Securing medical data by role-based user policy with partially homomorphic encryption in AWS cloud

BOOMIJA M.D (boomija.md@gmail.com)
SRM Institute of Science and Technology https://orcid.org/0000-0003-3516-2344
Kasmir Raja S.V.
SRM Institute of Science and Technology

Research Article

Keywords: Partially homomorphic encryption, cloud security, access policy, AWS S3, IAM, Elastic Beanstalk

Posted Date: January 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1047189/v1

Version of Record: A version of this preprint was published at Soft Computing on March 22nd, 2022. See the published version at https://doi.org/10.1007/s00500-022-06950-y.

Abstract

Cloud technology provides services for storing and accessing a large amount of data with ease of access and less cost. Sensitive data like patients' electronic health information should be encrypted before outsourcing into the cloud. Many traditional encryption methods are used for protecting data in the cloud, but unable to perform computation on encrypted data. Homomorphic encryption operates directly on the ciphertext. In this study, a Secure Partially Homomorphic Encryption (SPHE) algorithm is proposed to secure the outsourced data and perform multiplication and division operations on the ciphertext. The access control policy in the cloud environment is more flexible. An attacker can easily collect sensitive data by abusing the access policy of another user. So the database privacy is compromised. Creating a role hierarchy and managing the session is difficult in the cloud environment. The above issues motivate us to develop a model which is the integration of the proposed scheme SPHE with role-based user policy. The model is implemented in Eclipse IDE and AWS Toolkit for Eclipse and deployed in Amazon Elastic Beanstalk (EB) environment. This model is particularly used for securing the patient e-health details and performing computation on outsourced data. The patient details are encrypted by the algorithm SPHE and uploaded in AWS (Amazon Web Service) S3 bucket. The users are created by AWS Identity and Access Management (IAM) service and the access level policy is defined based on user roles in EB environment. The proposed model performance is studied by comparing with other partially homomorphic methods Elgamal, Pailler, and Benaloh. This model achieves data integrity and data confidentiality using the role-based user policy with SPHE.

1. Introduction

Cloud is used for storing a large volume of data with minimum cost. The data owners outsource their data to the cloud server. The outsourcing of data will compromise the integrity and security since the cloud is a semi-trusted platform.[19] So, sensitive details like medical information should be encrypted then uploading into the cloud. Many cryptographic algorithms are used to encrypt sensitive details. Sometimes, computation needs to be done for the outsourced data. The computation can be done after decrypting the ciphertext which is compromised the data security.

A homomorphic encryption method is a possible solution that will perform computation on encrypted data. [21] We proposed the algorithm SPHE which performs multiplication operation and division operation on the ciphertext.

[22] The outsourced data has to be shared by different users to perform various operations. The user roles should be assigned by access policy for the data. [23] The access policy allows the users to perform required operations. This motivates us to create a model which combines the SPHE with a role-based user policy.

This method is implemented by Eclipse IDE for Java EE with AWS toolkit for Eclipse. The method is deployed in the AWS Elastic Beanstalk service which is an easy way for creating the environment for the

application. The data owner encrypts the patient details by the SPHE method and outsources the data to the AWS cloud.

The patient medical detail is stored in AWS S3 bucket. The application can use the data from the S3 bucket. The user roles are created by AWS Identity and Access Management (IAM) service. The access policy is defined that depends on the user roles which will restrict the access of encrypted data. The access levels list, read, write, and tagged are assigned to the data owner who is considered as administrator user role. For the evaluator, the access levels are list and read-only, who can read the data and perform computation on the data. The role assigned for the evaluator is the power user. The patients are end-users who can able to read the data. The different access policies are administrator policy, evaluator policy, and end-user policy that are created to define the access levels of each type of user. In Amazon S3, the client-side encryption is enabled to upload the encrypted data by the data owner. To integrate Elastic Beanstalk with Amazon S3, the reference of Amazon S3 is set to EB environment when creating the application.

The paper is organized as follows. Section 2 describes the related work. Section 3 presents the proposed model. In section 4, the proof of the SPHE algorithm is given. The model implementation with result discussion is presented in section 5. Section 6 shows the comparative study with other encryption methods. Finally, we conclude in section 7.

2. Related Work

This section describes the related work in this study. Zhang *et al.* [1] proposed a role-based with time-bound access control scheme to ensure the security of health details in the cloud. The authorized user for the system can access data that depends on their role within the time. A time tree scheme is used for creating the time-bound mechanism for access control. The sensitive detail is encrypted before uploading into the cloud servers. Bertino [2] the role-based access control method is used for assigning roles for the user depends on their position in the organization. This mechanism is used for security for the resources. The user access policy provides authorization based on roles.

Lan Zhou et al.[3] addressed various issues in trust using an RBAC model. This model considers the users and their roles and feedbacks. Liu et al [4] supports user revocation mechanisms and proposed an RBAC method for health records based on two roles, one for medical authority and another for patients. The medical authority can be identified by the access policies and roles. In [5], the authors proposed a user authentication method with role-based access control in smart health systems. The method used three types of parties health professionals, health authority, and consumer.

[6] Implements attributed-based framework with secure provenance to ensure effective, fine-grained access for health data. [7] Proposed a model which combines the methods of attribute-based encryption and an access policy that is hidden for cloud environment. The attribute-based encryption for audit-free cloud storage is implemented in [8] to manage the access control policies. [9] Supports attribute to user roles to control the privileges, but it is not supported for evolving environments. Khan and Sakamura [10]

proposed a fine-grained access control for the digital healthcare model. The method used eTRON which encrypts data by the public key cryptography and establishes a key sharing model using the Diffie-Hellman method.

Xu et al. [11] introduced a robust method that uses multiple attribute authorities in a public cloud. This scheme resolves a single-point computation bottleneck and ensures security from insider attacks by using different attribute authorities. Sandhu et al. [12] proposed a method for role-based access control which allocates functions to users and roles are combined with access right permissions. But this scheme is an expensive process. Pussewalage and Oleshchuk [13] uses a proxy re-encryption which provides client revocation based on the requirement. This scheme combines both roles and attributes and analyzes the request with pseudo-roles.

[14] Proposed a hybrid scheme that uses RSA and homomorphic methods for increasing the data security for the OpenStack environment. The cloud clients can control the key management and cryptographic operations instead of the cloud provider. [15] Proposed the attribute-based, privacy-preserving authentication method for health records. It authenticates the user, based on attributes. Homomorphic encryption ensures data security by preserving the attributes' privacy but the cost of computation is very high. Barni *et al.* [16] considered a multiparty method to process electrocardiogram data which is in encrypted form. The approach uses a homomorphic encryption method to ensure patient privacy. Carpov *et al.* [17] implemented a practical medical diagnosis model which is privacy-preserving using a homomorphic scheme. The user will encrypt data and upload it into the cloud environment. The evaluation can be done for the data which is in encrypted form, so the cloud owner is unaware of the uploaded data.

Gentry [18] proposed the fully homomorphic method which can perform addition operation and multiplication operation on the ciphertext. The somewhat homomorphic method allows restricted operations on ciphertext using circuits of specified depth. The fully homomorphic encryption methods are impractical because of their difficult computation.

3. Secure Partially Homomorphic Encryption (Sphe)

3.1. Preliminaries

The proposed method is secure and having probabilistic property. An attacker is not able to guess the plaintext from the encrypted text. The scheme used a probabilistic factor which adds noise for encrypted text. A randomly generated prime number is used when encrypting the plain text. So it is difficult to get the original text without keys. The encryption process used the one-way functions, modulo inverse operation for encryption, and modulo exponentiation operation for decryption.

The algorithm has five parts, key generation algorithm, encryption algorithm, decryption algorithm, multiplication algorithm, division algorithm. The key generation algorithm generated the public key pk and secret key sk and p is randomly selected prime number in Zp*, the multiplicative group of a finite field

$Zp^* = \{if\ a\ \varepsilon\ Zp: gcd\ (a, p) = 1\}$ where $Zp = \{1, 2,... p-1\}$. g is randomly selected generator element of the subgroup Gq of Zp*, where q = p/2. To give input p and g, the method outputs a public key (g,k,p) and secret key S, Where S is uniformly selected factor of Zp* and compute $k = S^g \bmod p$ as follows:

$S \leftarrow Zp^*$,

$k \leftarrow S^g \bmod p$

In the encryption algorithm, two probabilistic factor r, where $1 < r < q/2$ is used for encryption to give more security of data.

$r \leftarrow Z_{q/2}$,

$m \leftarrow g.r$,

$n \leftarrow pt / k^r \bmod p$

To decrypt a ciphertext ct = (m,n), the decryption scheme computes

$pt \leftarrow n.S^m \bmod p$,

where S is the secret key

3.2. SPHE Algorithm

SPHE-Key Generation Algorithm
Input: Large prime number p 1. Select a large prime element p in Zp*, Zp* is the multiplicative group of a finite field 2. Generate primitive number g ε $Z_{p/2}$* where gcd (g, p) = 1 3. Select an integer S ε {0,1,.....p-2} where gcd (S, p) = 1 4. Compute k = S^g mod p **Output**: (pk,sk) The public key, pk = (g, k, p) & secret key, sk = (S)
SPHE-Encryption Algorithm - Encrypt(pk, pt)
Input: message/plain text: pt, where pt ε Zp pk = (g, k, p) 1. Select random r ε { 2,...., (p – 1)/2 } 2. Compute m = g. r 3. Compute n = pt / k^r mod p **Output**: ciphertext ct = (m,n)
SPHE-Decryption Algorithm - Decrypt(ct)
Input: ciphertext ct Compute pt = n. S^m mod p **Output**: Plain Text message pt
SPHE-Multiplication Algorithm Multply(pk,ct1,ct2)

SPHE-Key Generation Algorithm
Input: cipertexts ct1, ct2 and pk ct1 = (m1,n1) ct2 = (m2,n2) 1. Pre compute k = S^g mod p 2. m1 = g. r1, n1 = pt1 / kr1 mod p where r1 is a random number 3. m2 = g. r2, n2 = pt2 / kr2 mod p where r2 is a random number 4. compute multiply(pk, ct1.ct2) MRes = (pt1 / k^{r1} mod p) (pt2 / k^{r2} mod p) 5. Decrypt the multiplied ciphertext MRes using secret key, D(sk, Res) = pt1.pt2 **Output : pt1.pt2**
SPHE-Division Algorithm – Divide(pk,ct1,ct2)
Input: cipertexts ct1, ct2 and pk ct1 = (m1,n1) ct2 = (m2,n2) 1. Pre compute k = S^g mod p 2. m1 = g. r1, n1 = pt1 / kr1 mod p where r1 is a random number 3. m2 = g. r2, n2 = pt2 / kr2 mod p where r2 is a random number 4. compute divide(pk, ct1.ct2) DRes = (pt1 / kr1 mod p) / (pt2 / kr2 mod p) 5. Decrypt the divided ciphertext DRes using secret key, D(sk, Res) = pt1/pt2 **Output : pt1.pt2**

4. Proof Of Sphe

The SPHE algorithm is semantically secure for every randomly selected element R = {Rn}, n ε p of polynomial randomly selected variables which gives probabilistic polynomial time algorithm $P_{A,}$ there is a probabilistic time algorithm P_A^1. Given the prior information H(Rn), no such algorithm P_A can obtain any information of F(Rn) from the probabilistic polynomial time algorithm P_A^1, cipher text that cannot be computed by P_A^1 .

Proof

Consider plain text or message in message space

pt : pt ε Gq,

Compute k = S^g mod p, where S is secrete key, g is generator, p is large prime number in Zp*

Encrypt(g, k, p), message pt

Compute m = g.r, where r is a randomly selected element and

n = pt / S^r mod p

The cipher text ct = (m,n)

Decrypt(ct)

Compute pt = n.S^m mod p, where S is the secrete key

= (pt / k^r mod p). $S^{(g.r)}$ (mod p)

= (pt / $S^{(g\ r)}$ mod p). $S^{(g.r)}$ (mod p)

(since k = S^g mod p)

= (pt. $S^{-(g\ r)}$ mod p). $S^{(g.r)}$ (mod p)

= (pt. $S^{-(g.r) + (g.r)}$ mod p)

Thus the plain text has been obtained using secrete key S

Example

Let us consider the message pt = 66

Key generation:

Consider the large prime number p = 54673

generator g = 7, where gcd(g,p)=1

Secret key S = 11

Compute $k = S^g \bmod p$

$= 11^7 \bmod 54673 = 23583$

Public key (g,k,p) =(7, 23583, 54673) and

secrete key S = 11

Encrypt(pt) =Encrypt(66)

Consider any random number r = 5, where 1<r<p-1 Compute m = g.r = 7. 5 = 35

Compute $n = pt / k^r \bmod p$

$=66 / 23583^5 \bmod 54673$

$=66. 23583^{-5} \bmod 54673$

$=66. (23583^{-1} \bmod 54673)^5$

$=66. (31455)^5$

=203232136685253069301875 0

Ciphertext ct = (m,n)

= (35, 203232136685253069301875 0)

Decrypt(ct)

$pt = n. S^m \bmod p$

$= 203232136685253069301875 0. 11^{35} \bmod 54673$

= 66

4.1 Homomorphic property of SPHE

The SPHE algorithm has homomorphic property. It has a partially homomorphic property that performs multiplication and division operations on any two encrypted messages as well as integer multiplication and division operations also possible. Consider two messages pt1 and pt2 in the message space, their

corresponding ciphertexts are ct1, ct2. The partially homomorphic encryption [20] which has multiplicative property can be proved as follows.

Let consider the key space k = sk = (S) and pk = (g, k, p). Let the plain text creates a group (pt, .) and cipher text creates a group (ct, ·), where · is the modular multiplication. For plaintext pt1 and pt2 in p and its corresponding ciphertext ct1,ct2 in ct, such that

ct1 = E(pk, pt1)

ct2 = E(pk, pt1)

The cloud provider can perform operations on the ciphertext ct by multiplication or division algorithm

4.2 Muliplicative property proof

multiply(pk, ct1 * ct2)

Precompute k = S^g mod p

m1 = g. r_1

n1 = pt1 / k^{r1} mod p

m2 = g. r_2

n2 = pt2 / k^{r2} mod p

multiply(pk, ct1* ct2) = (m1.m2, n1.n2 **)**

=(g r_1. g $r_{2\,,}$ (pt1 / k^{r1} mod p) * (pt2 / k^{r2} mod p)**)**

=(g r_1. g $r_{2\,,}$ (pt1 / k^{r1}) * (pt2 / k^{r2})) (mod p)

=(g (r_1 * $r_{2)\,,}$ (pt1 * pt2) / k^{r1+r2})) (mod p)

=E (pk, pt1 * pt2)

The cloud consumer can decrypt the evaluated cipher text using secret key S,

D(sk, ct1 * ct2) =(n1.S^{m1} mod p) * (n2.S^{m2} mod p)

= n1. n2 $S^{(m1 +m2)}$ (mod p)

= (pt1 * pt2) $k^{-(r1+r2)}$ $S^{(m1+m2)}$ (mod p)

= (pt1 * pt2) $S^{-g(r1+r2)}$ $S^{(g .r1 +g. r2)}$ (mod p)

[since $k = S^g \bmod p$ & $m = g.r$]

$= (pt1 * pt2)\ S^{-g(r1+r2)}\ S^{(g\ (r1\ +\ r2)}\ (\bmod p)$

$= pt1 * pt2\ (\bmod p)$

$= D(sk, ct1) * D(sk, ct2)$

Hence the proposed encryption scheme supports the multiplicative homomorphism.

4.3 *Division property proof*

divide(pk, ct1/ ct2)

Precompute $k = S^g \bmod p$

$m1 = g.r1$,

$n1 = pt1 / k^{r1} \bmod p$

$m2 = g.r2$,

$n2 = pt2 / k^{r2} \bmod p$

divide(pk, ct1/ ct2)

=($m1_{/}\ m2, n1_{/}\ n2$ **)**

= ($g\ r_1 / g\ r_{2\ ,}\ (pt1 / k^{r1} \bmod p) / (pt2 / k^{r2} \bmod p)$**)**

=($g\ r_1 / g\ r_{2\ ,}\ (pt1.\ k^{-r1}) / (pt2.\ k^{-r2}\ ))\ (\bmod p)$

$=($ $(r_1 / r_{2)\ ,}\ (pt1 / pt2)\ k^{-(r1+r2)})\ )\ (\bmod p)$

$= E\ (pk, pt1 / pt2)$

The cloud consumer can decrypt the evaluated cipher text using secret key S,

$D(sk, ct1 / ct2) = (\ n1.S^{m1} \bmod p) / (n1.\ S^{m1} \bmod p)$

$= n1_{/}\ n2\ S^{(m1\ +m2)}\ (\bmod p)$

$= (pt1/pt2)\ k^{-(r1+r2)}\ S^{(m1+m2)}\ (\bmod p)$

$= (pt1/pt2)\ S^{-g(r1+r2)}\ S^{(g\ .r1\ +g.\ r2)}\ (\bmod p)$

[since $k = S^g \bmod p$ & $m = g.r$]

$= (pt1 / pt2)\ S^{-g(r1+r2)}\ S^{(g\ (r1\ +\ r2)}\ (\bmod p)$

$= pt1 / pt2 \ (\bmod p)$

$=D(sk, ct1) / D(sk, ct2)$

Hence the proposed encryption scheme supports the division property.

5. Implementation Of The Proposed Scheme

The platform used for implementing the proposed scheme is Eclipse IDE for Java EE, JDK 8, Tomcat 8, AWS Toolkit for Eclipse. The AWS toolkit is integrated with Eclipse IDE by creating the access key as security credentials in the AWS console. The project is exported as a war file and uploaded in Elastic Beanstalk. Create a new application in EBS, and create a new webserver environment, select tomcat as the platform, upload and deploy the project war file in EB.

5.1 Network model

This model creates different user roles by using IAM in AWS. The admin user who is the application owner has all access rights to the application. They generated keys (pk,sk) using a Role-based SPHE scheme. The public key (pk) shared with the evaluator and the private key (sk) is shared with the end-user. The evaluator can perform computation on ciphertext using (pk). The customer can decrypt the result by sk.

EB is used for deploying and managing the application quickly in the AWS environment. It automatically handles load balancing, scaling, and environmental health monitoring. The application is deployed by Elastic Beanstalk (EB) in AWS. By using EB, create the environment and platform as a tomcat server to deploy the proposed scheme. The application is uploaded as a war file into EB.

The permission and policy boundary is created based on user roles for the service EB. The access levels associated with EB are list, read, tagged, and write. For admin user, all-access levels list, read, tagged, and write was enabled. The evaluator has list and read access permissions whereas the end-users have only read access permission. The permission boundary controls the data access and application by user role. So the users are restricted to access the service by access policy which gives a security model.

5.1.1. AWS managed user policies.

Create users using IAM for accessing [23]Elastic Beanstalk. The user roles are admin user, power user, and end-user. The admin is the data owner who has all access rights permission on data and applications. The power user is the evaluator, can perform operations on data. The end users are patients

who want to upload their medical details in the cloud. The application will allow user policies based on their role. Figure 1 shows the network model of the proposed method.

5.2 Role based user policy-SPHE model

The following modules are used in the application.

(i) Administrator policy

AdministratorAccess - The data owner has full access rights on EB applications. The DO can create, change and delete EB applications. The access levels assigned for DO are list, read, tagged, and write. All-access levels are set for DO so that they can have full control over the application.

(ii) Evaluator policy

The evaluator can perform computation on data. They can able to perform operations like multiplication or division on two ciphertexts, integer multiplication, or division with the encrypted data.

(iii) Customer Access Policy

AwsElasticBeanstalkReadOnly – The customers can able to view applications and data and they cannot perform any operation on data.

(iv) Amazon S3 – Client-side encryption

The patient data is stored in Amazon S3 as objects. The data stored in Amazon S3 is not encrypted by default. There are two options to encrypt data in the S3 bucket, server-side encryption, and client-side encryption. The proposed method is used for encrypting the patient data and uploads it into Amazon S3. The keys are managed on the client-side so that this method is more secure. Because the Amazon server is not able to access both data and keys. The patient encrypted data in the S3 bucket can be accessed by the application running on the EB instance. For that, when the application is created, S3 URL is specified to get access to the S3 object. Figure 2 shows the user access policy of the proposed model.

Table 1
shows the simulation parameters used for this study

User roles	Access policy	Key size	Message size	Simulation Factors
• Admin	• List	1024 bits	128 bits	• Encryption Time
• Power User	• Read			• Decryption Time
• End User	• Tagged			• Key generation time
	• Write			• Multiplication time
				• Division time
				• cipher text storage

The scheme is implemented for the key size of 1024 bits and various message sizes from 128 bits to 1024 bits with different access policies of various users. The simulation factors considered are the time for encryption, decryption, multiplication, and division operation. Table 2 and Figure 3 show the simulation result which is the average value of 50 trials for various message sizes from 128 bits to 1024 bits.

Table 2
Time taken by SPHE Algorithm

Message Size (in bits)	Time taken in ms			
	Encrypt Time	Decrypt Time	Multiplication operation	Division Operation
128	2.9	0.99	1.01	1.51
252	3.6	1.56	1.78	2.51
512	4.9	1.95	2.34	3.15
1024	5.4	2.15	2.65	3.52

The encryption time taken by the SPHE algorithm ranges from 2.9 ms to 5.4 ms for message sizes 128 bits to 1024 bits. The decryption time ranges from 0.99 ms to 2.15 ms, the multiplication operation on ciphertext is from 1.01 ms to 2.65 ms and the division operation is from 1.51 ms to 3.52 ms for the sizes of the messages 128 bits to 1024 bits.

6. Comparison Of Homomorphic Schemes

In this section, the simulation result of the proposed method is compared with other homomorphic encryption methods Paillier, Elgamal, and Benaloh. The same simulation parameters are used for compared schemes. The simulation is repeated 50 times to get a more accurate result for all schemes. To compare the performance of the proposed algorithm with other schemes which are considered for this

study, the same system parameters, a key size of 1024, and a message size of 128 bits are used for all schemes. Table 3 shows the average values of all simulation factors for 50 trials.

The encryption time is measured for the input message size of 128 bits and key size of 1024 bits. It has been observed that Paillier encryption takes maximum time for encryption among all schemes because of more number of modular operations during encryption function. The Benaloh scheme takes minimum time. The role-based SPHE has a better performance than Elgamal and Paillier schemes. Figure 4 shows the comparison results of encryption time for all schemes. The results show that SPHE has almost 3 times better than Elgamal and nearly 6 times better than Paillier because of less number of modulo operations.

The decryption time for the proposed scheme is that Paillier encryption takes more decryption time because of the complicated decryption function. Figure 5 shows the comparison results of decryption time. The proposed method has taken minimum time for decryption because of one modular exponentiation operation to decrypt the ciphertext.

Table 3
The comparison result of the simulation factors with key size 1024 bits, message size 128 bits

	Key Generation Time	Encryption Time	Decryption Time	Multiplication Operation	Division Operation	Storage of Cipher Text
SPHE	476	2.9	0.99	1.01	1.51	390
Paillier	332	19.1	31.98	21.33	23.33	511
Elgamal	2517	9.14	8.6	7.89	10.89	519
Benaloh	3126	1.91	28.23	20.91	21.91	513

Figure 6 shows the comparison results of key generation time. It is observed that the Paillier scheme takes less time and Benaloh takes more time for key generation. The proposed scheme has a better performance than Elgamal and Benaloh. It has five times better than Elgamal and almost 6 times better than Benaloh. Figure 7 shows that the comparison on multiplication operation of ciphertext for all schemes. From the result, It is observed that the Paillier scheme takes more time for multiplication on two ciphertexts because of more number of modular operations. The proposed scheme has a better performance among all methods because of the minimum number of modular operations.

Figure 8 shows that the comparison of division operation on ciphertext for all methods. The result shows that the proposed scheme has taken less time for division operation. Figure 9 shows the comparison results for storage of ciphertext generated by all schemes. The graph shows that the proposed method takes minimum storage because of its simple encryption operation. It is observed that the proposed scheme takes almost 24% less storage than other compared schemes.

7. Conclusion

In this paper, a model for securing medical data is developed which is the combination of secure partially homomorphic encryption with role-based user policy. The proposed model is developed based on access level policy for different user roles in the AWS cloud platform. The Amazon Elastic Beanstalk deploys the application and the Amazon S3 bucket stores patients' medical records. The medical details are encrypted by the proposed SPHE algorithm and uploaded in the S3 bucket. The users for this model are created by Amazon IAM service and different policies are assigned for the users based on their roles. The proposed method is more secure and computation can be performed on the ciphertext by the homomorphic property. The proposed algorithm performance is studied by comparing with other encryption methods Elgamal, Paillier, and Benaloh. The result shows that the new scheme has better performance than those other encryption methods.

Declarations

Compliance with Ethical Standards statements

I. Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

II. Funding details (In case of Funding)

No funds, grants, or other support was received.

III. Conflicts of interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

IV. Informed consent

None

V. Authorship contributions

All authors contributed to the study conception and design. Data collection, analysis and implementation were performed by M.D.Boomija, S.V. Kasmir Raja. All authors read and approved the final manuscript.

References

1. Zhang R, Liu L, Xue R (2014) Role-based and time-bound access and management of EHR data," *Secur. Commun. Netw.*, vol. 7, no. 6, pp. 994_1015, Jun.

2. Bertino E, Ghinita G, Kamra A (2011) Access control for databases: Concepts and systems, vol 8. Now Publishers Inc
3. Zhou L, Varadharajan V, Hitchens M (2015) Trust Enhanced Cryptographic Role-Based Access Control for Secure Cloud Data Storage. IEEE Transactions on Information Forensics and Security 10, 11 (Nov 2015), 2381–2395. https://doi.org/10.1109/TIFS.2015.2455952
4. Liu W, Liu X, Liu J, Wu Q, Zhang J, Li Y (2015) Auditing and revocation enabled role-based access control over outsourced private EHRs," in *Proc. 17th Int. Conf. High Perform. Comput. Commun.*, pp. 336_341, Aug.
5. Tasatanattakool P, Techapanupreeda C (2017) User authentication algorithm with role-based access control for electronic health systems to prevent abuse of patient privacy," in *Proc. 3rd IEEE Int. Conf. Comput. Commun. (ICCC)*, Dec. pp. 1019_1024
6. Cui H, Deng RH, Li Y (Feb. 2018) "Attribute-based cloud storage with secure provenance over encrypted data,". Future Gener Comput Syst 79(2):461–472
7. Huang C, Yan K, Wei S, Zhang G, Lee DH (2017) "Efficient anonymous attribute-based encryption with access policy hidden for cloud computing," in Proc. Int. Conf. Progr. Informat. Comput. (PIC), Dec. pp. 266–270
8. Chi P-W, Lei C-L (2018) "Audit-free cloud storage via deniable attribute based encryption," IEEE Trans. Cloud Comput., vol. 6, no. 2, pp. 414–427, Apr./Jun.
9. Kuhn DR, Coyne EJ, Weil TR (2010) "Adding attributes to role-based access control," IEEE Comput., vol. 43, no. 6, pp. 79–81, Jun.
10. Khan MFF, Sakamura K (2015) "Fine-grained access control to medical records in digital healthcare enterprises," in Proc. Int. Symp. Netw. Comput. Commun. (ISNCC), May pp. 1–6
11. Xu K, Xu Y, Hong J, Li W, Yue H, Wei DS, Hong P (2017) "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," IEEE Trans. Inf. Forensics Security, vol. 12, no. 4, pp. 953–967, Apr.
12. Sandhu R, Ferraiolo D, Kuhn R (2000) "model for role-based access control: Towards a unified standard," in Proc. ACM Workshop Role-Based Access Control, Jul. pp. 1–11
13. Pussewalage HSG, Oleshchuk V (2016) "A patient-centric attribute based access control scheme for secure sharing of personal health records using cloud computing," in Proc. 2nd Int. Conf. Collaboration Internet Comput. (CIC), Nov. pp. 46–53
14. Bouchti AE, Bahsani S, Nahhal T (2016) Encryption as a service for data healthcare cloud security," in *Proc. 5th Int. Conf. Future Gener. Commun. Technol. (FGCT)*, Aug. pp. 48_54
15. Guo L, Zhang C, Sun J, Fang Y (2012) pp. 224_233. [80] M. Naehrig, K. Lauter, and V. Vaikuntanathan, Can homomorphic encryption be practical?" in *Proc. 3rd ACM Workshop Cloud Comput. Secur. Workshop*, Oct. 2011, pp. 113_124
16. Barni M, Failla P, Lazzeretti R, Sadeghi A (2011) and T. Schneider,Privacy-preserving ECG classi_cation with branching programs and neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 452_468, Jun.

17. Carpov S, Nguyen TH, Sirdey R, Constantino G, Martinelli F (2016) Practical privacy-preserving medical diagnosis using homomorphic encryption," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jul. pp. 593_599
18. Gentry C, Halevi S (2011) Implementing gentry's fully-homomorphic encryption scheme," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, pp. 129_148
19. Halevi S, Polyakov Y, Shoup V (2019) An improved RNS variant of the BFV homomorphic encryption scheme. In: Matsui M (ed) Topics in Cryptology –. CT-RSA Springer, pp 83–105
20. Nagarajan G, Minu RI, Jayanthiladevi A (2019) Brain computer interface for smart hardware device. International Journal of RF Technologies 10(3–4):131–139
21. Min Zhao E, Geng Y (2019) Homomorphic Encryption Technology for Cloud Computing. Elsevier, procedia computer science 154:73–83
22. Nagarajan G, Minu RI (2016) Multimodal fuzzy ontology creation and knowledge information retrieval. In Proceedings of the International Conference on Soft Computing Systems (pp. 697-706). Springer, New Delhi
23. Sethi K, Chopra A, Bera P, Tripathy BK (2017) Integration of role based access control with homomorphic cryptosystem for secure and controlled access of data in cloud. Proceedings of the 10th International Conference on Security of Information and Networks - SIN '17. doi:10.1145/3136825.3136902

Figures

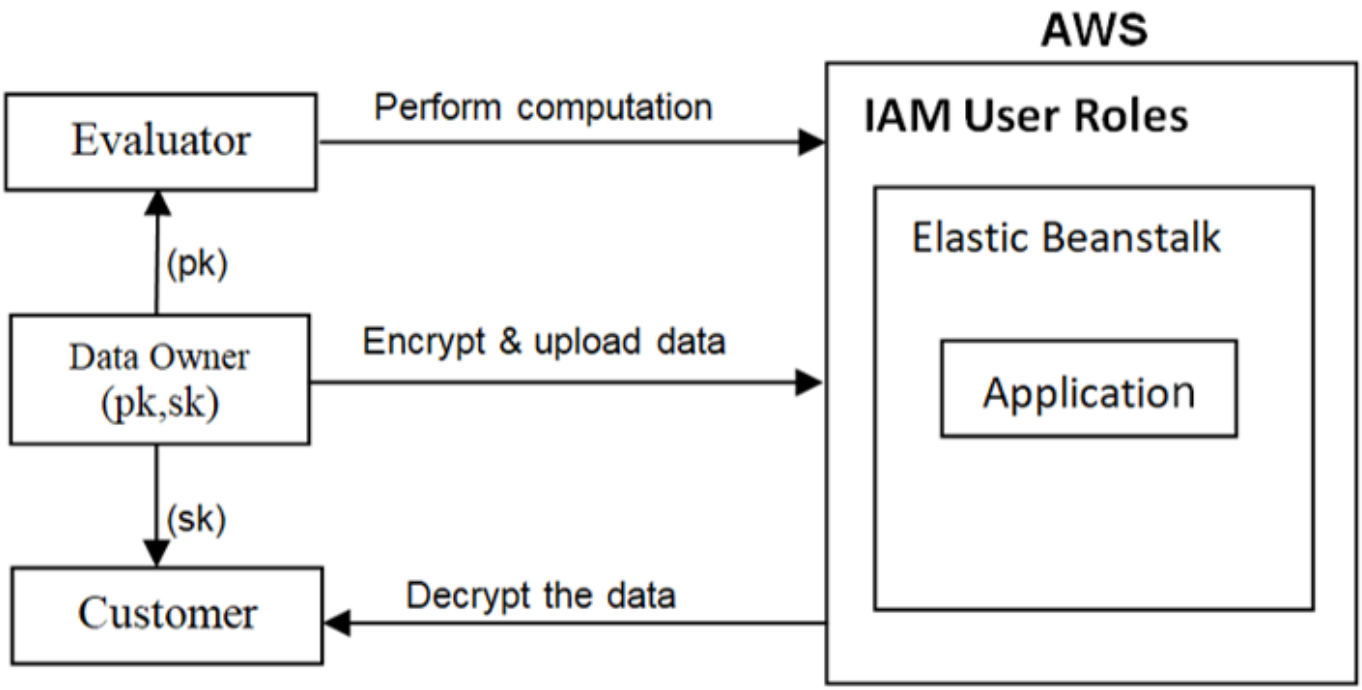

Figure 1

Network Model of Role based user policy-SPHE

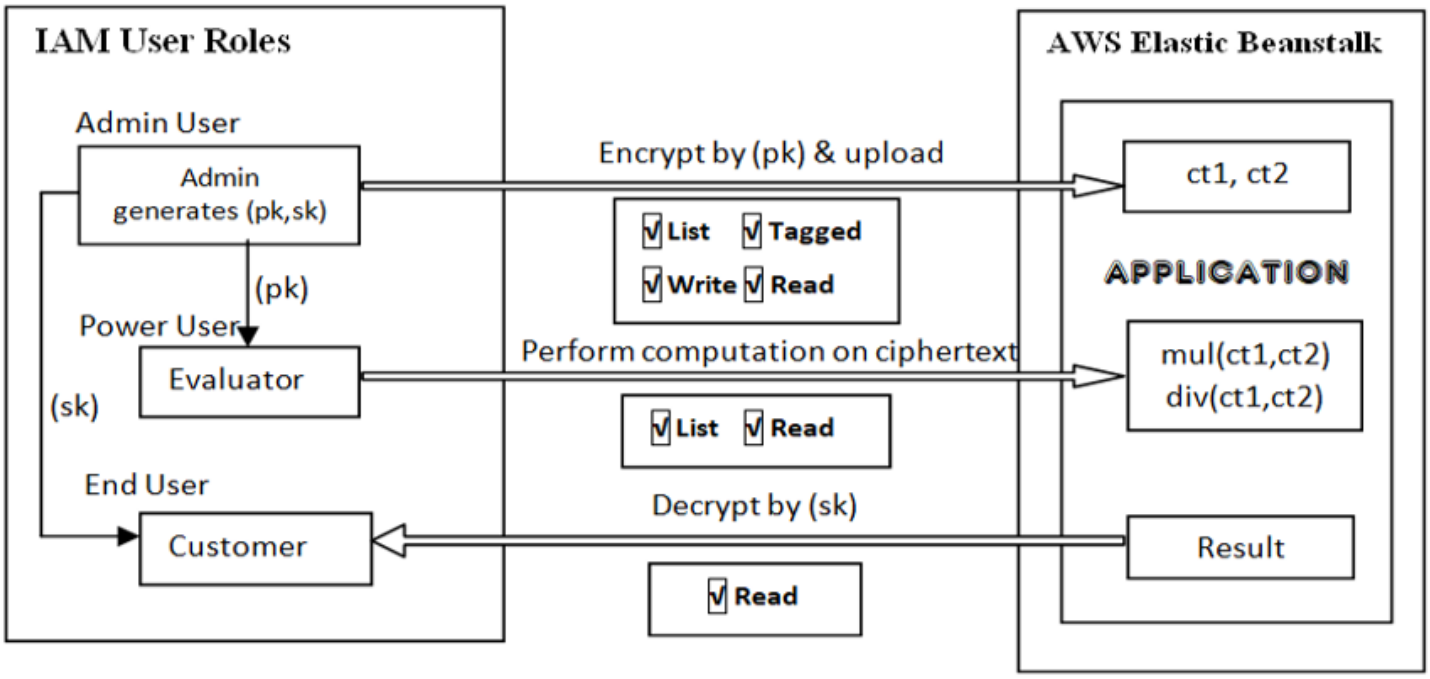

Figure 2

User access policy of the proposed model

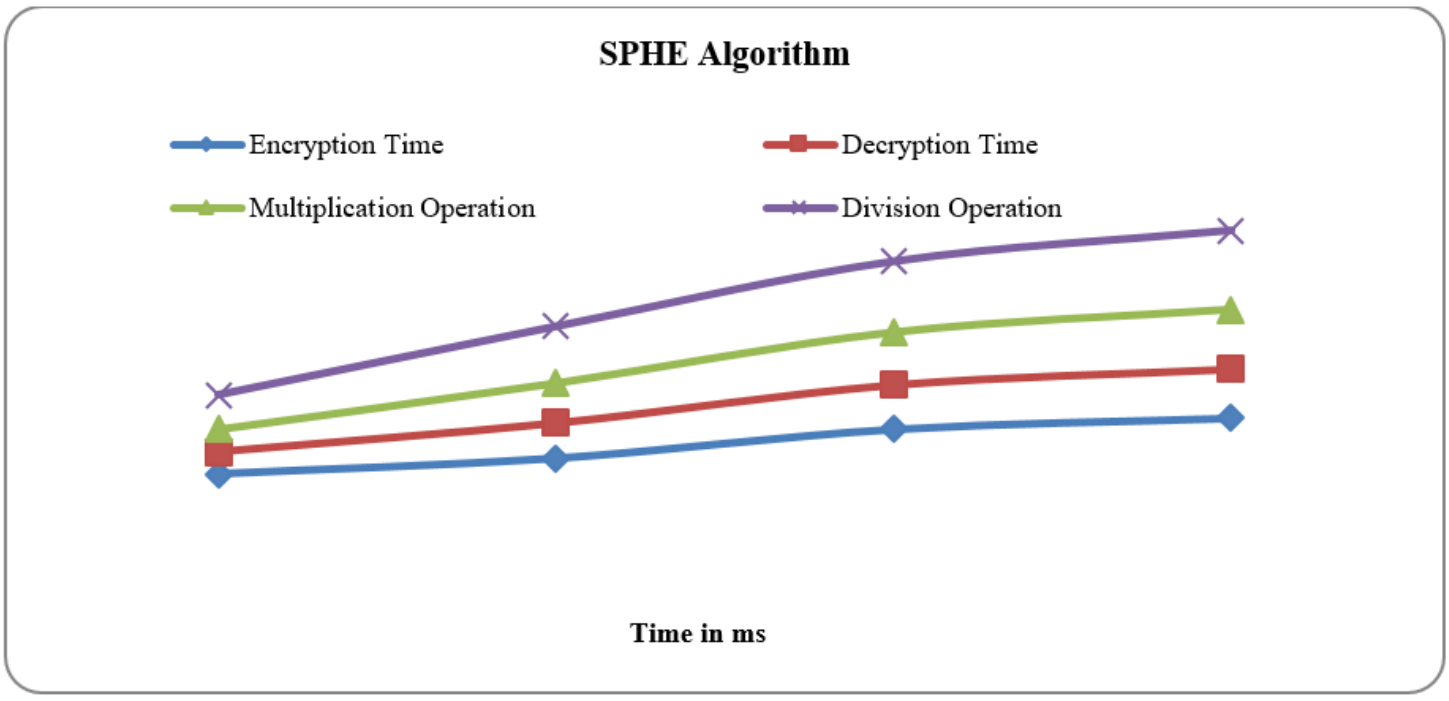

Figure 3

Time taken by SPHE Algorithm

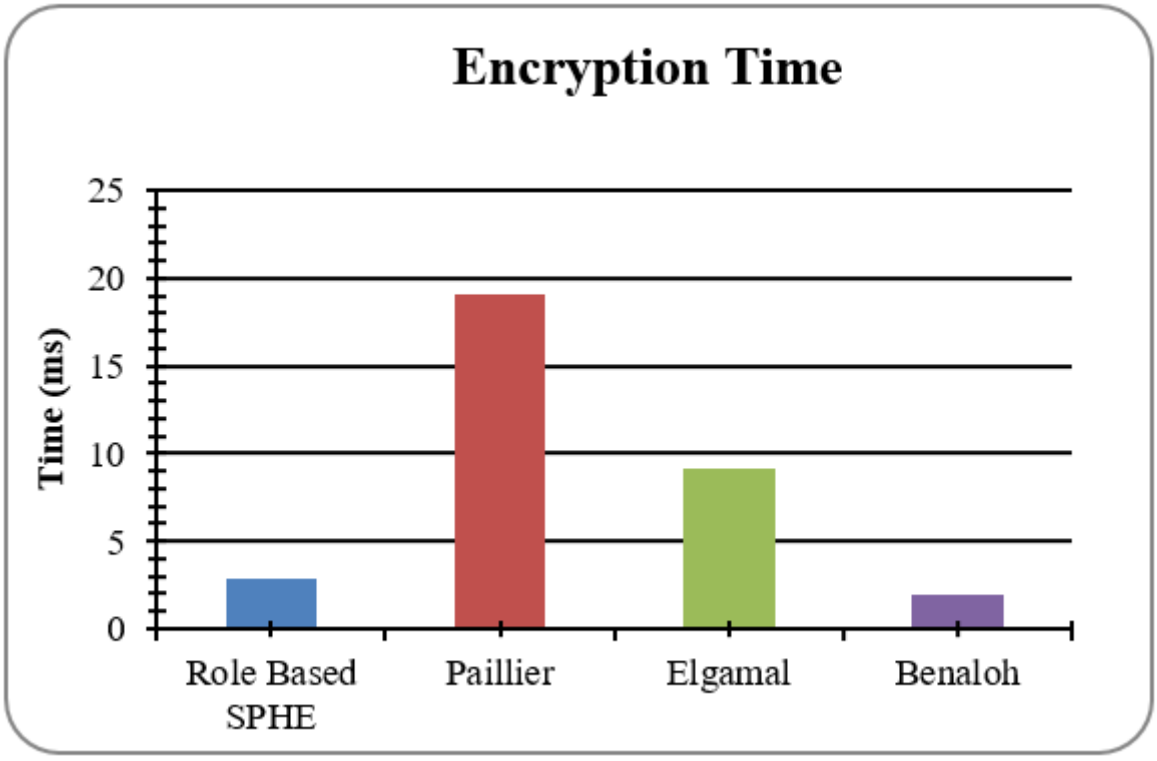

Figure 4

Comparison results of Encryption Time

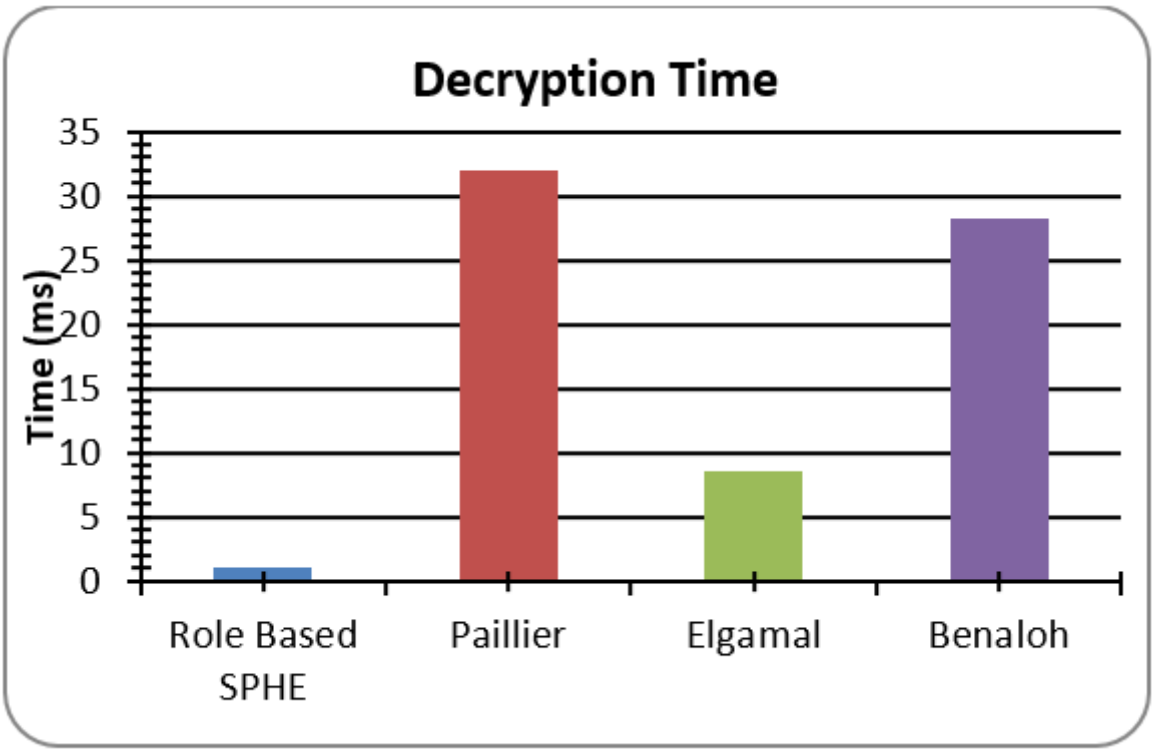

Figure 5

Comparison results of Decryption Time

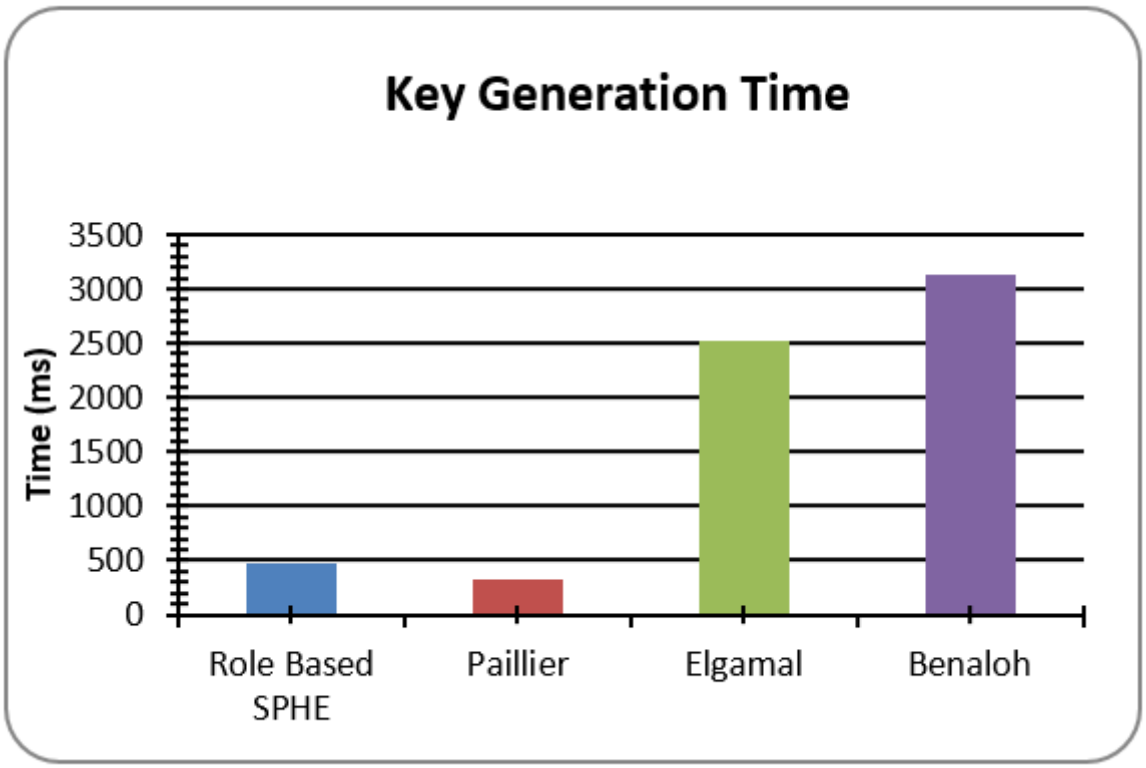

Figure 6

Comparison results of Key Generation Time

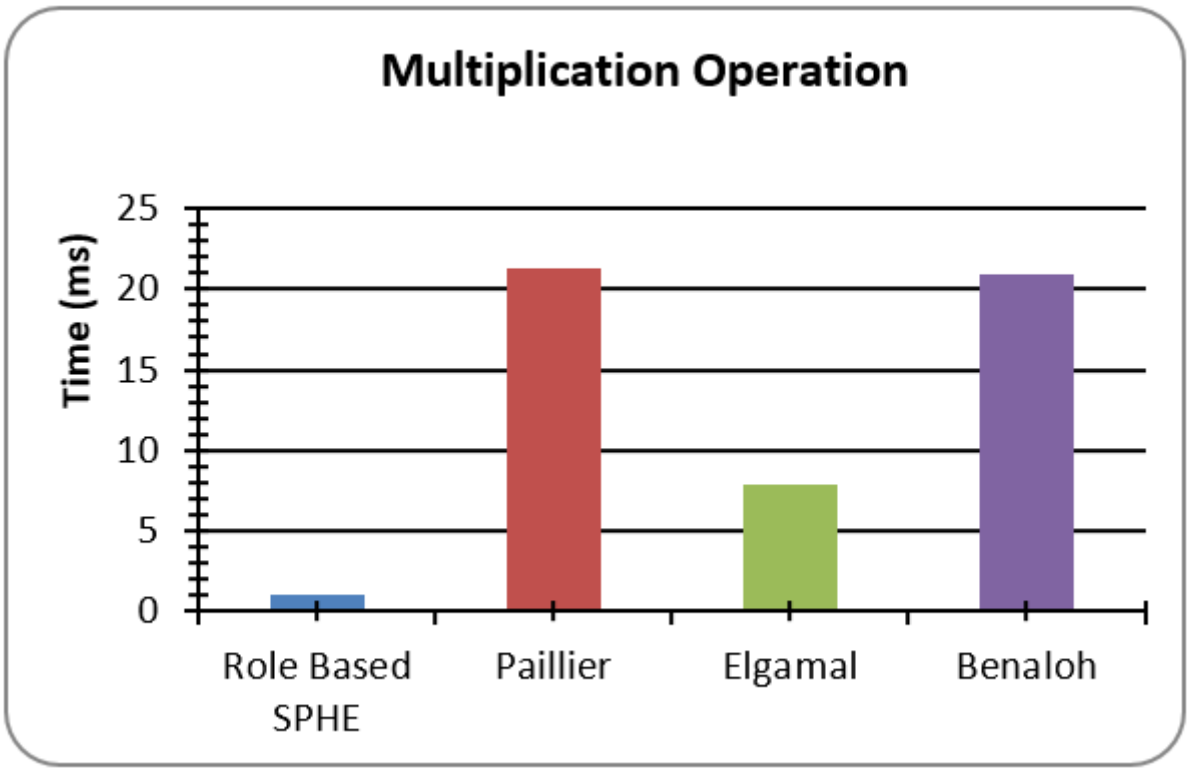

Figure 7

Comparison results of Multiplication time

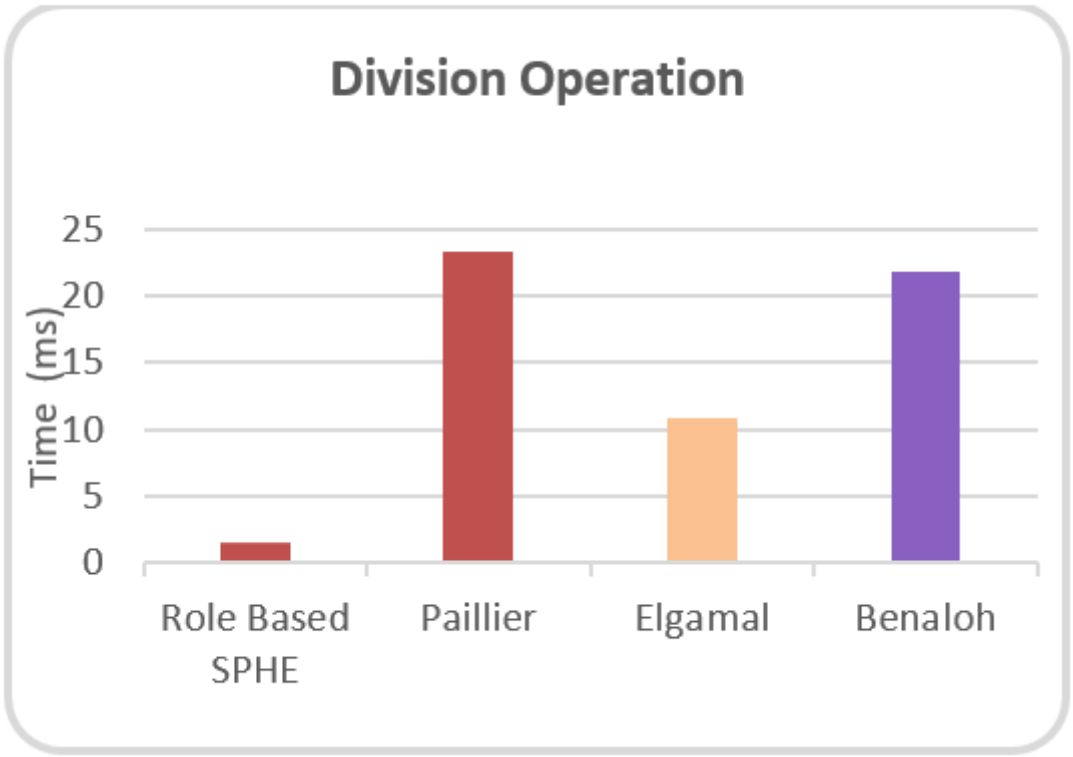

Figure 8

Comparison results of Division Time

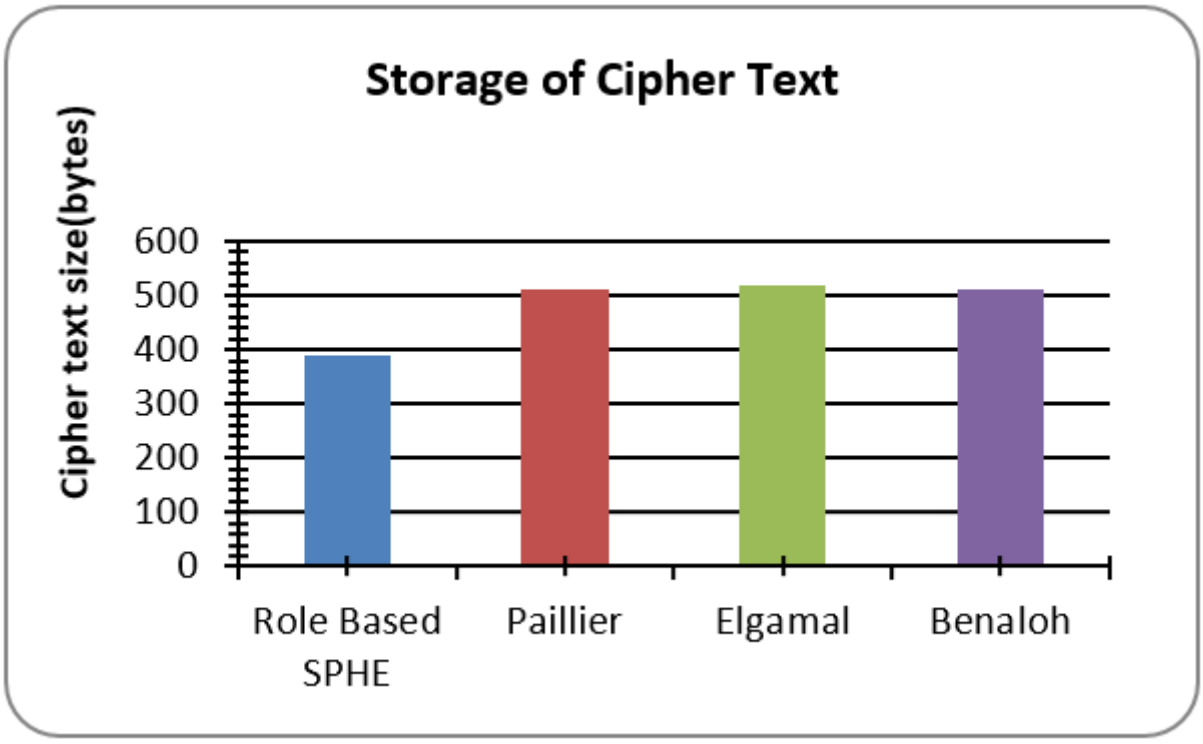

Figure 9

Comparison results of storage size