IBIMA Publishing
Journal of Electronic Banking Systems
http://www.ibimapublishing.com/journals/JEBS/jebs.html
Vol. 2015 (2015), Article ID 614386, 13 pages
DOI: 10.5171/2015.614386

Research Article

A Framework for Secure Online Bank System Based on Hybrid Cloud Architecture

Khaled Ahmed Nagaty

Faculty of Informatics & Computer Science
The British University in Egypt, Egypt

Correspondence should be addressed to: Khaled Ahmed Nagaty; khaled.nagaty@bue.edu.eg

Received date: 11 March 2014; Accepted date: 20 October 2014; Published date: 2 September 2015

Academic Editor: Jaishree Asarpota

Abstract

In this paper, we present a hybrid cloud architecture which combines public cloud, private cloud computing and cryptography to minimize the bank's operational costs, maximizing the flexibility, scalability, availability and reliability of the services provided by the bank and guarantees the privacy, confidentiality and safety of the client's record. A smart card which contains a secret key is produced for each client upon creating a new account. A smart card which contains a pair of public/private keys is produced for each bank. A smart card which contains a secret key is produced for the auditor to decrypt the database of the banks' public/private keys for auditing purposes. The secret key is used to encrypt and decrypt the client's account data The bank uses its public key to double encrypt the client's data that are temporarily stored in the bank's private cloud before being transmitted to be stored permanently in the public cloud. In order to perform any transaction on a client's account, the client's record must be retrieved from the public cloud and stored temporarily in the bank's private cloud in order to be decrypted with the bank's private key and then decrypted using the client's secret key in order to perform the required transaction in the private cloud.

Keywords: Online banking, Hybrid cloud, Cryptography, Security

Introduction

Online devices such as tablets or smart phones are now launching a new era of online services as they are giving themselves to cloud computing. Using online devices, information can be transmitted continuously between clients and their banks, where transactions can be performed anytime and anywhere. Online banking can benefit from the abundant resources available in the public cloud which make bank services more efficient and economic. This is because cloud providers adopt the pay-as-you-go service models which mean that a bank will pay only for what is used, instead of making a huge initial investment in IT infrastructure. However, many arguments regarding privacy

Cite this Article as: Khaled Ahmed Nagaty (2015), "A Framework for Secure Online Bank System Based on Hybrid Cloud Architecture ", Journal of Electronic Banking Systems, Vol. 2015 (2015), Article ID 614386, DOI: 10.5171/2015.614386

and security issues in cloud computing are raised by many organizations which depend mainly on cloud computing. By integrating cloud computing with cryptography, the privacy, confidentiality and protection of data in the public cloud storage will be guaranteed, this will encourage banks and financial organizations to adopt the cloud computing model. In this article, we present secure hybrid cloud architecture for online banking which uses cryptographic techniques to protect privacy, confidentiality and security of clients' data. Up to our knowledge there is no previous work in the field of cloud banking which combines hybrid cloud with cryptography. The most related work to our model is implemented in electronic health systems based on hybrid clouds in Yu-Yi et al (2012). This paper is organized as follows: section 2 is dedicated to cryptographic key generation, section 3 explains the role based access control method; section 4 is dedicated to explaining the proposed online banking system that is based on hybrid cloud integrated with cryptographic techniques; section 5 is dedicated to the analysis of the proposed system; while section 6 is dedicated to conclusions.

Cryptographic Keys Generation

Bank's Public/Private Key Pair Generation

To generate a key pair of public/private keys for a bank, we depend on identity based cryptography (IBE) where a string representing the bank identity *ID* can be used to generate a public key that can be used for encrypting and decrypting information. Using the bank's identity in identity based cryptography prevents keys collision. A trusted third party, called the private key generator (PKG), generates the corresponding private keys. It publishes a master public key and retains the corresponding master private key. By using the master public key any bank can generate a public key that corresponds to the bank identity *ID* by combining the master public key with the identity value. To generate the corresponding private key, the user was authorized to use the bank identity *ID* contacts the PKG, which uses the master private key to generate the private key for bank identity *ID*. Dan et al (2003) defined a set of four algorithms that form a complete IBE cryptosystem:

Setup: This algorithm runs by the PKG system one time for creating the whole IBE environment. The master private key is kept secret and used to derive banks' private keys, while the system parameters are made public. It accepts k a security parameter (i.e. binary length of key material) and outputs:

- A set P of system parameters.
- A master private key K_m.

Extract: This algorithm runs by the PKG when a bank requests its private key. It takes as input P, K_m and bank identifier $ID \in \{0,1\}^*$ and returns the private key d for the bank ID.

Encrypt: This algorithm takes P , message $m \in M$, $ID \in \{0,1\}^*$ and outputs the encryption $c \in C$.

Decrypt: This algorithm accepts d,P, $c \in C$ and returns $m \in M$.

Example

Assume that Bank "A" wants to store a client's information in the cloud storage:

1) Bank "A" encrypts and signs its identity *ID* and then sends it with a message to PKG in order to retrieve the public key.
2) The PKG uses the master public key and bank's "A" identity *ID* to generate bank's "A" public key.
3) The PKG then sends to bank "A" its public key.
4) Bank "A" encrypts the client's information and stores it in the cloud.

Fig.1 shows the generation of a public key using IBE and sends encrypted information to the cloud

computing storage.

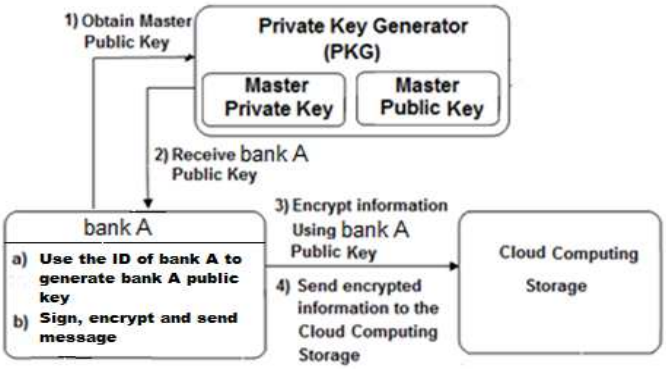

Figure 1: Generates a public key and sends encrypted information to the cloud storage using IBE

Now assume that bank "A" wants to retrieve encrypted information from the cloud storage:

1) Bank "A" encrypts and signs its identity *ID* and then sends it with a message to PKG in order to retrieve its private key.
2) The PKG uses the master private key and bank's "A" identity *ID* to generate bank's A private key.
3) The PKG then sends to bank "A" its private key.
4) Bank "A" decrypts the retrieved secret information using its private key.

Fig.2 shows the generation of private key using IBE and decrypts the received encrypted information from the cloud computing storage.

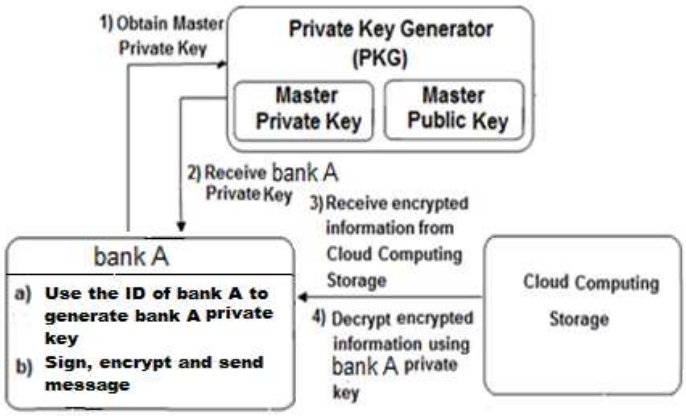

Figure 2 : Generates private key and decrypts the received encrypted information from the cloud storage using IBE

Client's Secret Key Generation

To generate a client's secret key we modified the algorithm of Dan et al. (2003) as follows: an identity string represents the concatenation of client's identity *ID,* client's account number *AC*, client's first name *FN* and the client's last name *LN* can be used to generate a client secret key which can be used for encrypting and decrypting client's record. Using the above mentioned identity string in identity based cryptography prevents secret keys collision. A trusted third party, called the secret key generator (SKG) retains a master secret key (MSK) which when combined with the identity string (*ID*, *AC*, *FN*, *LN*) of a client that client can generate a secret key which corresponds to the client's identity string.

Setup: This algorithm runs by the SKG system one time for creating the whole IBE environment. The master secret key is kept secret and used to derive users' secret keys, while the system parameters are made public. It accepts k a security parameter (i.e. binary length of key material) and outputs:

- A set P of system parameters.
- A master key MSK.

Extract: This algorithm runs by the SKG when a client requests his secret key. It takes as input P, MSK and identifier $(ID, AC, FN, LN) \in \{0,1\}^*$ and returns the secret key SK for user (*ID*, *AC*, *FN*, *LN*) as follows:

$$SK = H(ID, AC, FN, LN)^{MSK} \quad (1)$$

Where H is a cryptographic hash function such as MD5 or SHA-1

Encrypt: This algorithm takes P, a message $m \in M$ and $(ID, AC, FN, LN) \in \{0,1\}^*$ then outputs the encryption $c \in C$.

Decrypt: This algorithm accepts SK, P and $c \in C$ then returns $m \in M$.

Fig.3 shows secret key generation for client Alice and sends encrypted information to the cloud computing storage using IBE.

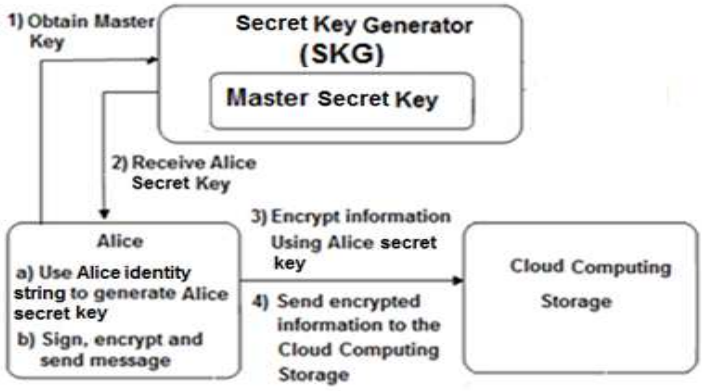

Figure3 : Generates secret key for client Alice and sends encrypted information to the cloud storage using IBE

Fig.4 shows secret key generation for client Alice and decrypts the received encrypted information from the cloud storage using IBE

.

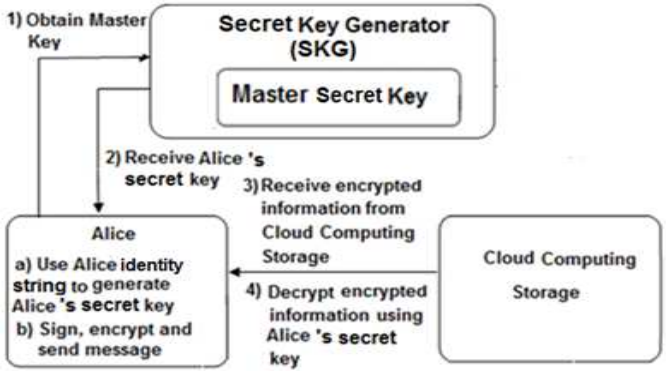

Figure 4: Generates secret key for client Alice and decrypts the received information from the cloud storage using IBE

Role Based Access Control (RBAC) Method

Role Based Access Control (RBAC) is a method that offers a satisfactory level of safety & security for organizational resources & data because of rules & policies put into effect for the user in the form of login & password. In all interpretations of RBAC, the notion of a role is introduced in between principals (e.g., users and processes) and privileges (e.g., method calls and files system access requests). Used in this way, roles are essentially a form of grouping. However, the description is not limited to the organization resources but gives security and protection for users' personal information and actions. Role Based Access Control (RBAC) mentioned in Shefali (2012) offers authentication, authorization and auditing for users using the cloud computing as follows:

Authentication: Cloud computing authentication includes validating the identity of users or systems. For example, facility to service authentication engages in certifying the access demand to the information which served by another service.

Authorization: After the authentication process, the system will put security rules to bring legitimate users.

Auditing: Auditing is a process that involves reviewing & examining the records of authorization & authentication to check over organizations compliance with set security standards & policies in order to evade system breaches

RBAC is very useful to verify the identity of a system user whether he is a client or an authorized bank employee.

Proposed Secure Hybrid Cloud Online Bank System

In our model, the cloud bank system will be constructed by the banks and cloud providers as shown in Fig. 5. Using hybrid cloud in the proposed system will be more secure than using the public cloud only because the encryption, decryption processes and transactions of the clients' data are performed in the bank's private cloud not in the public cloud. Double encrypted clients' records stored in the public cloud get use of the huge storage capacity of the public cloud.

Double encrypted client's records retrieved from the public cloud storage are firstly decrypted in the bank's private cloud using the bank's private key then they are decrypted with the client's secret key where the required transactions are performed. For the purpose of data confidentiality, privacy and safety, each bank record of a client *Client_record* is encrypted with a unique client's secret key $client_{key}$ using symmetric encryption scheme. The encrypted client record contains personal information, transaction records, balance, loans information and account statements. When creating a new bank record for a client, he must select a symmetric key for the purposes of encryption and decryption of his bank record. The newly created bank record is then encrypted using the selected client's secret key to become $Client\_record_{encrypted}$. By using asymmetric cryptography, each bank should create a public key $bank_{public}$ and a private key $bank_{private}$. In order to store the encrypted client record in the public cloud, it must be additionally encrypted with the bank's public key. To access a double encrypted client record from the public cloud, it must be first decrypted by the bank's private key and then decrypted by the client's secret key. In case of cross banks accessing, when a request bank wants to perform a required transaction on a client's record which belongs to another bank, we call this the owner bank, it sends a secret message which contains the account number of the client and the required transaction to be performed on the client's account. The request bank encrypts this message with the owner's bank public key $owner\_bank_{public}$ and sends it to the owner bank. When the encrypted message is received by the owner bank, it decrypts it with its private key $owner\_bank_{private}$ and then performs the required transaction. In this scheme, each client has only one symmetric key for the encryption and decryption of his bank record while each bank has two asymmetric keys a public key $bank_{public\_key}$ and a private key $bank_{private\_key}$ for the double encryption and decryption of the clients records in order to store them in the public cloud and encrypting banks' signatures and messages in cross banks transactions. A third party such as the central bank acts as an auditor in the public cloud. Audit means recording the bank activities of the bank system in chronological order, such as maintaining a log of every access to and modification of data. The auditor has a database of all banks' private/public keys. This database is encrypted/decrypted using the auditor's secret key.

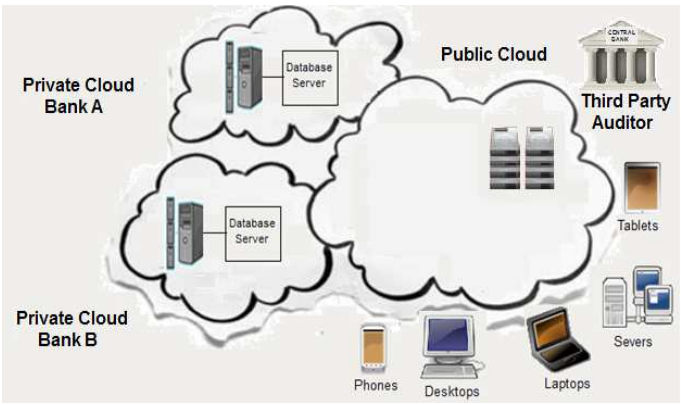

Figure 5: Online Bank Hybrid Cloud Architecture

Electronic Bank Record (EBR) Creation

The client's bank record *Client_record* is created by an authorized bank employee as shown in Fig.6. Each client during the creation of his/her account must select a symmetric key $client_{key}$ for encrypting and decrypting his bank record. This key is called the client's secret key. A smart card which contains the client's encryption/decryption secret key is produced and delivered to the client. This smart card is known as the secret key smart card. The authorized employee uses the client's secret key $client_{key}$ to encrypt the client's bank record. This secret key is added to the database of clients' secret keys in the bank. Moreover, this database is encrypted by the bank's public key $bank_{public\_key}$.

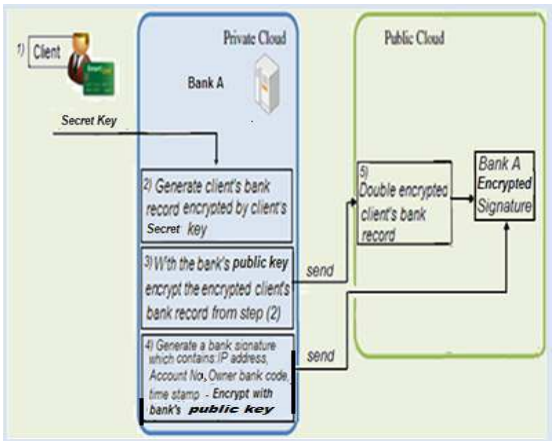

Figure 6: Client's EBR Creation (as adapted from Yu-Yi C. et al., 2012)

The electronic bank record (EBR) *Client_record* of each client contains the client's personal information, transaction records, official signature, balance….etc. This record is encrypted with the client's secret key $client_{key}$ as follows:

$$Client\_record_{encrypted} = client_{key}(Personal\_Info., Client\_Signature, Balance, Transations)$$

$$= (Personal\_Info, Client\_Signature, Balance, Transactions)^{client_{key}} \bmod N \quad \textbf{(2)}$$

The bank keeps a database with all the clients' names, account numbers and secret keys. This table is encrypted with the bank's public key as follows:

$$secret\_keys\_table_{encrypted} = bank_{public\_key}(Client\ Name, Client\_Signature, client_{key})$$

$$= (Client\_Name, Client\_Signature client_{key})^{bank_{public\_key}} \bmod N \quad \textbf{(3)}$$

Only authorized employees who are capable to modify clients' accounts can acquire the bank's private key in order to decrypt the table of clients' secret keys as follows:

$$secret\_keys\_table = bank_{private\_key}(secret\_keys\_table_{encrypted}) \quad \textbf{(4)}$$

To store a client's bank record in the public cloud, first the client's record should be encrypted with the client's secret key and the output is additionally encrypted with the bank's public key as follows:

$$Client\_record_{double\_encrypted}$$

$$=bank_{public\_key}(client_{key}(Personal\_Info,Client\_Signature,Balance,Transations)) \quad (5)$$

The double encrypted client's bank record $Client\_record_{double\_encrypted}$ is then securely transmitted to be stored in the public cloud storage. A bank signature which contains the *IP* address of the machine generated the signature, the client's account number and the code of the bank is generated. The created signature is encrypted with the bank's public key and then transmitted to the public cloud to be stored in a linked list associated with the double encrypted client's bank record:

$$Bank\_signature_{encrypted}$$

$$=bank_{public\_key}((IP\_address,Account\_No,Bank\_Code,Time)) \quad (6)$$

Note that the bank signature is encrypted with bank's public key so that an auditor can obtain the bank's private key from the database of all banks public/private keys to decrypt this signature as follows:

$$Bank\_signature = bank_{private\_key}(Bank\_signature_{encrypted}) \quad (7)$$

Fig. 7 shows a double encrypted client record that associated with it a linked list of encrypted bank signatures that were arrived at different time stamps. These bank signatures are ordered chronologically, which means that the newest arrived bank signature is the last one added to the linked list of bank signatures. These bank signatures are used by the auditor after being decrypted with the bank's private key which created these signatures. The auditor gets the bank's private key from the database of all banks' public/private keys after being decrypted by the auditor's secret key.

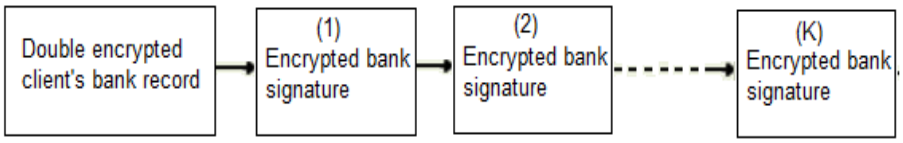

Figure 7: Double encrypted client record with linked list of bank signatures

This double encryption scheme of clients' records is used to guarantee client's data privacy, confidentiality and integrity by avoiding any unauthorized access to clients' accounts.

Electronic Bank Record Access

The situation is divided into two types A-type and B-type. In A-type, as shown in Fig.8 when a client wants to access his bank record online, he must firstly use his credit or debit card and submits his personal identification number (PIN) to the private cloud server in order to authenticate himself using RBAC. The private cloud then validates the role assigned to the user. If the input PIN matches that stored in the database, then the client's account number is sent to the public cloud to retrieve the double encrypted bank record from the bank's database of accounts. When the double encrypted bank record is retrieved, it is sent securely to the bank's private cloud. In the bank's private cloud, the bank's private key is used to decrypt the double encrypted client record, then the client uses his secret key to decrypt his encrypted bank record and starts the desired transactions as follows:

$$Client\_record = client_{key}(bank_{private\_key}(Client\_record_{double\_encrypted})) \quad (8)$$

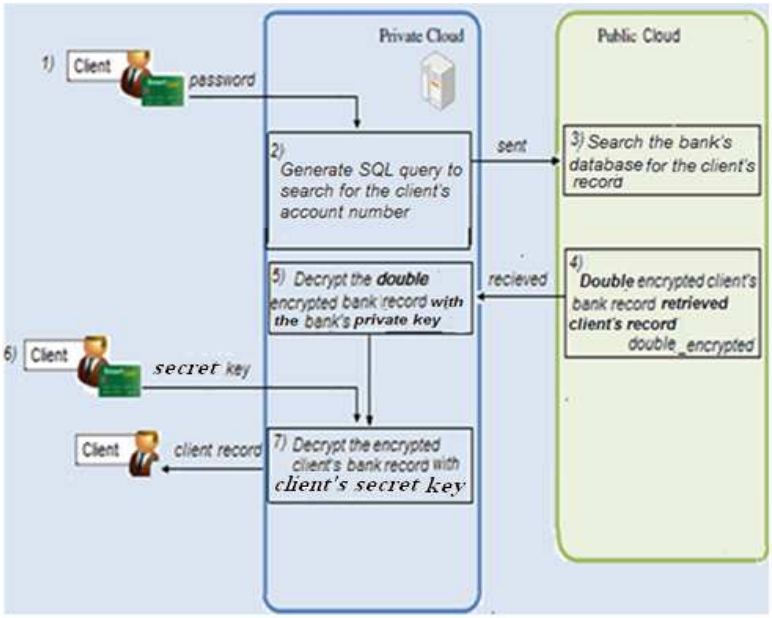

Figure 8: Client EBR Access (as adapted from Yu-Yi C. et al., 2012)

After finishing his transactions, the client uses his secret key again to encrypt his modified bank record which is then additionally encrypted with the bank's public key and securely sent to the public cloud to be stored as follows:

$$Client\_record_{double\_encrypted} = bank_{public\_key}(client_{key}(Client\_record)) \quad (9)$$

In the mean time, a bank signature is generated and encrypted with the bank's public key and added to the list of bank signatures associated with this double encrypted client's record and stored in the public cloud too.

Fig.9 shows a recently encrypted bank signature added to the list of encrypted bank signatures that is associated with the double encrypted client's bank record after a transaction is completed.

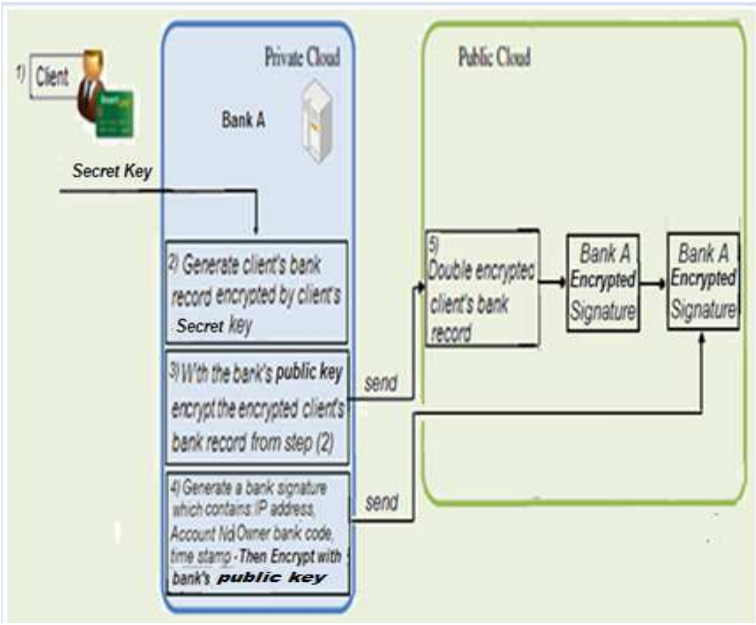

Figure 9: Client EBR Modification (as adapted from Yu-Yi C. et al., 2012)

In the B-type, the client's bank record is accessed by an authorized employee at the same bank as shown in Fig. 10. Before accessing the client's double encrypted bank record in the public cloud, the bank's private cloud validates the identity of the authorized employee and the roles assigned to him using the RBAC method. After validating the employee's role, the authorized employee should acquire the client's secret key from the database of clients' secret keys in the bank's private cloud. Firstly, the employee uses the bank's private key to decrypt the database of clients' secret keys, which guarantees that only authorized employees can access clients' secret keys as follows:

$$keys\_table = bank_{private\_key}(secret\_keys\_table_{encrypted}) \quad (10)$$

A SQL query containing the client's account number and the bank's code are both sent to the public cloud server to search for the client's double encrypted record in the database of the bank accounts. When the double encrypted client record is retrieved, it is first decrypted by the bank's private key and then by the client's secret key in order to perform the required transactions. After finishing the required transaction, the bank encrypts the client record with the client's secret key and then by the bank's public key and is sent securely to be stored in the public cloud as in Fig. 10. A bank signature is also generated and encrypted with the bank's public key and sent to the public cloud to be added to the linked list of encrypted bank signatures associated with the double encrypted client's record. These encrypted bank signatures are used for auditing purposes. Finally, the client is notified by his bank that his secret key is used by an authorized employee in order to access his bank record to perform a required transaction.

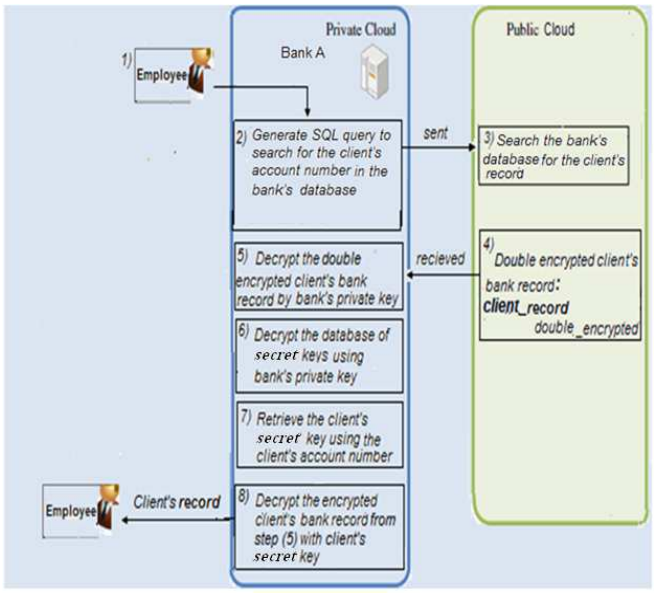

Figure10: Bank Employee EBR Access (as adapted from Yu-Yi C. et al., 2012)

In case of an employee from a requesting bank "A" wants to perform a required transaction on a client's record at bank "B", the employee of bank "A" will send the client's account number and the required transactions to bank "B" in a message encrypted with the public key of bank "B" i.e. $bank\_B_{public}$. Then bank "B" will decrypt the received message with its private key i.e. $bank\_B_{private}$ and perform the required transactions on the client's record as shown in Fig. 9 and Fig.10.

The client is then notified by bank "B" that his secret key is used to access his bank record to perform a transaction requested by bank "A".

Analysis

When considering analyzing the performance of the proposed hybrid cloud banking system, we mainly consider two major categories: performance from the point of view of the cloud bank provider and performance from the point of view of client's cloud banking view.

Cloud Bank Performance: The metrics cloud bank service provider is interested in involving the scalability, robustness and availability of the cloud bank services, protecting and monitoring customers' data in the cloud.

Scalability: A scalable system is one that can serve a large number of users and scale up operations while maintaining an acceptable response time, with constant or slowly degrading performance with little overheads. Our online bank system based on hybrid cloud platform achieves scalability through using the abundant resources of the public cloud platform. This means that a hybrid cloud online bank system can handle many clients simultaneously without any failure noticed due to resource exhausting. Given the pay-as-you-go model, the computational cost will grow slowly as the number of clients increase.

Robustness and availability: By using the technology of virtualization, resiliency, redundancy, data restoration and disaster recovery, the cloud environment is highly tolerant against many failure scenarios which makes the proposed cloud online bank

architecture robust against service disruptions due to power outages, denial of service attack, hardware failures, and system upgrades. This guarantees the availability and utility of online bank services at all times.

Security through cryptography: Our proposed architecture uses symmetric and asymmetric cryptography to secure, protect clients' data and to control access to critical operations. Each client has a symmetric key which is also called a secret key for the encryption and decryption of his/her bank record. The bank itself has a public key to encrypt clients' secret keys, double encrypt clients' records and encrypt all messages that contain clients' vital information.

Authentication and authenticity: The authenticity and authentication of a client can be confirmed by using his/her secret key and RBAC method. While the authenticity and authentication of an employee can be confirmed by using the bank's private key. A client should use his account number to retrieve his double encrypted bank record from the public cloud. When the double encrypted bank record is retrieved and stored in the safety of the private cloud, the bank automatically decrypts the double encrypted client's record by using the bank's private key. After this, the client uses his encryption/decryption smart card that contains his secret key to decrypt his bank record, and finally uses his/her password to access his/her bank record in order to perform the desired transaction. If an authorized person in a bank requires accessing a client's bank record in the same bank, he firstly decrypts the database of secret keys for the clients using the bank's private key in order to access the required client's secret key. An SQL query that contains the client's account number is sent to the public cloud server in order to retrieve the double encrypted client's bank record and temporarily store it in the private cloud. In the private cloud, the bank automatically decrypts the client's bank record with the bank's private key, and then the authorized employee can use the client's secret key to decrypt the client's bank record and do whatever transactions are required. After finishing these transactions, the process is reversed, and an encrypted signature is generated and added to the list of bank signatures stored in the public cloud. In the mean time, the client is notified by his/her bank about the committed transactions.

Monitoring: One of the hot issues in cloud computing is monitoring the use of the data by the cloud providers. In our proposed hybrid cloud bank architecture, a third party such as the central bank will act as an auditor. Audit means recording all activities on a client's bank record in chronological order which can be done using bank signatures. The auditor has a database for all banks' public/private keys which is encrypted by the auditor's secret key. Bank signatures are encrypted with the bank's public key which generated the signatures in order to prevent tampering with them and to allow the auditor to decrypt them at any time using the bank's private key. Bank signatures maintain a log for all modifications done to the client's record stamped with the time of modifications. Monitoring the utilization of cloud resources and infrastructure is an important factor in cloud bank monitoring.

Transaction metrics: Success percentage of transactions gives a clear picture of the performance of the application in the cloud at a particular instance.

Cloud Bank Customer's View Performance

Customers of the cloud banking system are interested in the flexibility of the banking services and the performance of the bank applications hosted in the cloud.

Flexibility: Clients using online bank systems based on cloud platform will be flexible to access their bank records and doing transactions on their accounts from any place they are and at any time they like. Banks will be more flexible using IT resources available in the cloud, they can double their clients in half an hour. They can

scale up and down depending on the number of clients using their bank system. They can use extra resources in peak hours in order to sustain their clients and to avoid intolerable time latency or performance degradation. In the mean time, they pay for what they get.

Application Response Time: It actually calculates the time taken for each transaction to complete. It measures the impact factor of the application performance and availability.

Conclusion

We proposed a secure online hybrid cloud banking architecture which allows banks to store and retrieve their clients' accounts using the public cloud in a more secured way. As a potential product of our model, smart cards which contain client's secret key are produced and delivered to the clients upon creating new bank accounts. Smart cards which contain a pair of public and private keys are produced for the banks to double encrypt/decrypt the clients' records. Smart card which contains a secret key is produced for the auditor to encrypt/decrypt the database of all banks public/private keys. In our model, the client's bank data such as transactions, bank statements, loans information, and personal information are stored permanently in the public cloud and stored temporarily in the bank's private cloud for processing. Information stored in the safety of private cloud can only be encrypted with the client's secret key, but clients' data stored in the public cloud should be additionally encrypted with the bank's public key. Within this kind of cloud environment, the cross bank access is an important issue between a request-bank and an owner-bank. We set up a mechanism to make sure that the privacy, confidentiality and integrity of bank records are protected in the scheme of normal and cross bank situations. In the normal situation, if a client requests to access his bank record or an authorized employee requests to access this client's bank record both the client and the employee must use the client's secret key in order to decrypt the client's bank record. In case of cross banks, the request bank encrypts the client's account number and the required transaction with the owner's bank public key, and then sends them to the owner bank. The bank owner will use its private key to decrypt the received message in order to acquire the client's account number and performs the required transaction. In both normal and cross bank situations an encrypted signature is created and added to the list of encrypted signatures associated with the client's record and stored in the public cloud for auditing by a third party such as the central bank.

References

1. Dan, B., Matthew F. (2003), 'Identity-Based Encryption from the Weil Pairing', *SIAM J. of Computing* 32 (3), 586-615, 2003

2. Shefali M., (2012), Design a Fine Grain Role Based Access Control', Master Thesis, University of Colorado at Denver.

3. Yu-Yi C. & Jun-Chao L. & Jinn-Ke J. (2012), 'A Secure EHR System Based on Hybrid Clouds,' *Journal of Medical Systems* 36 (5), 3375-3384