Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 9852472, 8 pages
https://doi.org/10.1155/2019/9852472

Research Article

An Efficient Framework for Sharing a File in a Secure Manner Using Asymmetric Key Distribution Management in Cloud Environment

K. V. Pradeep,[1] V. Vijayakumar,[1] and V. Subramaniyaswamy[2]

[1]*School of Computing Science & Engineering, VIT, Chennai, India*
[2]*School of Computing, SASTRA University, Thanjavur, India*

Correspondence should be addressed to K. V. Pradeep; pradeep.kv@vit.ac.in

Received 18 February 2019; Revised 29 April 2019; Accepted 19 May 2019; Published 27 June 2019

Guest Editor: Arash H. Lashkari

Cloud computing is a platform to share the data and resources used among various organizations, but the survey shows that there is always a security threat. Security is an important aspect of cloud computing. Hence, the responsibility underlines to the cloud service providers for providing security as the quality of service. However, cloud computing has many challenges in security that have not yet been addressed well. The data accessed or shared through any devices from the cloud environment are not safe because they are likely to have various attacks like Identity Access Management (IAM), hijacking an account or a service either by internal/external intruders. The cryptography places a major role to secure the data within the cloud environment. Therefore, there is a need for standard encryption/decryption mechanism to protect the data stored in the cloud, in which key is the mandatory element. Every cloud provider has its own security mechanisms to protect the key. The client cannot trust the service provider completely in spite of the fact that, at any instant, the provider has full access to both data and key. In this paper, we have proposed a new system which can prevent the exposure of the key as well as a framework for sharing a file that will ensure security (CIA) using asymmetric key and distributing it within the cloud environment using a trusted third party. We have compared RSA with ElGamal and Paillier in our proposed framework and found RSA gives a better result.

1. Introduction

As per NIST definition, cloud computing is the one that enables us to share the configured computing resources on demand over the network, which requires less effort from management or from service provider to provision and release the resources rapidly [1]. It is a new pattern to avail the computational power with higher flexibility and availability at low cost [2]. IaaS, PaaS, and SaaS are the three service models evolved [3–5] and can be deployed as public, private, hybrid, and community.

When IT services are outsourced, there is a need to concern about privacy and security, particularly while moving the data from an organization environment to the cloud environment. The primary motivation is to reduce cost, reducing the responsibility of both security and privacy. All the outsourced services are accountable, and it is the responsibility of an organization to process, monitor, and address the various issues involved in both security and privacy. Along with this, an organization has to focus even on performance, availability, and recovery.

Accessing the data often from the data centers with the high-speed Internet by a software application through devices with mobility in nature is what comprises the term cloud. Using cloud computing, the concept has evolved that any kind of services are provisioned quickly on demand as per user requirements. Services offered by the cloud service providers (CSP) can be categorized into Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). Based on the usage of these services from the cloud comes with its own set of merits and demerits. Few may say that it involves unnecessary risk; others may argue that it is a way to minimize the cost for any kind of start-up business. Accessibility, availability, agility, scalability, and efficiency are the main reasons that make most of the business entities attracted and migrated to cloud.

Note that the concept foretold revolves around data, and whenever we talk about data we must address both security and privacy issues [6–9] involved around it, especially with the rate at which data are enormously out growing in quantity. The lack of a well-defined entrance and egress points or the ability to physically control arises the whole host of security issues like data leakage, backups and recovery, abuse of privileged user access, regulatory requirements, multitenancy issues, and many more. Thus, with many gaps to be filled, more work is required in the area of securing the data [10, 11] to make the concept of cloud more acceptable to cloud service consumers.

Cloud is used for storing the data by many organizations, but the computation is done at cloud service provider (CSP) due to its resource (CPU/memory) elasticity, pay as per use, access to data from any location throughout the world, no requirement of any installation and maintenance, and accessing at most relief from storage burden. Normally, the cloud is deployed as private, public, community, or hybrid. There is not a proper interface that can lead to threats to the cloud.

In cryptography [12–21], key is the vital component in the cryptosystem. It is one of the biggest challenging tasks to manage them because it is the state of being legally responsible for any loss beyond the encryption and decryption process, such as people and flawed policies. Just as in a physical lock, a key is vital for providing access. The question arises when the cloud service providers offer their own encrypted and decryption [22–25] mechanism to protect the data; they provide the key along with data. With the encrypted data and the key lying with the cloud service providers itself, they do act as owners and there is very little that is preventing them for abusing that right.

The rest of the paper is organized and ordered into various sections. In Section 2, all the related works associated with cloud storage are discussed and the methods used to overcome them are also reviewed. In Section 3, the proposed technique for storing the data and sharing the key among multiple users was discussed. In Sections 4 and 5, the experimental results with some simulation and analysis are done. Finally, in the last section, both conclusion and summarization of work are discussed.

2. Related Work

As we all know that the cloud provides the facility to avail software, platform, and infrastructure as services and can be availed as either public, private, or hybrid deployment models, most of the organizations prefer to have both computation and storage as a service from the cloud so that they can be free from storage and computation management, and accessing the data based on demand from anywhere, any time, with low cost and also does not require any installation and maintenance. The only thing that should be taken care is to provide secure interfaces and API's so that no malicious intruders (insider/outsider) can share the infrastructure or can hijack the account which may lead to threats in a cloud [26]. For example, booking a train, flight, or bus ticket or updating PAN card through online is possible through the cloud service transparently [27].

The data which are to be stored in the cloud are encrypted by any chosen cryptography technique so that not even cloud service provider (CSP) can access. Normally, users would like to store the data in the encrypted form, so it is necessary to protect the keys used for encrypting the data. Now, the question arises whether the keys are to be stored with either CSP or customer or any other third party authority. Sometimes the data which are stored in the cloud are to be shared among multiple members of the particular members [28–30] using group key.

In the paper [31], the author provides an interface as a service to manage files from different or multiple cloud service providers and also a tool to monitor, download, or upload throughput and single or multikey user accounts. Data stored in the public cloud environment need more attention if the data are more sensitive. Because customers do not have any control over that environment, even he/she will not know where exactly data will be stored or exist geographically. Therefore, it is needed to use any standard cryptography approach to protect both data and keys used. Managing the keys used to encrypt the user's data becomes an integral part, especially when multiple tenants were using the same cloud environment to share the resources. The importance of key management is not well addressed in the European network and Information Security Agency (ENISA) [32].

There are many standard cryptographic techniques available, and can be chosen one among them to store the data in an encrypted form. There are millions and billions of users using storage as a service, and keys used to encrypt and decrypt need to be protected. Now the problem arises, where the key is to be stored either at customer, provider, or any trusted third party.

In [33, 34], the authors introduce a public key infrastructure certificate scheme to secure the cloud application program to encrypt the key, which also binds the key with files based on File-Id attributes for personal storage. It is easy to maintain and manage the keys to update, but it is required to update the user credentials regularly. Suppose if the user credentials are weak, then the security of file cannot be assured. Here, the author encrypts the stored cloud data using an elliptic curve.

In [35], the author divides the data into several portions and the keys are generated randomly to encrypt each portion of data. In [36, 37], to protect the outsourced data or resources, the author uses overencryption which is a two-layered model; i.e., owners will encrypt the resources or data before uploading it to cloud. Once data are received in the encrypted form, now once again the cloud server will reencrypt it for the second time before other users download it.

Yan and Zhou [38] based on the hypergraphs proposed the key management which is scalable in nature and well suited for the various groups in an organization for large-scale applications. The key management based on keying hypergraph is a group key management.

3. Proposed Method

The functionalities of key management are to deal with the generation and storage of key along with its distribution among two parties. In cloud security, it plays an important role in the attacks on public key cryptographic algorithms. Here, we will be using a modified Diffie–Hellman distribution scheme for the distribution of keys which are used to encrypt and decrypt the data or files which are stored in a cloud on a secure channel.

Public key cryptographic systems are purely based on one-way functions, which consume less computing power to convert plain text into ciphertext. The reason behind the one-way function is that, in a reasonable amount of time, it is not feasible for anyone to bring back the plain text from the ciphertext (i.e., $\mathrm{C = fxb(P)}$ and $\mathrm{P \neq f^{-1}xb(C)}$, where- C = ciphertext, P = plain text, and Xb is a public key of party B). Diffie–Hellman key exchange is one of the most used key distribution method usually in a public key distribution system (PKDS) to exchange a piece of information.

3.1. How It Works.

3.1. How It Works. Assume two parties (P and Q) are involved in communication to exchange the keys over an insecure channel, and it follows as below:

(1) P and Q chooses a large prime number "n" and a primitive element "a" (both are public) along with the one-way function $Y = f(x) = g^x \bmod n$

(2) Both the parties P and Q select their own random numbers to say "p" and "q" and compute the values "M" and "N," respectively, and send them to each other

 (a) P computes $M = g^p \bmod n$ and sends M to Q
 (b) Q computes $N = g^q \bmod n$ and sends N to P

(3) Upon receiving, both computes the following:

 (i) P computes $S = N^a \bmod n = (g^{pq} \bmod n)$
 (ii) Q computes $S = M^b \bmod n = (g^{pq} \bmod n)$

(4) Now both the parties P and Q have the same key which is shared over the insecure communication channel

This communication uses a one-way function, which is extremely hard to get back $x = f^{-1}(y)$, even by knowing the values of M, N, n, and g.

From Figure 1, public key cryptography is used to secure and share a file from a cloud computing environment. The proposed framework consists of 5 entities like cloud storage (CS), cloud key management system (CKMS), certificate authority (CA), data owner, and users.

(1) Data owner is the one who uploads the file into the cloud storage and has full control over it. Each file is identified by its File-Id.

(2) Users are the one, who would like to seek permissions and access the file stored in the cloud.

(3) CKMS plays an important role in cryptography, which provides confidentiality (encryption and decryption), authentication of an entity, data origin authentication, integrity, and digital signatures with the help of CA (certificate authority)

(4) CA is responsible for establishing and vouching for the authenticity of public keys.

(5) CS is simply used to store the files, as per KMS and CA approval.

Initially, the owner of the file, which is to be uploaded in the cloud environment, has to register with CKMS by creating an account with login credentials. The same login credentials can be used even to upload the list of users and their permissions with whom the uploaded file can be shared. Based upon the user login details, the CKMS uses asymmetric encryption to generate both public and private key pair associated with the user details. After receiving the request from the file owner to upload the file in the cloud environment, the CKMS will encrypt it by using the private key of an associated file owner, and then, the encrypted file is stored in the cloud storage. Any certified and a standard asymmetric or symmetric key algorithm can be used, which is not discussed in our paper (e.g., RSA and AES). All the above process is being done by CKMS with the help of the certificate authority (CA) and its working strategy is given in Section 3.2.

3.2. Encryption Process. In the above process, the generated private key (R_{k}) is used to encrypt the file and then the file is stored in the cloud. The public key (U_{k}) is divided into two parts by CKMS: one part of it is transmitted to the owner, and rest is kept with CKMS.

Later, at the time of decryption, the whole key is used to decrypt the uploaded file. Algorithm for generating the keys and encrypting the file is given in Figure 2, and its notations with definitions are provided in Table 1.

3.3. How Certificate Authority (CA) Works. Normally, it comprises with registration authority, name server, key generator, and certificate directory. The relationships with these components are illustrated in Figure 3:

(a) Certificate authority: it is responsible to bind the public keys to the distinguished user through signed certificates, managing them with the serial number, revocation of certificate, and also to authenticate the public keys

(b) Name server: unique user names are managed

(c) Registration authority: entities are authorized by unique names

(d) Key generator: it generates public/private key pairs and symmetric keys or passwords

(e) Certificate directory: it is a database used to store/organize/manage all the user certificates and also permits the CA

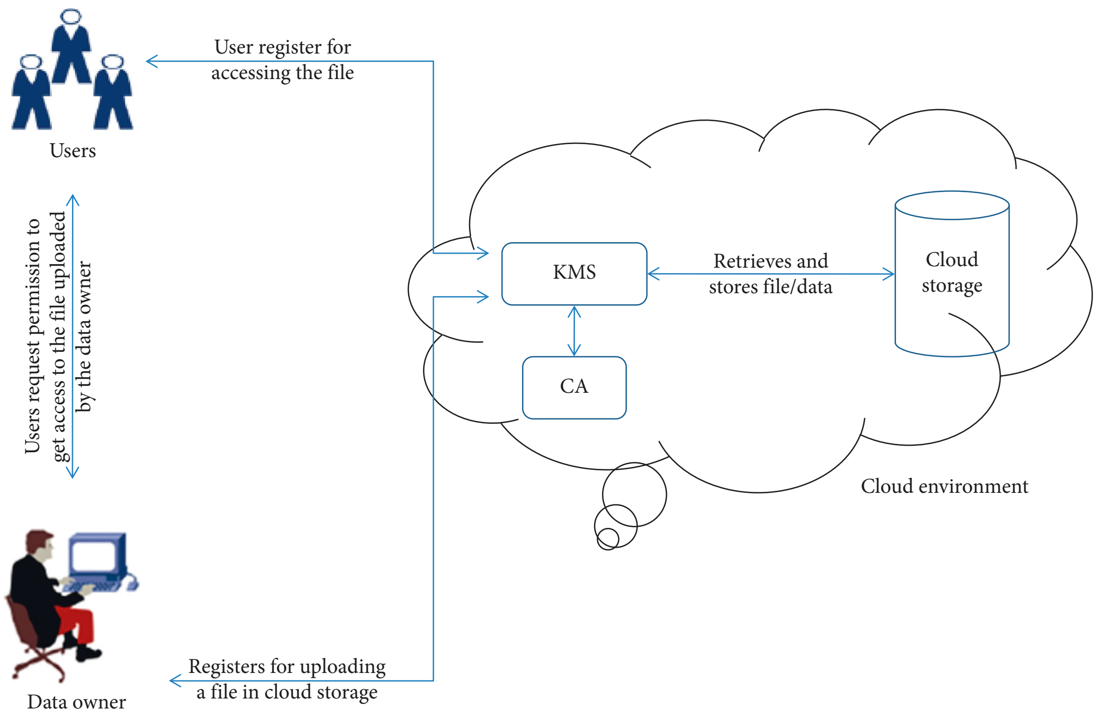
User register for accessing the file
Users
Users request permission to get access to the file uploaded by the data owner
KMS
Retrieves and stores file/data
Cloud storage
CA
Cloud environment
Data owner
Registers for uploading a file in cloud storage

Figure 1: Block diagram of the proposed framework.

```
Algorithm for encrypting the file by cloud-KMS

Input: file ($F$), length of the key ($K_L$), asymmetric encryption ($A_{enc}$)
Output: private key($R_k$), public key ($U_k$), owners key($O_k$), KMS key ($U_{kms}$)
If (Owner = TRUE) then do the following
    (1) Use $A_{enc}$ to generate $U_k$ and $R_k$ (public and private keys)
    (2) $E_F = A_{enc}(F$ and $R_k)$        // encryption of file using private key
    (3) $U_k = (U_{k1}$ and $U_{k2})$       // divides the public keys into 2 parts
    (4) $U_0 = U_{k1}$
    (5) $U_{kms} = U_{k2}$
    (6) DELETE($R_k$, $U_k$)          // delete public and private keys
endif.
```

Figure 2: Encryption process by CKMS.

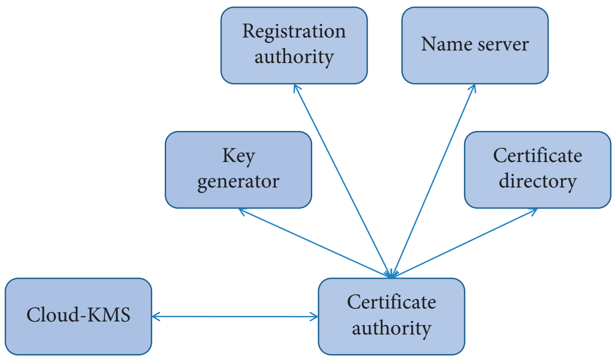

Figure 3: Services related to certificate authority.

Table 1: Notations and definitions.

Notations	Definitions
A_{Enc}	Asymmetric encryption algorithm
U_k	Public key
R_k	Private key
U_{kms}	KMS's part of public key
U_o	Owner's part of public key
E_f	Encrypted file
L_{users}	List of users
APL_{users}	Access permission list
F	File to be uploaded in the cloud
F_{id}	File identifier

3.4. Decryption Process. The file owner shares and uploads the user list and their access permission list with the CKMS and also sends an invitation to all the users who are all in the list to access the file uploaded in the cloud along with the part of owner's public key for self-registration. A user can access the file uploaded, by registering with CKMS. Once registration is completed, the user will receive the login credentials. Upon receiving, the user requests the file through file identifier (F_{id}) along with the part of the owner's public key. Now CKMS checks the user permission details from access permission list associated with file identifier. If the list contains the user information and access permissions, then CKMS regenerates the public key by combining the parts, i.e., the one which received from the user with the part of public key it is having.

During this process, it also downloads the file which is in the encrypted form, from the cloud storage. Now with the generated public key, it decrypts the downloaded file and sends back to the corresponding user. Key translation protocol is used to distribute and exchange the key between users, owner, and CKMS (Figure 4).

```
Algorithm for decrypting the file by cloud-KMS

Input: $F_{id}$, $E_F$, $U_{kms}$, $L_{user}$, $U_o$, $APL_{users}$, $A_{enc}$
Output: $F_{id}$, $F$
(1) If (($L_{users}$ = TRUE) && ($APL_{users}$ = TRUE)) then
(2) If ($F_{id}$ = TRUE)
(3) Use $A_{enc}$ to decrypt the File
(4) $U_k \longleftarrow$ concatenate ($U_{kms}$, $U_o$)
(5) $F \longleftarrow D(F, U_k)$
(6) DELETE ($U_k$)
(7) Else "File Not Found"
(8) Endif.
```

Figure 4: Decryption process by CKMS.

```
Key translation protocol used by cloud-KMS

Summary: A interacts with a KMS (T) and the party B.
Result: A transfers a secret message "M" to B.
(1) Notation: $E$ is a symmetric encryption algorithm
(2) One-time setup: A and T share key $K_{AT}$. Similarly, B and T Share $K_{BT}$.
(3) Protocol messages:
    (a) $A \longrightarrow T : A, E_{K_{AT}}(B, M)$
    (b) $T \longrightarrow A : E_{K_{BT}}(M, A)$
    (c) $A \longrightarrow B : E_{K_{BT}}(M, A)$
(4) Protocol actions:
    (a) A encrypts M under $K_{AT}$ and sends to T with its own identifier.
    (b) Upon decrypting the message, T determines it is intended to B, Look up
        the key ($K_{BT}$) of the indicated recipient, and reencrypts M for B.
    (c) T returns the translated message for A to send to B.
    (d) Finally, A sends it to B.
```

Figure 5: KT protocol used by CKMS.

CKMS acts as key translation centre (KTC), which allows any two parties (owner to the user) or (CKMS to owner/user), which do not directly share any kind of keying material, to establish a secure communication through the use of long-term keys K_{AT} and K_{BT} with the CKMS. The key translation algorithm is illustrated in Figure 5.

Suppose if the user requesting an uploaded file from cloud to access is not present in the user list, then obviously CKMS will deny the permission.

3.5. Implementation Details. The web-based application is designed to implement the proposed methodology in the cloud environment in OpenStack with the help of JAVA technology. The Graphical User Interface is created and used for self-registration by the owner to access or upload the file, respectively. All users and owners make use of this GUI to register with CKMS and set up the credentials. Upon registrations, file owner will also upload the intended user list and access control or permission list in CKMS using the interface provided. Once the user list is added, these users will get the invitations either from CKMS or owner for self-registration. Now CKMS uses a proposed asymmetric encryption method to encrypt the file which too is uploaded in the cloud environment. Normally, here we have used the RSA-the asymmetric encryption algorithm to generate the keying material (i.e., both public/private keys) and private key is used to encrypt the file which is to be stored in the cloud storage. For all the validations, certification is been verified with certification authority (CA) by CKMS.

The commonly used mechanisms to generate the key in public key infrastructures are Rivest, Shamir, and Adelman (RSA), Diffie–Hellman (DH), digital signature (DS), and so on. The framework which we have proposed performs all the fundamental functionalities of key management such as key distribution and generation, encryption, and decryption, and all are tested on two instances in OpenStack (Virtual Machines) with 4 GB of RAM and a dual-core 1.4 GHz Intel Core i5 processor each. Finally, the performances of the different public key infrastructure algorithms are considered and compared to the process of encryption and decryption, usage of CPU and memory. We carried out experiments on multiple text files with different key sizes like 48 bits, 128 bits, 256 bits, 512 bits, 1024 bits, and 2048 bits.

Since CKMS uses key translation protocol, the parties involved (owner, user, and CKMS) can exchange the messages or keying material more effectively, efficiently in an insecure communication channel.

4. Experimental Results

The time plays a vital role during the generation of key, encryption, and decryption process. All investigations are made on Intel(R) Core-2 with CPU 1.4 GHz processor, 4 GB RAM on Windows-10 working framework by using JAVA. Any algorithm can be measured with the time parameter that the system has consumed. If anyone wants to design an optimized system, it must ensure that the above three in the cryptographic system designed must consume less time. Here, in Figure 6, we have shown the comparative analysis of all the three algorithms of time taken to generate the key. We have found that RSA algorithms consume less time compared to all the other two algorithms irrespective of key size. Here, we have analyzed only time taken to generate the key; likewise, we can even do for time consumed for the encryption and decryption process also because they are the most important to provide the security in a cloud environment which is not focused here in our paper.

The result in Figure 7 shows that the RSA algorithm utilizes fewer resources like CPU, memory, and RAM when compared to ElGamal and Paillier during encryption and decryption of the files stored in the cloud. As we all know, the RSA has limitations while handling the large file (bytes), which consumes more resource utilization. For smaller files, the proposed architecture gives better result by consuming fewer CPU cycles.

Here, we have tested the proposed architecture on the files with different sizes like 48 MB, 64 MB, 128 MB, 256 MB,

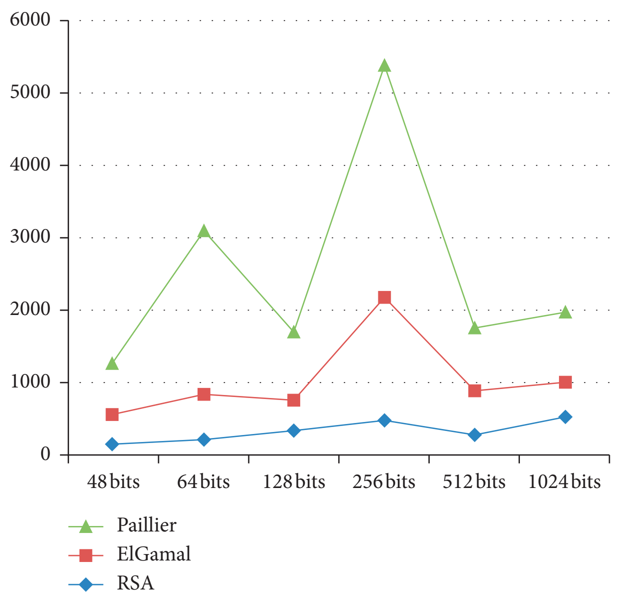

FIGURE 6: Time consumed to generate a key of different sizes by CKMS used by 3 different algorithms.

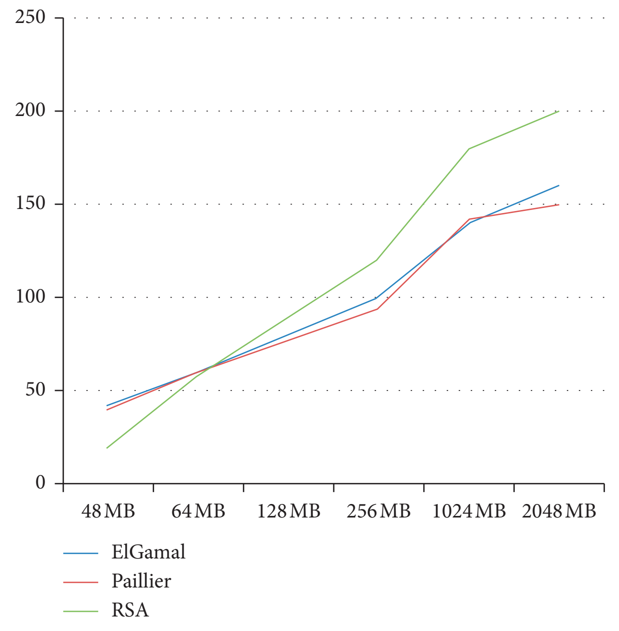

FIGURE 7: Percentage of CPU usage for different public key algorithms like ElGamal, Paillier, and RSA.

1024 MB, and 2048 MB and the plotted against CPU cycles, as shown in Figure 7.

Apart from CPU usage and time consumption to generate the key, we have investigated the time consumed for encryption and decryption in Figures 8 and 9, respectively. From these figures, we can analyze that RSA consumes less time for encryption and ElGamal consumes less time for the decryption process. ElGamal is usually slower than symmetric ones for the same level of security, so it is faster to encrypt the symmetric key with ElGamal and the message (which can be arbitrarily large) with a symmetric cipher.

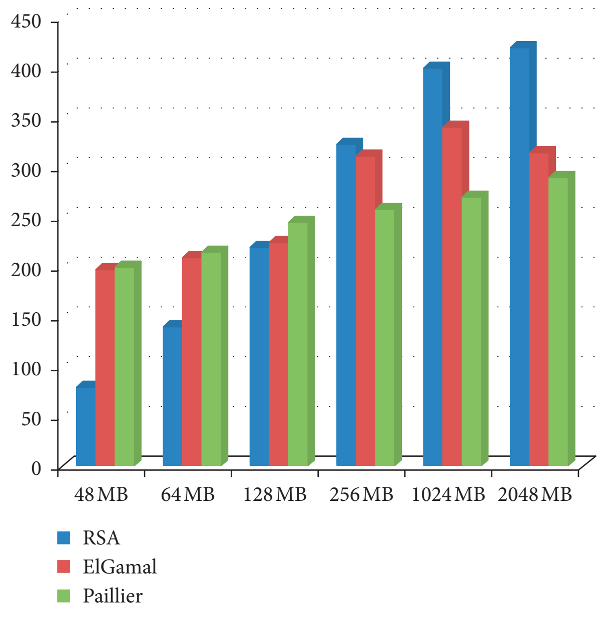

FIGURE 8: Time consumption during the encryption process.

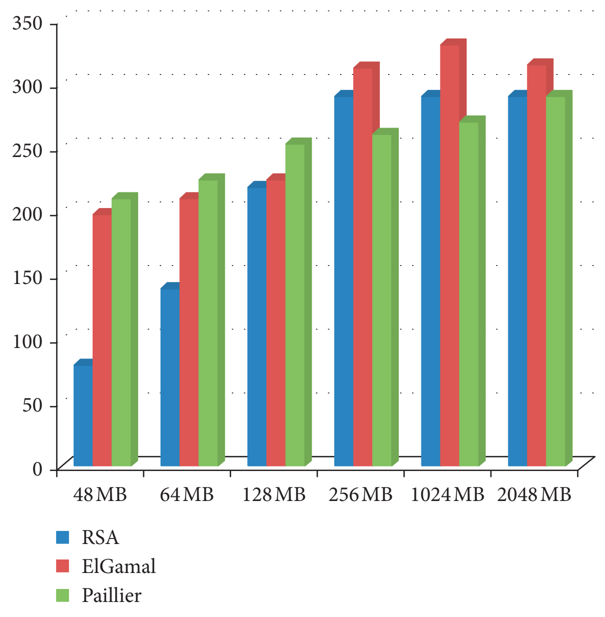

FIGURE 9: Time consumption during the decryption process.

5. Complexity Analysis

As from the results, we can easily say that RSA is better than ElGamal and Paillier. The complexity analysis w.r.t time for RSA during encryption/decryption is $O(\log N^3)$. If the public "e" is fixed then computing M^e, Mod N has the time complexity $O(\log N^2)$ which is better than other Paillier and ElGamal.

In the encryption, the RSA performs better than ElGamal in all cases. In the decryption process, the ElGamal outperforms RSA, meaning that text messages are decrypted faster by ElGamal than by the RSA technique. When viewed as a single tool, the RSA is superior to the ElGamal algorithm

Table 2: Homomorphic encryption cryptosystems.

Characteristics	RSA	Paillier	ElGamal
Platform	Cloud computing	Cloud computing	Cloud computing
Homomorphic type	Multiplicative	Additive	Multiplicative
Asymmetric in nature	Yes	Yes	Yes
Data privacy	All ensures privacy both in communication and storage (data at rest)		
Data security	Yes	Yes	Yes
Key used by	Separate keys are used for encrypt/decrypt either by customer/provider		
Speed (enc/dec)	Fast/slow	Slow/slow	Slow/fast

in terms of computational speeds. It is the reason why the RSA algorithm has been and is still being used in designing many security protocols for data communication (Table 2).

6. Conclusion and Future Directions

In this paper, we have proposed the framework to share a file in a secure manner using public key infrastructure in the cloud environment to store, share, and download by ensuring and satisfying the three important property of security. In the proposed architecture, CKMS plays a vital role and protect the file, which is shared among multiple users and protected from the intruder either from inside or outside. And also a small attempt is made to compare the different public key infrastructure algorithms in terms of time consumed to generate the key and time consumed during the process of encryption and decryption. Key translation protocol is used to share the key among owner, users, and CKMS, which will also ensure authenticity. Here, in the proposed architecture, we focused only on providing privacy for the file or the data stored in the cloud, but not on access level control. So the biggest open challenge is to handle efficiently the revocation mechanism, accountability, and user access control. Finally, we tested the metrics like time consumption for encryption/decryption and CPU and memory usage for the different PKI algorithms.

Data Availability

As a part of my research work, we used a multiple virtual machine within the private cloud and monitored the parameters like CPU time and its usage while generating the key of different sizes with respect to algorithms and also consumption of time during encryption and decryption of the proposed algorithm as mentioned in the paper. All the required metadata are also mentioned in the implementation section.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Mell and T. Grance, *The NIST Definition of Cloud Computing, Version 15*, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2009, http://csrc.nist.gov/groups/SNS/cloud-computing.

[2] G. Fowler and B. Worthen, “The Internet industry is on a cloud–whatever that may mean,” *Wall Street Journal*, vol. 26, 2009.

[3] N. Leavitt, *Is Cloud Computing Really Ready for Prime Time?*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2009.

[4] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” *Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[5] L. Youseff, M. Butrico, and D. Da Silva, “Toward a unified ontology of cloud computing,” in *Proceedings of the 2008 Grid Computing Environments Workshop IEEE*, Austin, TX, USA, September 2008.

[6] A. Gholami and E. Laure, “Security and privacy of sensitive data in cloud computing: a survey of recent developments,” *Computer Science and Information Technology*, vol. 5, 2016.

[7] R. P. Padhy, M. R. Patra, and S. C. Satapathy, “Cloud computing security issues and research challenges,” *International Journal of Computer Science and Information Technology & Security*, vol. 1, no. 2, pp. 136–146, 2012.

[8] K. Manpreet and H. Singh, “A review of cloud computing security issues,” *International Journal of Advances in Engineering and Technology*, vol. 8, no. 3, p. 397, 2015.

[9] H. Suo, Z. Liu, J. Wan, and K. Zhou, “Security and privacy in mobile cloud computing,” in *Proceedings of 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 655–659, Sardinia, 2013.

[10] S. Nepal, C. Friedrich, Henry et al., “A secure storage service in the hybrid cloud,” in *Proceedings of 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC)*, New York, NY, USA, 2011.

[11] Y. Zhang, Q. Chen, and S. Zhong, “Privacy-preserving data aggregation in mobile phone sensing,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 980–992, 2016.

[12] H. Kwon, C. Hahn, D. Koo, and J. Hur, “Scalable and reliable key management for secure deduplication in cloud storage,” in *Proceedings of the IEEE 10th International Conference on Cloud Computing (CLOUD)*, Honolulu, HI, USA, June 2017.

[13] K. V. Pradeep and V. Vijayakumar, “Survey on the key management for securing the cloud,” *Procedia Computer Science*, vol. 50, 2015.

[14] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Proceedings of Advances in Cryptology (Eurocrypt ‘99)*, pp. 223–238, Prague, Czech Republic, May 1999.

[15] Z. E. Dawahdeh, S. N. Yaakob, and A. M. Sagheer, “Modified ElGamal elliptic curve cryptosystem using hexadecimal representation,” *Indian Journal of Science and Technology*, vol. 8, no. 15, pp. 1–8, 2015.

[16] A. R. Buchade and R. Ingle, “Key management for cloud data storage: methods and comparisons,” 2014.

[17] H. Nayana and S. S. Manvi, “Thesis proposal summary: key management authentication and non repudiation for

information transaction in vehicular cloud environments," in *Proceedings of the 2016 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) IEEE*, Bangalore, India, October 2016.

[18] O. Ethelbert, F. F. Moghaddam, P. Wieder, and R. Yahyapour, "A JSON token-based authentication and access management schema for Cloud SaaS applications," in *Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud) IEEE*, Prague, Czech Republic, August 2017.

[19] R. Chandramouli, S. Chokhani, and M. Iorga, "Cryptographic key management issues & challenges in cloud services," NIST Interagency or Internal Report 7956, 2013.

[20] H. Wang, Y. Zhang, H. Xiong, and B. Qin, "Cryptanalysis and improvements of an anonymous multi-receiver identity-based encryption scheme," *IET Information Security*, vol. 6, no. 1, pp. 20–27, 2012.

[21] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer, and J. A. Manjón, "Fast transmission to remote cooperative groups: a new key management paradigm," *IEEE/ACM Transactions on Networking*, vol. 21, pp. 621–633, 2013.

[22] S. Rajarajeswari and K. Somasundaram, "Data confidentiality and privacy in cloud computing," *Indian Journal of Science and Technology*, vol. 9, no. 4, pp. 1–8, 2016.

[23] R. Sugumar and S. B. S. Imam, "The symmetric encryption algorithm to secure outsourced data in public cloud storage," *Indian Journal of Science and Technology*, vol. 8, no. 23, pp. 1–5, 2015.

[24] R. Saikeerthana and A. Umamakeswari, "Secure data storage and data retrieval in cloud storage using cipher policy attribute based encryption," *Indian Journal of Science and Technology*, vol. 8, no. S9, pp. 318–325, 2015.

[25] L. Harn, C.-C. Chang, and H.-L. Wu, "An anonymous multi-receiver encryption based on RSA," *Journal of Network Security*, vol. 15, pp. 307–312, 2013.

[26] C. Alliance, "Security guidance for critical areas of focus in cloud computing v3.0," *Cloud Security Alliance*, vol. 15, 2011.

[27] J. Wayne and G. Timothy, *NIST Guidelines on Security and Privacy in Public Cloud Computing*, Draft Special Publication, NIST, Gaithersburg, MD, USA, 2011.

[28] S. Dorwani, J. Aghav, and B. Bhutkar, "NoSQL: features, classification, and comparison," in *Proceedings of the IEECS-2013*, Coimbatore, India, April 2016.

[29] L. Harn and C.-F. Hsu, "A Practical Hybrid Group Key Establishment for Secure Group Communications," *The Computer Journal*, vol. 60, no. 11, pp. 1582–1589, 2017.

[30] K. Bicakci, D. Deniz Yavuz, and S. Gurkan, "TwinCloud: secure cloud sharing without explicit key management," in *Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS)*, Philadelphia, PA, USA, October 2016.

[31] H. Dinh, A. Dworkin, C. O'Neill et al., "Omnibox: Efficient cloud storage by evaluating dropbox and box," in *Proceedings of the 2017 24th International Conference on Telecommunications (ICT) IEEE*, Limassol, Cyprus, May 2017.

[32] I. Indu, P. M. Rubesh Anand, and S. P. Shaji, "Secure file sharing mechanism and key management for the mobile cloud computing environment," *IJST*, vol. 9, 2016.

[33] G. Jindi, D. Zishan, and S. Lei, "Research on key management infrastructure in cloud computing environment," in *Proceedings of the International Conference on Grid and Cooperative Computing*, pp. 404–407, Nanjing, China, November 2010.

[34] A. Kumar, B. G. Lee, H. J. Lee et al., "Secure storage and access of data in cloud computing," in *Proceedings of the 2012 International Conference on ICT Convergence (IC-TC)*, pp. 336–339, New York, NY, USA, October 2012.

[35] F. H. Chen, Y. Liu, and Y. U. Qi, "Key management scheme of personal cloud computing," *Modern Computer*, vol. 30, 2011.

[36] S. D. C. D. Vimercati, S. Foresti, S. Jajodia et al., "Over-encryption: management of access control evolution on outsourced data," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, Vienna, Austria, September 2007.

[37] A. Armando, R. Carbone, L. Compagna, J. Cuéllar, G. Pellegrino, and A. Sorniotti, "An authentication flaw in browser-based single sign-on protocols: impact and remediations," *Computers & Security*, vol. 33, pp. 41–58, 2013.

[38] D. Yan and X. Zhou, "Secure group communications using key hypergraphs," *Journal of Computational Information Systems*, vol. 8, pp. 5035–5042, 2012.