DC.

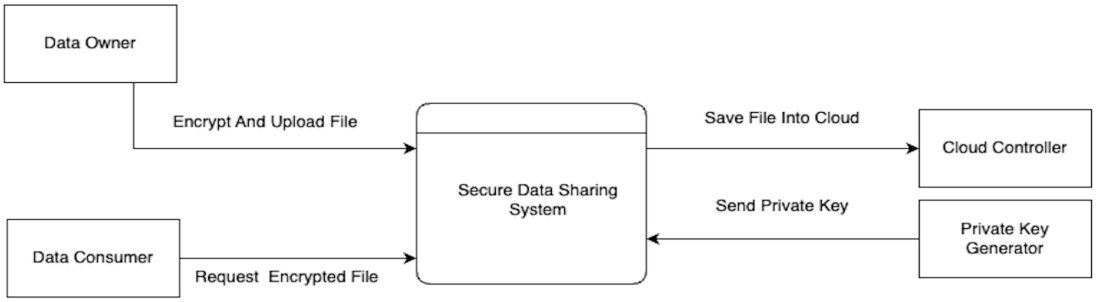


Figure 1. Context diagram

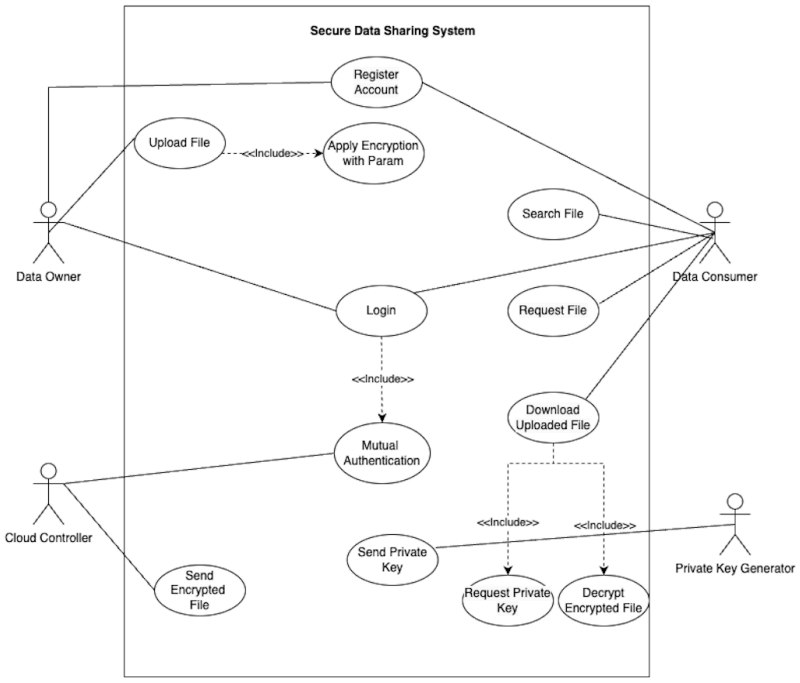


Figure 2. Use case diagram

Implementation of Data Sharing Protocol Application

According to Karati *et al.* (2018), the IBADS protocol uses an IBE scheme and symmetric encryption to perform secure data sharing which provides mutual authentication in order for the CC and user to verify each other's authenticity. The IBE scheme in IBADS protocol uses

http://doi.org/10.18080/jtde.v11n4.772

the device number and mobile number of the DC as the public key to perform encryption. This makes it unnecessary for the DO to ask for the public key from the DC when encrypting files to be sent, thus shortening the time and process of encryption.

Mutual authentication algorithm

Mutual authentication is a technique that discovers if cloud users (DOs and DCs) are valid (non-revoked registered users). It therefore runs through the phases below: registration of client, login and mutual authentication, and password renewal (Karati *et al.*, 2018).

- **Registration of client:** The client MC_c decides a unique identity ID_c and returns it with the identity of email and mobile number to the CC. Knowing that the ID_c parameter is securely sent to the CC, and after the information is received, the CC executes these steps:
  - Creates an app by storing $A_c = h(ID_c \| S_{CC}) \oplus h(ID_c \| PW_c)$, where S_{CC} is the secret key of CC and PW_c is the password of MC_c.
  - Sends the URL link is securely to the email of the MC_c and (PW_c, W_c) to the MC_c mobile number, where W_c is the information generated randomly.
  - Updates the user list LU and stores W_c as the public and unique information of client's email TID_c.

  After receiving the URL link, the MC_c installs software to the mobile device. After the installation is done, the MC_c runs the software and gives ID_c, PW_c and W_c. An array of groups with the group identity G_{ID_j} is received by the mobile device as it is connected to the internet. Now, the MC_c chooses the group fulfilling its requirement and demands as V. The data W_c can only be given once. After that, MC_c runs the below operations:

  - Extracts $B_c = A_c \oplus h(ID_c \| PW_c)$ and new password is asked to be inputted to the MC_c.
  - After receiving the new password PW_c^{new}, it computes

    $$A_c^{new} = B_c \oplus h(ID_c \| PW_c^{new}), D_c = h(ID_c \oplus PW_c^{new})$$

    and $E_c = W_c \oplus h(PW_c^{new})$.
  - Stores $\langle A_c^{new}, D_c, E_c, TID_c \rangle$ and drops A_c.
- **Login and mutual authentication:** This phase is imperative during CC login and the purpose is to execute key agreement and mutual authentication. The phase is defined as follows.

  The MC_cexecutes the software and generates ID_c and PW_c^{new}, after which the software runs the following operations:

  - Calculates $D_c^* = h(ID_c \oplus PW_{new})$ and checks whether $\left(D_c^* \stackrel{?}{=} D_c\right)$ is true.

http://doi.org/10.18080/jtde.v11n4.772

- If $(D_c^* \neq D_c)$, then the session is aborted.
- Otherwise,
  - Random number R_c is generated.
  - Calculates

    $$B_c = A_c^{new} \oplus h(ID_c \parallel PW_c^{new}), W_c = E_c \oplus h(PW_c^{new}), F_c = h(ID_c \| B_c \| R_c)$$

    and then encrypts ID_c and R_c using W_c as the secret key to generate $G_c = E_{W_c(ID_c \| R_c)}$.
  - Forwards through secure channel $\langle TID_c, G_c, F_c \rangle$ to the CC.

After receiving $\langle TID_c, G_c, F_c \rangle$, parameter TID_c is then searched by the CC first in the database and if it returns as false, the session aborts immediately, otherwise it will extract W_c. The CC then runs following operations:

- Decrypts G_c to get ID_c and R_c.
- Calculates

  $$B_c^* = h(ID_c \parallel S_{CC}), F_c^* = h(ID_c \| B_c^* \| R_c)$$

  and determines if $\left(F_c^* \stackrel{?}{=} F_c\right)$ is true.
- If $(F_c^* \neq F_c)$, then the MC_c is denied access.
- Otherwise,
  - A random number R_{cs} is generated.
  - Computes the session key

    $$SK_{cs} = h(ID_c \parallel R_{cs} \parallel R_c), K_c = h(ID_c \| SK_{cs} \| R_{cs}), R_{ccs} = R_c \oplus R_{cs}.$$
  - Forwards $\langle K_c, R_{ccs} \rangle$ to the MC_c.

On receiving $\langle K_c, R_{ccs} \rangle$, the MC_c performs the following operations:

- Extorts $R_{cs}^* = R_{ccs} \oplus R_c$.
- Computes

  $$SK_c^* = h(ID_c \| R_{cs}^* \| R_c), K_c^* = h(ID_c \parallel SK_c^* \parallel R_{cs}).$$
- If $\left(K_c^* \stackrel{?}{=} K_c\right)$ is true, then
  - The protocol achieves mutual authentication as the CC is authenticated and session key is verified.

- **Password renewal:** This is for valid users to renew their password. Firstly, the MC_c runs the installed software and generates ID_c and PW_c^{new}. Following this, the software executes the following tasks:
  - Calculates $D_c^* = h(ID_c \oplus PW_{new})$.

http://doi.org/10.18080/jtde.v11n4.772

- Determines if $\left(D_c^* \stackrel{?}{=} D_c\right)$ if true. If $(D_c^* = D_c)$, it then aborts.
- Otherwise,
  - A new password to the $MC_c$is requested.
  - After receiving a new password PW_c^* , it calculates

$$B_c = A_c \oplus h\left(ID_c \parallel PW_c^{new}\right), A_c^* = B_c \oplus h(ID_c \parallel PW_c^*),\ D_c *= h(ID_c \oplus PW_c^*)$$

and

$$E_c *= W_c \oplus h(PW_c^*).$$

Finally, it records the information $\langle A_c^*, D_c^*, E_c^* \rangle$, and deletes $\langle A_c^{new}, D_c, E_c \rangle$ from the mobile device saved by the MC_c.

IBE scheme

This IBE scheme consists of four algorithms: *IBE.Setup*, *IBE.Extract*, *IBE.Encrypt* and *IBE.Decrypt* (Karati *et al.*, 2018). *IBE.Setup* is an algorithm for generating the parameters that need to be used in the three other IBE algorithms for calculation purposes; *IBE.Extract* is an algorithm which generates the private key for decrypting the message, and *IBE.Encrypt* and *IBE.Decrypt* are the algorithms used to encrypt and decrypt messages, respectively.

- **IBE.Setup (1^k):** The PKG executes the protocol by inputting a security parameter 1^k. After that, it creates a prime p and a bilinear map, e: $G_1 \times G_1 \rightarrow G_2$ of two multiplicative groups $G_1$and G_2. Then, it picks a generator $g \in_R G_1$ randomly and executes $h = \left(g^{\beta}\right)$ with random $\beta \in_R Z_p^*$. Then, it computes $Y = e(g, g)^{\alpha}$ with $\alpha \in_R Z_p$. At last, it picks a one-way cryptographic hash function $H: \{0,1\}^* \rightarrow Z_p$. It then publishes the parameters, params = $\langle g, h, Y, H \rangle$ and keeps MSK = $\langle \alpha \rangle$.
- **IBE.Extract (MN_i, UN_i, params, MSK):** After verifying the mobile number MN_i and device number UN_i properly, the PKG computes

$$r_i = \frac{\alpha}{\beta + ID_i};\quad K_i^{(1)} = g^{r_i} \quad t_i = \frac{\alpha}{\beta + GID_i};\ K_i^{(2)} = g^{t_i}$$

where $ID_i = H(MN_i \parallel UN_i)$, M. Then, it generates the private key $SK_i = (K_i^{(1)}, K_i^{(2)})$ and sends it via a secure network. $K_i^{(1)}$ is the private key for the individual user to decrypt the message while $K_i^{(2)}$ is the private key for decrypting the group message.

- **IBE.Encrypt (M, U, V, params):** The protocol receives the $M \in_R G_2$ and U as the client, where U contains the user's set IDs to publish the message by the recipient. After that, it picks $s_1 \in_R Z_p^*$ and computes $Y1 = (Y^{-s_1})$. It then executes the following operations:

  a) For every client $i \in U$,

http://doi.org/10.18080/jtde.v11n4.772

- Computes $ID_i = H(MN_i \parallel UN_i)$.
- Computes $T_i = \left(\left(g^{ID_i} \cdot h\right)^{s_1}\right)$, $ID_i = H(MN_i \parallel UN_i)$ which is a public key used to encrypt the data.
- Sets $T = T \cup \{T_i\}$ (Initially $T = NULL$).
- Computes $s_2 = H(Y1 \parallel T)$ and $C_U = \left((M \,.\, Y1)^{\frac{1}{s_2}}\right)$.

b) For every group $j \in V$,

- Computes $W_i = \left(\left(g^{GID_j} \cdot h\right)^{s_1}\right)$.
- Sets $= W \cup W_i$ (Initially $W = NULL$).
- Computes $s_3 = H(Y1 \parallel W)$ and $C_V = \left((M \cdot Y1)^{\frac{1}{s_3}}\right)$.

Then, it computes

$$MD = H(IDs \parallel M \parallel C_U \parallel C_V \parallel T \parallel W) \text{ where } ID_s = H(MN_s \parallel UN_s).$$

Finally, it computes the ciphertext $CT = \{C_U, C_V, T, W, MD\}$.

- **IBE.Decrypt** $(\mathbf{CT}, \mathbf{ID_s}, \mathbf{params}, \mathbf{SK_i})$: The ciphertext CT is to be decrypted for a user i. This protocol runs the following operations:

a) Not a group message,

- Computes $Z_1 = e\left(T_i, K_i^{(1)}\right)$ and $Y1' = Z_1^{-1}$.
- Computes $s_2' = H(Y1' \parallel T)$ and $C' = C_U^{s_2}$.
- Computes $M' = (C' \cdot Z_1)$.

b) Otherwise,

- Computes $Z_2 = e\left(W_i, K_i^{(2)}\right)$ and $Y2' = Z_2^{-1}$.
- Computes $s_3' = H(Y2' \parallel W)$ and $C' = C_v^{s_3}$.
- Computes $M' = (C' \cdot Z_2)$.

If $MD \stackrel{?}{=} H(ID_s \parallel M')$ is true, this algorithm will return M'; otherwise, it is NULL. In the *IBE.Decrypt*, the user computes this algorithm and gets the value of $Y1'$, s_2' and $C'$to be the same as $Y1$, s_2 and $M \cdot Y1$ in *IBE.Encrypt*, respectively, which means that the step of retrieving the data is correct.

Implementation of IBADS protocol

The data sharing protocol is developed using the Java Pairing-Based Cryptography Library (De Caro & Iovino, 2011) and the Java Cryptography packages which include java.security and javax.crypto (*Java Platform, Standard Edition Security Developer's Guide*, n.d.).

http://doi.org/10.18080/jtde.v11n4.772

- **System initiation**

  Users must register their account before using the secure data sharing application. After the registration is completed, users will receive an email sent from the CC. The email will provide the link for users to download the application, first time login password and Wc value. They then need to provide their user ID and password to perform mutual authentication with the CC. If users are first time login, they are required to do the first-time setup which is to reset their password and provide the Wc value. Then, the system will generate params for executing the algorithms of mutual authentication. Subsequently, users provide their user ID and password to the system to execute mutual authentication. If users provide the correct information, then they will successfully log in.

- **File encrypting and sharing**

  After the DO logs in successfully, the DO needs to provide a keyword which serves as an index for the file to be encrypted. The DO needs to select the DC as U and the group as V in order to share the data with them. After that, DO performs *IBE.Encrypt*. The *IBE.Encrypt* generates $M \in_R G_2$, so the DO will use this M as a secret key and perform the symmetric encryption to encrypt the file that needs to be shared with the DC. Then, the DO uses W_{DO} as a private key to perform symmetric encryption, encrypt the ciphertext M (CT_M) and send the encrypted to file, CT_M,ID_{DO},U,V and keyword then pass it to the CC.

  The CC receives the encrypted file, CT_M,ID_{DO},U,V, and keyword. Since the CC knows the W_{DO} , the CC can use it as a secret key to perform symmetric decryption to decrypt CT_M and store the encrypted file, CT_M, ID_{DO}, U,V, and keyword.

  Finally, the DO will send the keyword to the users under list U and V as a notification to inform them that there is file to be shared with them.

- **File accessing and decryption**

  After the DC logs in successfully, they need to provide a keyword to the CC for requesting the encrypted file and the related information. In this time, the CC will search the related keyword to make sure it exists. If such a keyword exists in the system, then the CC will determine whether the DC is authorised to receive the file by searching the list U and V. If access is permitted, then the CC will send CT_M and the encrypted file to the DC.

  After the DO successfully receives the details from the CC, the DC needs to request the private key to decrypt the CT_M in order to get the proper M' to perform the symmetric decryption on the encrypted file. So, he needs to provide the keyword as an index to request the key from the PKG. Once the PKG receives a request from the DC, the PKG must determine whether the specified keyword already exists in the system. Then, the PKG will determine whether the DC is authorised to receive the key and to check if the DC is in the

http://doi.org/10.18080/jtde.v11n4.772

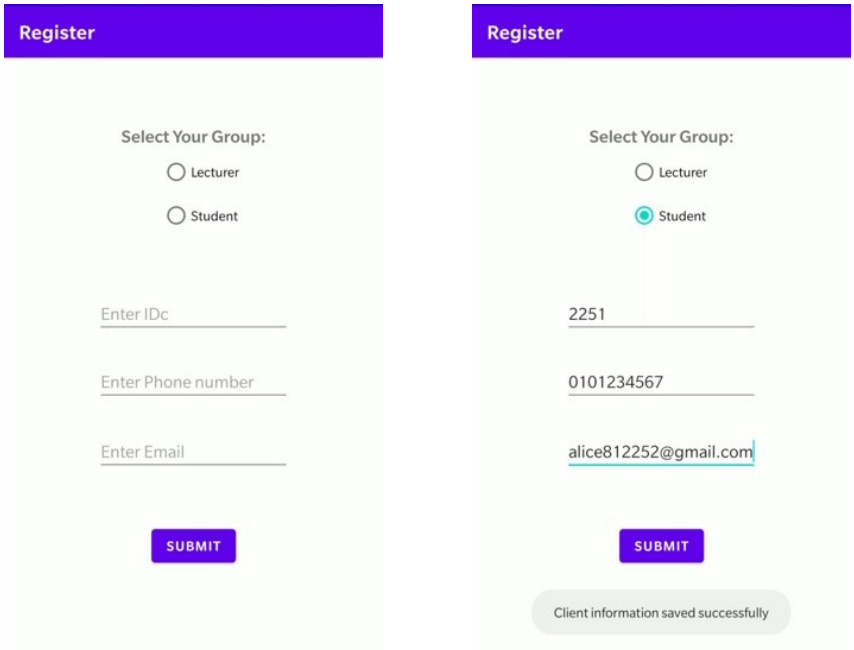
list U or V for generating the proper private key. If access is permitted and the user is in the list U, then the PKG will execute *IBE.Extract* algorithm by using the MN_{DC} and UN_{DC} to generate the private key $K_i^{(1)}$ and send it to the DC. Otherwise, if the user is in the list V then the PKG will generate the private key $K_i^{(2)}$.

Finally, the DC will execute *IBE.Decrypt* by using the proper inputs and the DC will receive the M′. After that, the DC will use the M′ as a private key to execute symmetric decryption to decrypt the encrypted file that he requested from the CC. If M′ and M are the same value, it means that the DO and the DC are using the same key to perform the symmetric encryption and symmetric decryption. So, the DC will receive the correct original file by using the correct M′to decrypt the encrypted file.

User Interface and Functionality

Register

Figure 3(a) shows the register interface for a user to register the account. The user needs to select the group and fill in the ID, phone number and email, then click the submit button to submit the information. The system will check whether the submitted information has been registered with the system. If not, the user will receive notification that client information is saved successfully (see Figure 3(b)) and get the email notification for downloading the secure data sharing application and the related details for login.



(a) (b)

Figure 3. (a) User registration interface; (b) Successfully registered

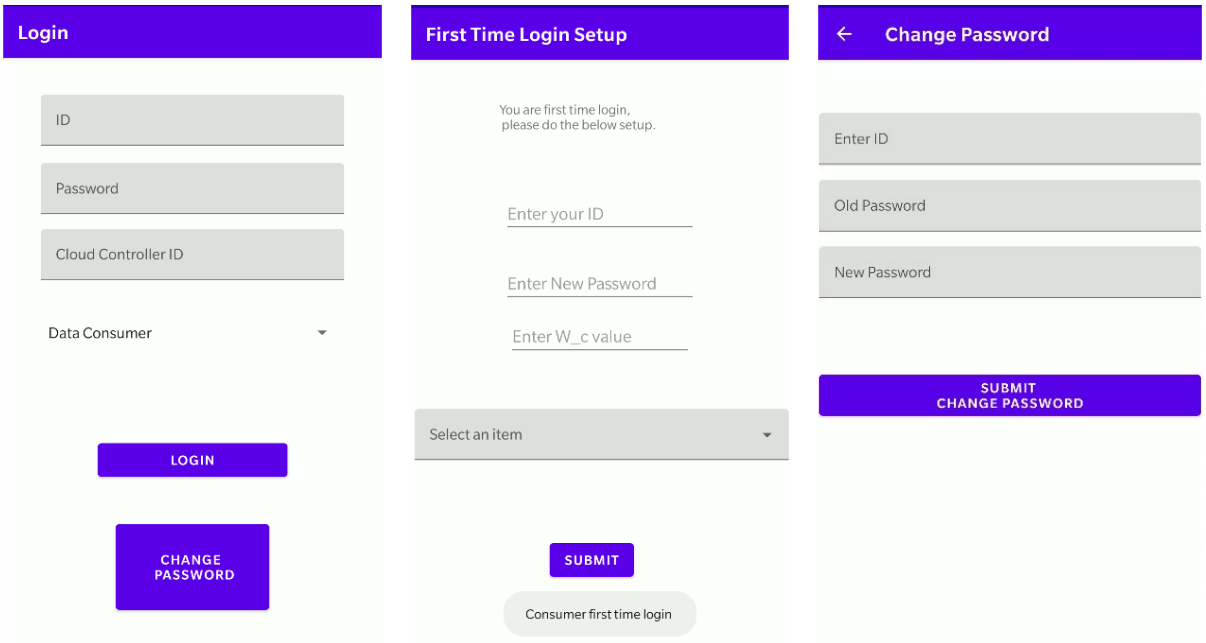
http://doi.org/10.18080/jtde.v11n4.772

Login

After the user downloads the application (see Figure 4(a)), the system will allow user input ID, password and cloud controller ID received from the previous email. If the user is first time login, then the system will navigate the user to first time login setup (see Figure 4(b)). The user needs to fill in the ID, new password and W_c value, and then click the submit button. After completing the first-time setup, the user needs to fill in the required information to log in.

After submitting, the system will execute mutual authentication to check the user and CC. If successful, the user will navigate to the home page and receive notification of authentication success.

Furthermore, the user clicks the change password button and the system will navigate to the change password page (see Figure 4(c)). The user is required to enter the ID, old password and new password to change the password. After completing the required information, the system will run a password renewal algorithm to change the user's password.



(a) (b) (c)

Figure 4. (a) Login interface; (b) First time login interface; (c) Reset password interface

Figure 5(a) shows the home page interface comprising five buttons: sharing/encrypt file, decrypt file, request file, request key and data analysis. Figure 5(b) is the interface of a sharing/encrypt file where the DO needs to enter a keyword as an index of an encrypted file and selects a user and a group list for them to access the encrypted file. After the selection, the user needs to click the encrypt file button, and the system will allow the DO to select which file the user intends to encrypt. Next, the system will run the *IBE.Setup and IBE.Encrypt* to generate the parameter M as a private key and use M to execute the Advanced Encryption

http://doi.org/10.18080/jtde.v11n4.772

Standard (AES) to encrypt the file. After that, the system will request the DO to save the encrypted file on his phone (see Figure 5(c)) and the encrypted file and CT_M will be sent to the CC. The CC will upload them to the cloud. Afterwards, the system will send an email notification to the user and the group selected by the DO.

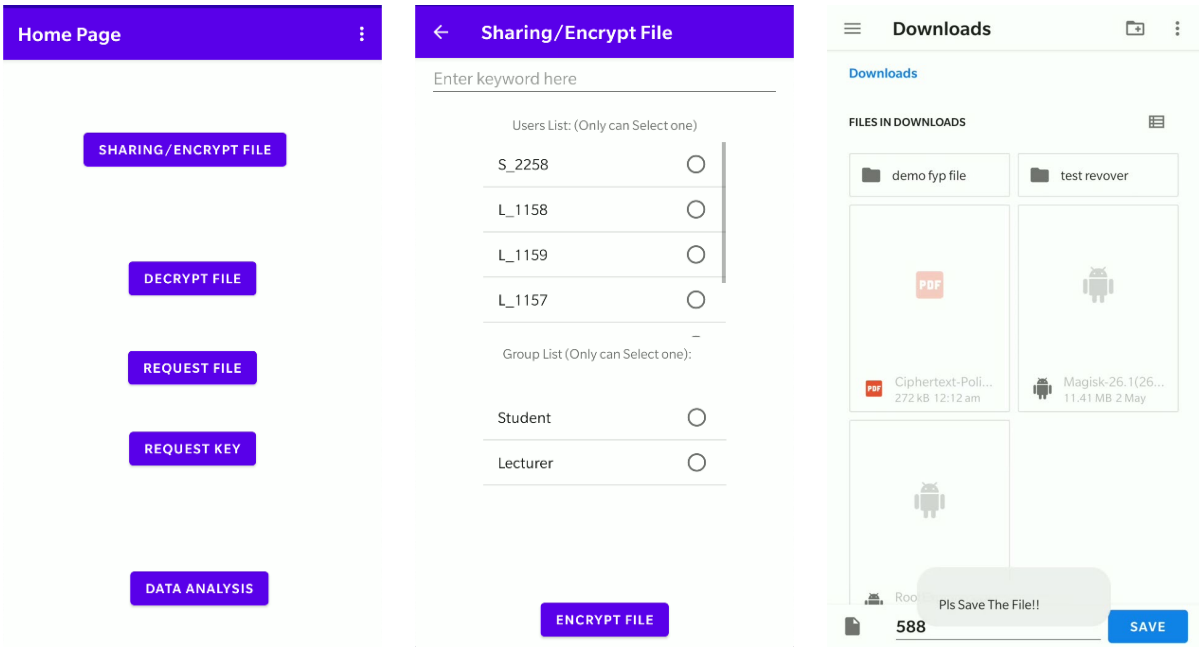


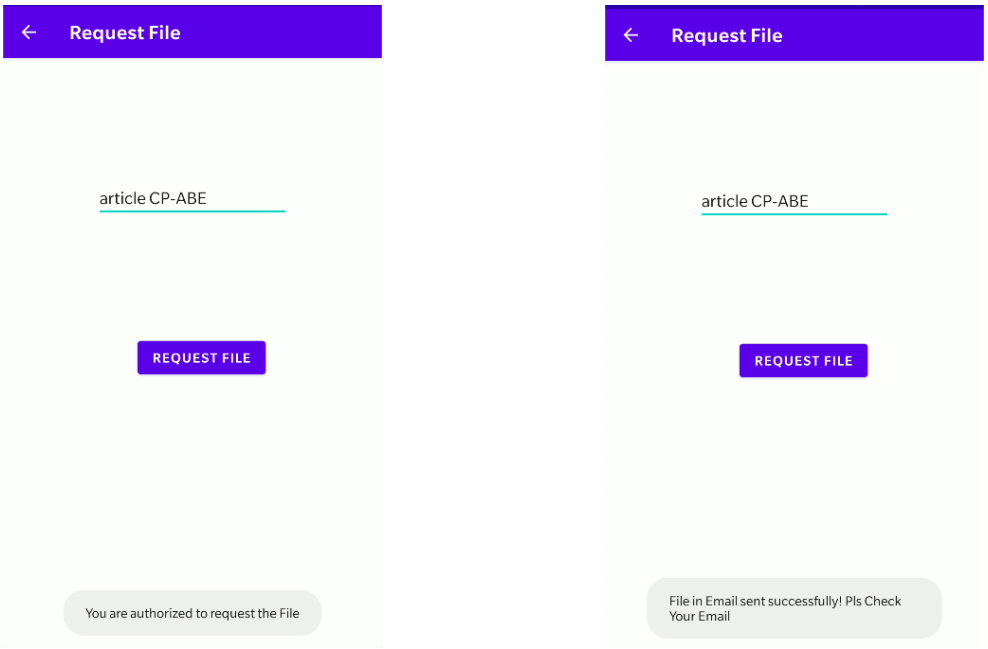
Figure 5. (a) Home page interface; (b) Sharing/encrypt file interface; (c) Save the encrypted file

When data consumers (DCs) request the file, they must click the *request file* button from the home page and fill in the keyword. After that, the CC will check if the DCs have been granted permission and send them the requested file. Figure 6(a) shows the user has permission to request the file and the requested file will be sent via email as shown in Figure 6(b).

To request the key, the DCs must click the *request key* button on the home page and enter the keyword. After that, the PKG will check if the DCs have been granted permission, the PKG will run *IBE.Extract* and send them the requested private key. Figure 7(a) shows the user has permission to request the private key and the requested private key will be sent via email as shown in Figure 7(b).

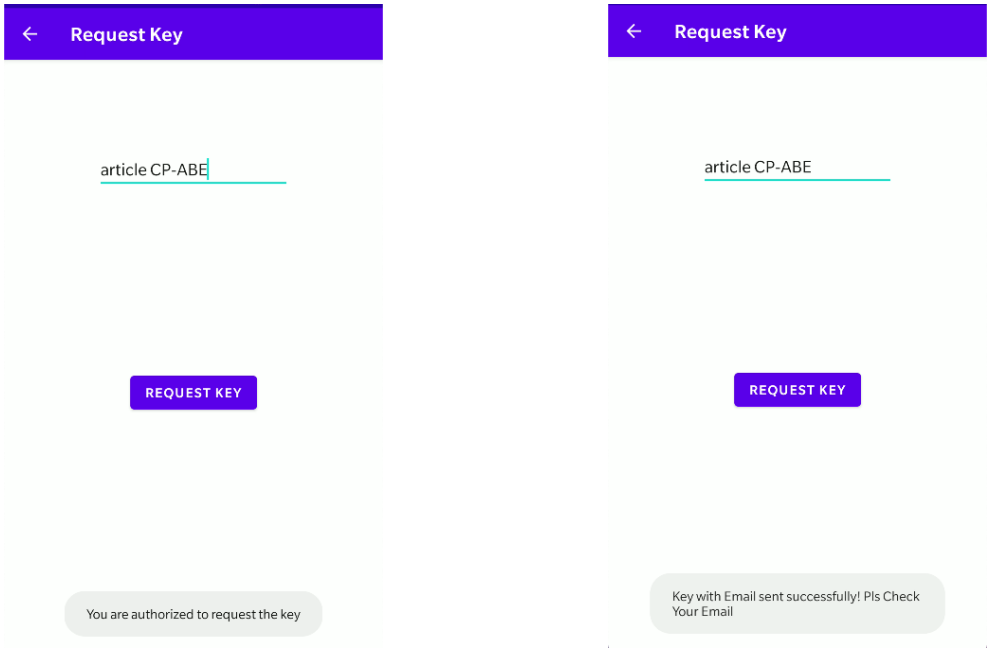
After the DCs successfully requested the encrypted file and private key, they need to click the button *decrypt file* in home page. Figure 8 shows the interface of a decrypt file. The DC needs to enter private key, keyword and DO ID. The DO ID is for the CC to decrypt the encrypted CT_M to be sent to the device of the DC, then the system will use this CT_M to run the *IBE.Decrypt* and generate the final value which is M'. After that, this M' will be used as a secret key and the system will perform AES to decrypt the file. If the DC successfully decrypts the file by providing the true information, the system will require the DC to save the decrypted file on his device.

http://doi.org/10.18080/jtde.v11n4.772



(a) (b)

Figure 6. (a) Result of check DC permission request file; (b): Email notification file sent successfully



(a) (b)

Figure 7. (a) Result of check DC permission request private key; (b): Email notification private key successful

http://doi.org/10.18080/jtde.v11n4.772

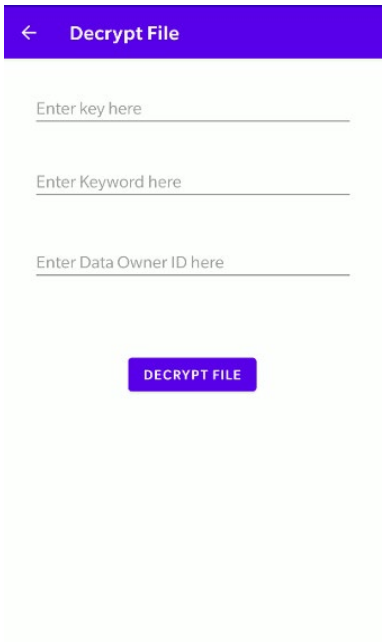


Figure 8. Decrypt file interface

Performance Analysis

IBADS is a protocol which employs IBE and AES to perform and achieve secure data sharing. Following is a discussion of the performance analysis of the IBADS protocol. The first analysis includes the execution time of the IBE scheme and include *IBE.Setup*, *IBE.Extract*, *IBE.Encrypt*, *IBE.Decrypt* in a java environment and mobile application environment. The second analysis includes the time required for encrypting and decrypting file in a mobile application environment.

The analysis is performed using MacBook Air with M2 chip 3.49GHz with 16GB RAM running on macOS Monterey for IBE scheme in java environment. The analysis is performed using Xiaomi 12T with Snapdragon 8+ Gen1 Mobile Platform Octa-core Max 3.2GHz with 11GB RAM for IBE scheme and AES encryption in Android application environment.

Performance analysis for IBE scheme

It is important to note that Karati *et al.* (2018) adopted the Pairing-Based Cryptography Library to run the bilinear pairing cryptographic operations, while this project used the Java Pairing-Based Cryptography Library in the implementation. Thus, the execution time cannot be compared directly since different cryptographic libraries are used to implement the IBADS protocol, respectively. In order to achieve faster pairing computation, Type-A curve of group size 512-bit is chosen to compute the pairing.

The performance analysis in Figure 9 shows that the IBE scheme running on the mobile application obtains good performance based on its running time. There is little difference

http://doi.org/10.18080/jtde.v11n4.772

when compared to its execution on a computer. We also take into consideration that the technical specifications of the device will also affect the running time.

Figure 9 depicts the performance of IBE algorithms, namely *IBE.Setup, IBE.Extract, IBE.Encrypt* and *IBE.Decrypt*. The X-axis represents the respective algorithms while the Y-axis represents the time taken in milliseconds. The two-colour legend represents the Java platform and Android application platform, respectively.

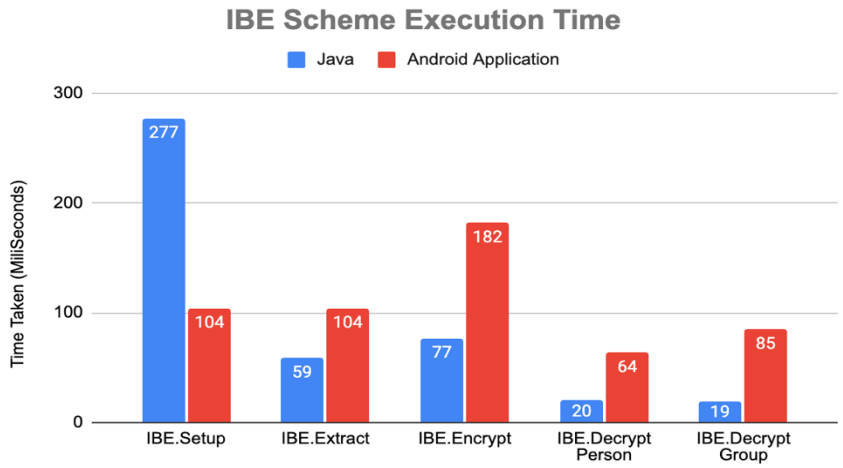


Figure 9. Performance of IBE scheme

Performance analysis for secure data sharing

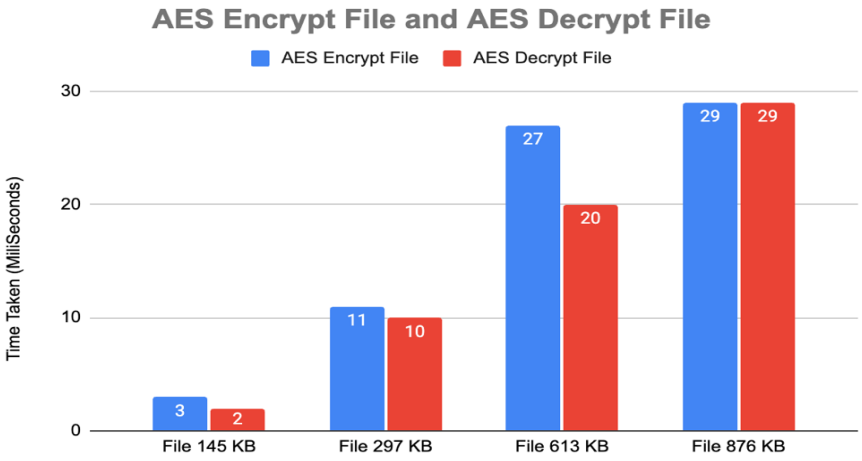


Figure 10. Performance of AES encryption

Similarly, Figure 10 shows that the AES encryption experiences good performance in the process of file encryption and decryption. This results in fast AES encryption runtime and less consumption of time in performing encryption and decryption in the mobile application. From this analysis, it is obvious that the IBADS protocol aligns with the characteristics of lightweight efficiency. This means the IBADS protocol can carry out encryption tasks on a mobile device swiftly and smoothly, assuring efficient running time and speed.

http://doi.org/10.18080/jtde.v11n4.772

This analysis records the time taken using a 256-bit key to perform AES file encryption and decryption in the mobile application. In this analysis, the files used to run the AES encryption are 145 kilobytes (KB), 297 KB, 613 KB and 876 KB, respectively.

Figure 10 presents the performance via Android application. The X-axis represents the file sizes to be encrypted and decrypted while the Y-axis represents the time taken in milliseconds. The two-colour legend represents the AES encryption time and AES decryption, respectively.

Conclusion and Potential Future Work

With the increasing number of users, the issue of secure data sharing in an untrusted cloud has become a major concern. It can lead to the privacy of users being unprotected, and the data of users being potentially stolen in a mobile cloud environment. Hence, this proposed application has implemented a secure data sharing protocol based on the IBADS protocol to ensure the data sharing process is secure and protected. The IBADS protocol performs lightweight computation operations and uses IBE as the main algorithms to perform the data sharing protocol. IBE is a user-friendly encryption based on using simple user identity as the public key to perform the encryption. It does not require the user to enter or obtain any public key prior to encryption.

Future work would consider improving the algorithms of *IBE.Encrypt* and *IBE.Decrypt* such that they can be designed to perform one-to-many operations, meaning that the DO can encrypt a file and share it to multiple users and multiple groups at once. Considering the lightweight running requirement in Android applications, this adjustment needs to be improved carefully as the one-to-many operations will increase the running times accordingly. Also, the IBE scheme employed in the IBADS protocol is using Type 1 symmetric pairing $G_1 \times G_1 \rightarrow G_T$ to execute the pairing-based cryptography. This type of pairing is simpler to implement, but has a drawback in that it requires larger parameters for a given level of security, which can then lead to inefficiencies. For better performance, Type 3 asymmetric pairing $G_1 \times G_2 \rightarrow G_T$ should be used for better performance and security as it can provide smaller parameters for the same level of security.

Acknowledgements

A version of this paper was presented at the Third International Conference on Computer, Information Technology and Intelligent Computing, (CITIC 2023), held in Malaysia on 26–28 July 2023.

This work was supported by the Telekom Malaysia Research and Development Grant (RDTC/221045).

http://doi.org/10.18080/jtde.v11n4.772

References

Cheng, B., Zhang, J., Hancke, G. P., Karnouskos, S., & Colombo, A. W. (2018). Industrial cyberphysical systems: Realizing cloud-based big data Infrastructures. *IEEE Industrial Electronics Magazine*, *12*(1), 25–35. https://doi.org/10.1109/MIE.2017.2788850

De Caro, A., & Iovino, V. (2011). Introduction. JPBC – Java Pairing-Based Cryptography Library. Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Corfu, Greece. http://gas.dia.unisa.it/projects/jpbc/

Deng, H., Wu, Q., Qin, B., Domingo-Ferrer, J., Zhang, L., Liu, J., & Shi, W. (2014). Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts. *Information Sciences*, *275*, 370–384. https://doi.org/10.1016/j.ins.2014.01.035

Han, S., Han, K., & Zhang, S. (2019). A data sharing protocol to minimize security and privacy risks of cloud storage in big data era. *IEEE Access*, *7*, 60290–60298. https://doi.org/10.1109/ACCESS.2019.2914862

Java Platform, Standard Edition Security Developer's Guide. (n.d.). Oracle. https://docs.oracle.com/javase/9/security/java-security-overview1.htm#JSSEC-GUID-2EF91196-D468-4D0F-8FDC-DA2BEA165D10

Karati, A., Amin, R., Islam, S. H., & Choo, K.-K. R. (2018). Provably secure and lightweight identity-based authenticated data sharing protocol for cyber-physical cloud environment. *IEEE Transactions on Cloud Computing*, *9*(1), 318–330. https://doi.org/10.1109/TCC.2018.2834405

Li, H., Huang, Q., & Susilo, W. (2022). A secure cloud data sharing protocol for enterprise supporting hierarchical keyword search. *IEEE Transactions on Dependable and Secure Computing*, *19*(3), 1532–1543. https://doi.org/10.1109/TDSC.2020.3027611

Lu, X., Pan, Z., & Xian, H. (2020). An efficient and secure data sharing scheme for mobile devices in cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, *9*, 60. https://doi.org/10.1186/s13677-020-00207-5

Salvi, A. H. (2019). Information and network security with cryptography. *International Journal of Research in Electronics and Computer Engineering – a Unit of I2OR*, *7*(3), 879–883. http://nebula.wsimg.com/8340cc87de92fbce086c7959a2ab45e1?AccessKeyId=DFB1BA3CED7E7997D5B1&disposition=0&alloworigin=1

Shao, J., Lu, R., & Lin, X. (2015). Fine-grained data sharing in cloud computing for mobile devices. *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2677–2685. https://doi.org/10.1109/INFOCOM.2015.7218659