Wireless Personal Communications
https://doi.org/10.1007/s11277-019-06450-7

ECC Based Secure Sharing of Healthcare Data in the Health Cloud Environment

V. Sri Vigna Hema[1] · **Ramesh Kesavan**[1]

Abstract
In today's world, Cloud computing based health cloud is an effective platform for enhancing e-health productivity, i.e., healthcare professionals can provide medical service by accessing health information on any device in any location at any time. The major challenge faced by a healthcare organization is secure sharing of healthcare data in health cloud among general practitioner, medical services, and insurance firms. In order to overcome this issue, the security of sensitive healthcare data is accomplished with encryption mechanisms. In this research work, the medical data files are encrypted using elliptic curve cryptography algorithm to possess confidentiality and integrity in the healthcare data while sharing data via health cloud. The experimental results exhibits better performance in terms of key generation time, turn-around time, file encryption time, file decryption time, file upload time, file download time, file uploading speed, as well as security overhead. Furthermore, it also securely transmits the healthcare data over health cloud by defending against insider threats.

Keywords Cloud computing · Elliptic curve cryptography (ECC) · Healthcare · Health cloud · Security

1 Introduction

Communication technology has developed rapidly in recent decades. The incorporation of wireless communication technology and healthcare has been widespread than in the past. Now-a-days, several healthcare related schemes like patient health monitoring and diagnosis [1, 2], telemedicine method [3, 4], e-health system [5], m-health system [6, 7], has technologically advanced. With the emergence of cloud computing, it has attracted most healthcare service providers to shift their healthcare service as well as storage into the health cloud. Health cloud is a cloud computing service used for storing, analyzing, sharing, maintaining and backing up personal health information. Health cloud services

✉ V. Sri Vigna Hema
vignahema@auttvl.ac.in

Ramesh Kesavan
ramesh.k@auttvl.ac.in

[1] Department of Computer Applications, Anna University, Tirunelveli, Tamil Nadu, India

Published online: 08 May 2019

are capable of storing significantly more data than an on-site physical server, particularly, when it comes to the large image files. Though cloud-based health care systems are advantageous, many physician and healthcare organizations are reluctant to use health cloud services for fear of suffering in data breach. And also, due to the sensitive nature of the data being stored and accessed, many health care organizations are opting to avoid public cloud services and implementing a private cloud service instead.

The electronic health records are essential to be securely shared over networks to retain data integrity and privacy of the patients. The challenges of employing cloud database for storing health information are pointed out in [8, 9]. To defend the electronic health records from being revealed in their communication, a cryptosystem is frequently required. Conventional methodologies such as Rivest-Shamir-Adleman (RSA)-based system [10], modified RSA digital signature scheme [11], secure user data via encryption methods [12] have been effectively applied in various applications. Moreover, these schemes commonly restrict the opportunity of applying them to portable devices due to the larger keys as well as complex computation. Elliptic Curve Cryptography (ECC) has progressively been recognized and applied in cryptosystems for better solution than other traditional methods. ECC provide efficient performance and highly scalable environment by reducing the key complexity with smaller keys. However, for storing and sharing data in the health cloud, it is necessary for each cloud customer to encrypt their data before uploading to the cloud servers. Moreover, this research work contributes the health cloud with the ECC implementation to improve and enhance healthcare services in terms of data confidentiality, data security, and data integrity.

The proposed methodology has three major entities as follows: (1) the Cloud Users (CUs), (2) the Trusted Third Party-Cryptographic Server (TTP-CS), and the Health Cloud (HC). The owner of the data (data owner, who is one among the CUs) submits the healthcare data, the list of CUs, and the factors need to generate an Access Control List (ACL) to the TTP-CS. The responsibility of TTP-CS is to manage the cryptographic process, namely, key management, encryption, decryption and access control. To ensure integrity, the TTP-CS generates private key and public key, where the private key is with the CU side and the public key is in the TTP-CS side. Consecutively, the inventive key is removed via the concept of secure overwriting [13]. On behalf of the CU, the encrypted files are uploaded subsequently to the HC for storage. The CU who wants to access the health data files will send a download request to the TTP-CS. After verifying the authentication of the download request from the CU, the TTP-CS either denied the access or it will fetches the encrypted data files from the HC and corresponding decryption key from the data owner. Besides, the conventional cloud computing, the proposed methodology can also be employed with the Mobile Cloud Computing (MCC) model. This is because of the circumstance that compute-intensive processes are accomplished by the TTP-CS.

The major contributions of the proposed methodology are stated as follows:

- It guarantees the data confidentiality on the health cloud by using stronger encryption method.
- Among the cloud users, the data is shared securely over the health cloud using ECC mechanism that is highly scalable with fewer resources and faster uploading time.
- It offers high security on data against the insider threats.
- Correspondingly, it provides data integrity and verification by employing hash function.

The rest of this paper is systematized as follows. Section 2 overviews the relevant literature work in the field. In Sect. 3, the details of the proposed methodology are described

with an overall system architecture. Section 4 explains the performance evaluation of the proposed system, and finally, Sect. 5 concludes the paper.

2 Related Works

In this section, recent advancement in data sharing methodologies in cloud computing environment has been briefly discussed. Xu et al. [14] presented a novel Certificate-Less Proxy Re-Encryption (CL-PRE) technique for sharing the data with the public cloud in a flexible and secure way. Deprived of fully trusting the infrastructure of cloud, this technique [14] conserves access control over data owner. The data owner uses symmetric key to encrypt the data before storing into the cloud. Further, proxy re-encryption keys are generated by the data owner, which can be decrypted using the private key of the user. Instead of certificates, the identity of the user is employed for creating the public–private key pair. This technique is computationally intensive as the PRE is based on bilinear pairing and Bilinear Diffie–Hellman. While comparing with the standard finite field operations, the computational cost is high in bilinear pairing.

In order to decrease the bilinear pairing's computational overhead, a medicated CL Public Key Encryption (mCL-PKE) approach [15] is proposed by Seo et al. This approach securely shared the sensitive data in the public cloud without applying pairing operations. Herein, the cloud is not only employed as a secure storage but also as the center for key generation. The cloud produces the public–private key pairs but the public keys are transmitted only to the contributing users. Partial decryption as well as key management is also managed by the cloud. Therefore, user revocation is at ease to handle and free from key escrow difficulty. The decryption accomplishes two-folds, which moderates the benefit of non-pairing operations.

Itani et al. [16] introduced a lightweight framework to ensure the data integrity of the mobile user along with the concept of incremental cryptography. Most works of the integrity verification are offloaded on cloud in order to make the framework appropriate for the mobile device. Khan et al. [17] defined the PRE scheme, which provide data security of the mobile user. As in [16], a novel incremental version of PRE scheme is described in [17] for enhancing the operation of file modification while offering the confidentiality services. In incremental cryptography concept, the blocks are incrementally encrypted by dividing the data into blocks. This scheme applies bilinear pairing along with El-Gamal cryptosystem in order to share the sensitive information in the cloud. It also employs a proxy (i.e. trusted third party), which accomplishes the operations of compute-intensive. Furthermore, the computational complications of bilinear pairing still occur in the scheme.

Chen and Tzeng [18] presented a shared key derivation scheme to share the data among a group, which uses binary tree for the key computation. As the rekeying technique is greatly used in this scheme, the cost of computation is high. However, certain operations of this scheme require centralized mediations and it is not designed for public clouds. Similarly, RSA-based methodology is introduced in [19]. Moreover, this methodology is susceptible to collusion attacks. Instead of employing bilinear pairing, bilinear Diffie–Hellman, and El-Gamal cryptosystem, Ali et al. [20] proposed a secure method to exchange the data within a group. Rather than re-encryption, this lightweight method is based on symmetric cryptography to offer access control over malicious attack. Since both encryption and decryption functions are performed at the cryptographic server, the trust level is limited in the system.

Zhang et al. [21] introduced a new Personal Health Record (PHR) sharing system architecture using Anonymous Attribute-based Encryption (AABE). Further, the architecture is extended efficiently based on Hierarchical AABE (HAABE). Both the standard models accomplish compact security. Another key benefit of this scheme is smaller private and public keys. Rao [22] described a novel Ciphertext-Policy Attribute-Based Signcryption (CP-ABSC) method, which is demonstrated to be secure against malicious attacks. This method accomplishes crucial security objectives of Attribute-based Encryption (ABE) as well as Attribute-Based Signature (ABS) systems like privacy of signcryptor, unforgeability, and data confidentiality. While comparing to the other schemes, it uses only small ciphertext and fewer pairing operations.

Based on the inference from the survey on the related works, ECC-based secure sharing of healthcare data in the health cloud is proposed. ECC algorithm is identified since the encryption is stronger as well as it is relatively compute-intensive when compared with other methods. Besides, this scheme prohibits insider threats in order to provide high security on the healthcare data.

3 ECC Based Secure Sharing of Healthcare Data in the Health Cloud Environment

In this section, the design of proposed methodology for secure sharing of the healthcare data in the health cloud environment is presented.

3.1 Architecture Overview

An overall architecture (Fig. 1) of ECC-based healthcare system comprises the following entities:

Health Cloud (HC): The HC provides cloud services to the user for storing, maintaining, and backing up healthcare information. The server in the HC maintains the healthcare information and supports all cloud services. The data on the health cloud required to be secured against several threats. The healthcare data confidentiality is ensured by keeping encrypted data over the health cloud.

Trusted Third Party-Cryptographic Server (TTP-CS): TTP-CS is the trusted entity available outside the cloud, owned by a third party to perform the cryptographic process. This proposed mechanism adapts ECC algorithm to ensure security of the sensitive healthcare data. It is liable for security processes like data confidentiality, data integrity, key management, encryption, and decryption to provide secure sharing of healthcare data.

Cloud Users (CUs): The cloud users (i.e., researcher, analyst, physician, and more) are the customers of the health cloud. The CUs are necessary to be registered with the TTP-CS for accomplishing the security services. One of the CU will be the file owner for each data file (say, data owner), while the others will be the cloud data consumers.

3.2 System Model

This model upholds the asymmetric or public key cryptographic system for secure sharing each medical data file to/from the health cloud. The DO submits the healthcare data file along with the list of CUs to the TTP-CS. TTP-CS generates the public key, K_{Pb} and a

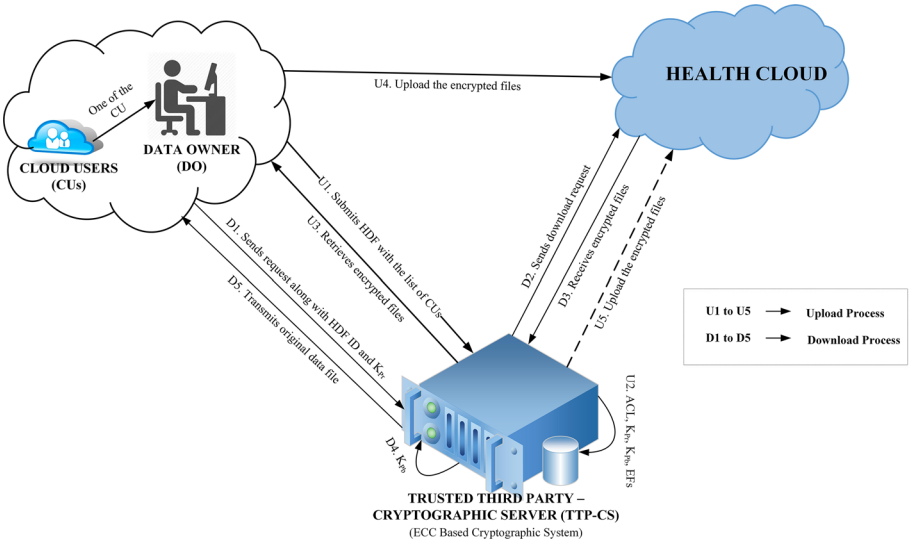

Fig. 1 An overall architecture of ECC-based Secure healthcare data sharing in the health cloud

private key, K_{Pr}, each of length 256 bits, which is a random one for all medical data files. According to the necessity of the asymmetric key algorithm, the length of the key can be modified. K_{Pb} and K_{Pr} are generated by hashing a random number *RN* using a SHA-256 hash function. It is further employed in the encryption and decryption process for securing the healthcare data. The entire key is not controlled by any of the existing entities after encryption or decryption process. The TTP-CS creates different K_{Pb} for each of the CUs, which can be distributed freely and employed for encryption process. Whereas, K_{Pr} is kept secret by every CU and used for decryption process. The following are the cryptographic operations to contribute the proposed methodology accomplish for security goals.

3.2.1 Uploading a File to the HC

Whenever a CU would like to upload a medical/healthcare data to the health cloud, the DO sends the request for encryption to the TTP-CS. The healthcare data file (HDF), and the cloud user's list (CUL) are accompanied to mention access privilege to the user. The CUL encompasses the access rights for every CUs and may have access like Read-only and/or Read–Write to the HDF. The Access Control List (ACL) is created using CUL for the healthcare data by the TTP-CS. While the HDFs are to be share among a new center, the data owner will update the new CUL to the TTP-CS. Otherwise, it transfers only the center ID of the surviving center. The TTP-CS creates the ACL and maintain it for each HDF after receiving the encryption requisition. The ACL consists of file information like the file ID, the file size, the file owner ID, the information about file sharing, and other metadata. After creating the ACL, the TTP-CS generates private key K_{Pr} and public key K_{Pb} for every CUs. Then, the HDF gets encrypted using ECC encryption mechanism. The resultant of this process is the encrypted files, E_{f_1} and E_{f_2}. In this, E_{f_1} is the product of randomly selected number k within the range of 1 to (n−1) and the point on the elliptic curve Pt_c. E_{f_2} is the summation of HDF along with the product of k and the public key, K_{Pb}. For further process, the K_{Pb} is incorporated into the ACL

for each CU. The hash-based message authentication code (HMAC) signature and its key is computed and maintained by the TTP-CS on every encrypted file for protecting the file integrity. The center ID, the encrypted files (E_{f_1} and E_{f_2}), and the owner's K_{Pr} are forward to the requesting owner of that data. Whereas, the remaining CUs gets only the center ID and K_{Pr} through a Secure Socket Layer (SSL). Eventually, K_{Pr} and K_{Pb} are detached from the TTP-CS after the encryption process through a secure overwriting [13]. Once the encrypted files (E_{f_1} and E_{f_2}) are received, it either directly uploads to the HC by the DO or by the TTP-CS (on behalf of CU).

Algorithm 1: ECC Encryption Process

Input : The HDF, the ACL, the Hash Function (HF): 256-bit

Output : Encrypted files

Compute:

For *each CU 'a' in the ACL,* ***do***

Select random number K_{Pr} within the range of 1 to (n-1);

Randomly select 'k' from 1 to (n-1); // n is the maximum limit

// Key generation process

$K_{Pb} = K_{Pr} * Pt_c$; // Pt_c– Point on the elliptic curve.

$E_{f_1} = k * Pt_c$;

$E_{f_2} = HDF + k * K_{Pb}$;

Add K_{Pr} for CU 'a' in the ACL;

Send K_{Pr} for CU 'a';

End For;

Remove K_{Pr};

Remove K_{Pb};

Return *E_{f_1} and E_{f_2} to the HC directly by the TTP-CS or by the DO;*

Algorithm 1 provides the process for key generation and encryption using ECC algorithm at the TTP-CS level. Once the center gets initiated or the file is submitted for encryption, the process for generating the key is performed. There are two possible ways to handle the file uploading process: (1) after receiving the encrypted files, E_{f_1} and E_{f_2}, the DO can be directly uploaded to the HC as mentioned before, or (2) On behalf of the CU, the TTP-CS (who has the authority delegation) can upload the file to the HC. A simple instance of uploading a medical data file to the HC is explained step by step as follows (as in Fig. 1):

U1 Physician submits the healthcare data file along with the list of users to the TTP-CS

U2 TTP-CS generates the ACL, private key, and public keys for the physician. The file is then encrypted using ECC encryption mechanism

U3 The physician retrieves the center ID, the encrypted files, and the physician's private key from the TTP-CS

U4 The physician upload the encrypted files directly to the health cloud

U5 In special case, on behalf of the physician, TTP-CS uploads the encrypted files to the health cloud

3.2.2 Downloading a File from the HC

For downloading a file from the HC, either the CU must send a request with the necessary authentication credentials to the TTP-CS or the DO downloads the encrypted files from the HC and further, it sends the request for decryption to the TTP-CS. The HC authenticates the CU authorization via a locally uphold ACL. From the ACL, the TTP-CS retrieves K_{Pb}. If the K_{Pb} does not exist in the ACL, then the access denied message will be forwarded to the requesting CU. None of the CUs can employ other CU's K_{Pr}, because every CUs has a distinct K_{Pb}. Consequently, once the integrity of the file is verified by the TTP-CS, the decryption process is progressed. If the TTP-CS receives the valid K_{Pr}, the end result will be either an effective decryption process or a failure one. The HDF is forward to the corresponding CU via a SSL channel after successful ECC decryption process. Eventually, K_{Pr} and K_{Pb} are removed from the TTP-CS through the secure overwriting concept. Algorithm 2 overviews the process of decryption technique.

Algorithm 2: ECC Decryption Process

Input : Encrypted files E_{f_1} and E_{f_2}, ACL
Output : HDF

Compute:
Obtain K_{Pr} from the requesting CU;
Obtain E_{f_1} and E_{f_2} from the requesting CU or Download from the HC;
Retrieve K_{Pb} from the ACL;
    If** the K_{Pb} does not exist in the ACL, **then
        ***Return** the access denied message to the CU;*
    Else
        $HDF = E_{f_2} - K_{Pr} * E_{f_1}$;
        Send HDF to the requesting CU;
    End If;
Remove K_{Pr};
Remove K_{Pb};

On behalf of the CU, the file downloading process can be also done by the TTP-CS as in the case of the file uploading process. As mentioned before, the request for decryption is directed to the TTP-CS together with the authentication credentials. For the stated file, the TTP-CS forwards this request to the HC after verifying the CU authenticity. The HC further sends the encrypted files to the TTP-CS and the remaining process is the similar as mentioned above. A simple illustration of downloading a medical data file from the HC is described as follows (as in the Fig. 1):

D1 The CU (may be analyst, physician, or researcher) send requests along with the file ID and its private key to the TTP-CS.
D2 TTP-CS send download request to the health cloud after the verification of the CU authorization via ACL.
D3 TTP-CS receives the encrypted files from the health cloud.
D4 TTP-CS retrieves the public key from the ACL.
D5 The original data file is send to the corresponding CU.

3.2.3 File Restore

File restoring has a same process to that of file uploading process, but, the major dissimilarity is that, all the functionalities regarding ACL creation as well as key generation are not carried out while restoring a file. If any modifications has been made by the CUs (who has already downloaded the file), then the CU forwards a restore request to the TTP-CS. The TTP-CS authenticates whether the CU has the WRITE file access or not. If there exists a valid request of file restore, then the keys are computed by the TTP-CS. Furthermore, the file is encrypted and then the HMAC computations are performed. The encrypted files are sent back to the CU or uploaded to the HC. Finally, the K_{Pr} and K_{Pb} are neglected. Figure 2 shows the process of restoring a file.

The proposed model provides the following features to the healthcare data:

- Data confidentiality
- Secure sharing of healthcare data among the center
- Protect healthcare data from insider threats by avoiding unauthorized access within the center
- Data integrity

4 Performance Evaluation

4.1 Experimental Setup

The proposed methodology is implemented on Windows 10 system of type 64-bit OS with an Intel Core i5-6200U CPU @ 2.40 GHz and 8.00 GB RAM. As discussed in the system model, it has three major entities, i.e., the HC, the TTP-CS, and the CUs. A Java Pairing-Based Cryptography library (JPBC) v.2.0.0 is employed in this system for communication between the entities. It delivers functions for implementing elliptic curve as well as pairing operations. Using Java libraries (java.security and java.security.spec), the communication among the entities is achieved. Further, the communication is secured via the SSL. The system has been executed using Cloudsim toolkit and the performance of the system is evaluated by the following parameters: key generation time, file encryption time, file

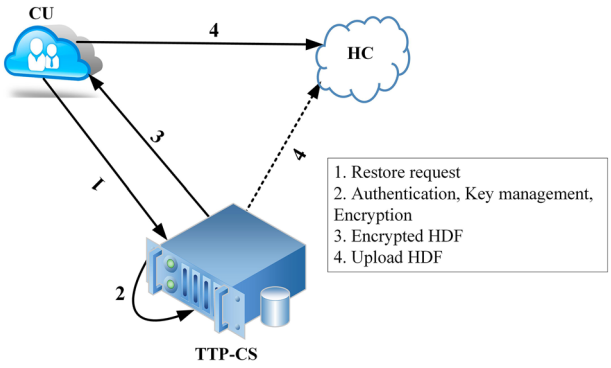

Fig. 2 Process of file restore

decryption time, file upload time, file download time, file uploading speed, and security overhead.

4.2 Experimental Results

4.2.1 Time Consumption for Key Generation

The time taken to generate the keys in cryptography is termed as the key generation time. The generated key is employed further for both encryption and decryption process. The key generation time in this system is calculated for set of CUs from 10 users to 100 users at the time interval of 10 units. Figure 3 shows that the key generation time rises with the rise of the CUs as expected. The analysis on the time consumption on key generation reveals that the rise in time consumption is not equivalently proportional to the growth of CUs. With the rise of CUs from 10 users to 100 users, the time consumption for key generation differs only by 0.003 s.

Table 1 gives a detailed comparison for key generation using various methods, namely, CL-PRE scheme [14], certificateless encryption [15], PRE scheme [17], and AES method [20]. The observation reveals that the ECC is more significant than all other compare methods.

4.2.2 File Encryption and Decryption Time

In this ECC-based system, the file encryption time is defined as the time taken by the TTP-CS to encrypt a file as per the requisition of encryption request by the CU. The file encryption time, the file decryption time, and the key computation time are considered to estimate the computational overhead in the file upload and download process. Both encryption and decryption times are estimated with different file sizes from 0.1 to 500 MB. The time taken for the encryption of files with varying size is visualized in the Fig. 4. The plot for encryption time shows a constant rise in time consumption with the increase in file size. It also exhibit that file size is independent against the key computation time which remains stable with minor variation. It is notable that the total time for encryption key computation ranges between 0.010 and 0.0185 s.

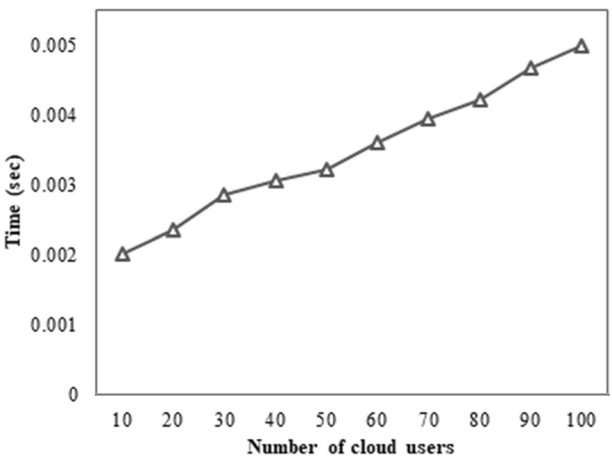

Fig. 3 Time consumption for generating key

Table 1 Observation of time consumption for key generation

Number of users	Methodologies (time in seconds)				
	CL-PRE [14]	Certificateless encryption [15]	PRE [17]	AES [20]	ECC
10	1.494	1.594	1.534	0.004	0.00212
20	1.598	1.741	1.606	0.00425	0.00235
30	1.673	2.321	1.684	0.00476	0.00286
40	1.791	1.888	1.799	0.005	0.00302
50	1.907	1.952	1.866	0.00512	0.00328
60	1.954	2.193	1.923	0.0055	0.0035
70	1.994	2.286	2.034	0.00598	0.00398
80	2.092	2.694	2.129	0.00632	0.00427
90	2.401	2.827	2.388	0.00664	0.00463
100	2.495	2.887	2.545	0.00697	0.00499

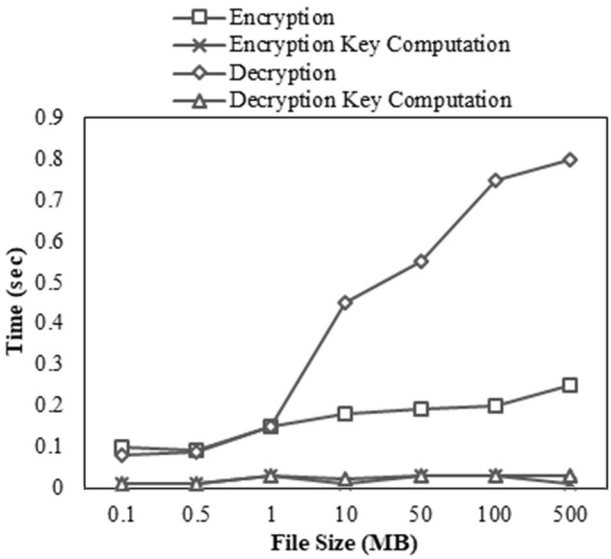

Fig. 4 Time consumption for encryption and decryption with varying file size

The file decryption time is the time taken to decrypt a file as per the requisition of decryption request by a CU. Figure 4 demonstrates that the outcome for decryption has comparable tendency as in the case of encryption, but the decryption time has a drastic rise in time proportional to the file size. It is also noted that the time consumption ranges from 0.1 to 0.3 s in the case of encryption, whereas, in decryption process the time consumption ranges from 0.1 to 0.8 s. With smaller files, the decryption key computation time creates a high percentage of the total decryption time, while it makes a minor proportion for larger files.

4.2.3 File Upload and Download Time

The performance of the proposed methodology is also evaluated based on the total time taken to upload/download a file to (or from) the health cloud. The total time is comprised of: (a) the key computation time, (b) the encryption (or decryption) time, (c) the upload (or download) time, and (d) the total time consumed for all communications (request and response time for each communication for upload and download). The file upload time is defined as the time taken to transmit a healthcare data file from the data owner system to the health cloud system. Figure 5 illustrates the performance for the file upload time process with varying file size. The time to upload a file to the health cloud gets increased with the increase in the size of the file. In the instance of both 0.1 MB and 0.5 MB files, the time consumption for encryption establishes 2.8% of the total encryption time. Furthermore, the proportion has increased into 7% for 1 MB file size and 13% for 10 MB file size, which is 6% rise in encryption time estimation. 50 MB file size establishes 20% of total encryption time and the percentage remains 28% while the tendencies carry on with a larger file size of 500 MB. Wherein, the file encryption time upturns with the rise in the file size as clearly revealed in the Fig. 5. It may be noticeable that the key computation time and communication time remains stable by stating that it is independent to the file size.

File downloading process is the reverse of the uploading process, where it receives original data file from the HC. Depends on the file size and network bandwidth, the download time may differs. There is a gradual increase in the download time with rise in the file size from 0.1 to 500 MB. The smaller file size of 0.1 MB takes only 0.70 s, while 500 MB larger file size rises its download time prominently to 38.22 s for downloading a file. Figure 6 depicts the outcome for the file download time process, which downloads the file from the health cloud and then follows the decryption process. The tendency of outcomes is same as in the case of uploading a file. Moreover, the times in download as well as decryption are different.

Table 2 depicts the observation of turnaround times for upload and download process in various existing methodologies along with the proposed system. Wherein, 'FS' denotes the

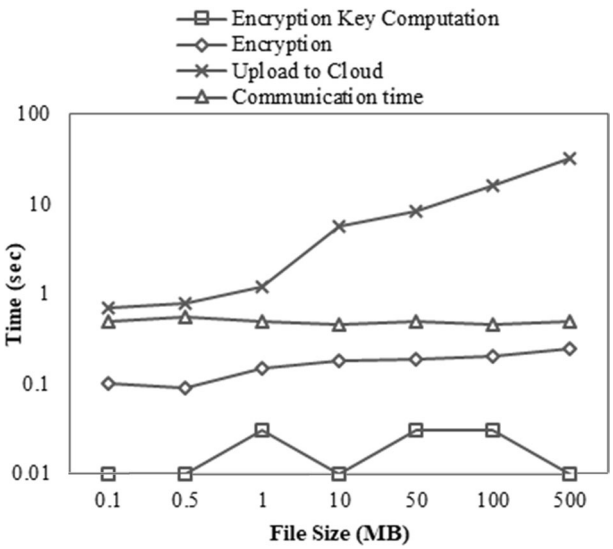

Fig. 5 Time consumption for file upload to the HC with varying file size

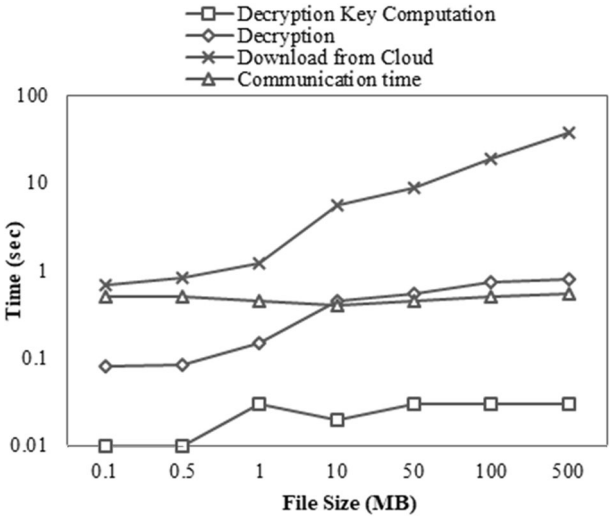

Fig. 6 Time consumption for file download from the HC with varying file size

Table 2 Observation of turnaround times for various methodologies

FS (MB)	Methodologies (time in seconds)									
	CL-PRE [14]		Certificateless encryption [15]		PRE [17]		AES [20]		ECC	
	UL	DL	UL	DL	UL	DL	UL	DL	UL	DL
0.1	0.90	0.81	1.4	0.99	1.48	1.15	0.80	0.80	0.70	0.70
0.5	1.18	0.96	1.48	1.03	1.89	1.31	0.94	0.96	0.80	0.82
1	1.80	1.39	2.06	1.48	2.90	1.85	1.24	1.18	1.20	1.24
10	13.05	9.91	14.95	9.90	14.59	10.45	6.43	6.48	5.60	5.68
50	53.68	33.45	58.56	35.57	60.37	35.90	9.01	10.24	8.25	8.78
100	99.69	57.14	112.41	59.14	115.15	61.59	17.39	20.68	16.35	18.98
500	369.72	215.3	492.03	229.81	872.09	400.21	33.24	39.25	32.10	38.22

File Size, 'UL' mentions 'upload', and 'DL' refers 'download'. This table clearly reveals that the proposed methodology outperforms the other existing systems such as CL-PRE scheme [14], certificateless encryption [15], PRE scheme [17], and AES method [20]. The proposed ECC based method provides better UL and DL performance, where the values can vary greatly due to the file size and internet connection speed. The smaller the file size, the smaller the UL and DL time. For the file size of 500 MB, the time difference between UL and DL is 6.12 s. ECC based method accomplishes optimum results for both smaller as well as larger file sizes.

4.2.4 File Uploading Speed

Generally, the upload speed is determined as the speed at which the data files are being send from the personal system to the internet. In the proposed ECC-based system, the uploading speed is calculated by how fast the CU can transmit the encrypted files to the health cloud. Depending upon the bandwidth rate, the uploading speed may varies under

Table 3 Performance analysis for uploading speed and security overhead

File size (MB)	Uploading speed (Mb/s)	Security overhead (%)
0.1	11.5	19
0.5	12	18.8
1	11.9	18
10	12.92	15
50	12.5	14
100	13	13
250	13	13
500	13	13

several traffic load. The bandwidth rate of 100 Mbps is used in this system for the estimation of uploading speed. The upload speed is calculated for file size varying from 0.1 to 500 MB. The calculated upload speed ranges from 11.5 to 13 Mbps as in the Table 3. The result also conveys that the upload speed is almost constant with only the difference of 1.5 Mbps for the calculated range varying from 0.1 to 500 MB.

4.2.5 Security Overhead

In the ECC-based system, there may be a delay in the system over TTP-CS side due to the security overhead. The security overhead in % is defined as the security operation time dividing the file transmission time. Table 3 displays the security overhead, which is clearly revealed that the percentage become constant for large files and is no more than 13%. The smaller file size (0.1 MB) establishes 19% of security overhead that is 6% higher than that of larger files. For the files 1 MB and 50 MB, the difference in overhead percentage is 4%, whereas it remains only 1% for 10 MB and 50 MB files. The outcome of this parameter evidently stated that the larger the file size, the smaller the security overhead by means of percentage.

5 Conclusion

In this research work, we have presented a security mechanism for secure sharing of health center's medical data through the health cloud system. The proposed health cloud architecture takes the advantages of ECC based cryptography system for secure sharing. Furthermore, the encryption and decryption processes are accomplished at the TTP-CS level. Our performance analysis indicates that ECC based model performs better than other existing systems in terms of the key generation time, file encryption time, file decryption time, file upload time, file download time, uploading speed, and security overhead. Another major significance of ECC based mechanism is smaller key size (256 bits), which makes key management simple. The outcomes exposes that the ECC based methodology can be practically employed for secure healthcare data sharing in the health cloud. Due to the performance of compute-intensive tasks at the TTP-CS, the ECC based scheme can be also utilized in mobile cloud computing environment.

References

1. Miranda, J., Memon, M., Cabral, J., Ravelo, B., Wagner, S. R., Pedersen, C. F., et al. (2017). Eye on patient care: Continuous health monitoring: Design and implementation of a wireless platform for healthcare applications. *IEEE Microwave Magazine, 18*(2), 83–94.
2. Abo-Zahhad, M., Ahmed, S. M., & Elnahas, O. (2014). A wireless emergency telemedicine system for patients monitoring and diagnosis. *International Journal of Telemedicine and Applications, 2014,* 4.
3. Jin, Z., & Chen, Y. (2015). Telemedicine in the cloud era: Prospects and challenges. *IEEE Pervasive Computing, 14*(1), 54–61.
4. Hsieh, J.-C., & Hsu, M.-W. (2012). A cloud computing based 12-lead ECG telemedicine service. *BMC Medical Informatics and Decision Making, 12*(1), 77.
5. Wang, X. A., Ma, J., Xhafa, F., Zhang, M., & Luo, X. (2017). Cost-effective secure E-health cloud system using identity based cryptographic techniques. *Future Generation Computer Systems, 67,* 242–254.
6. Ruiz-Zafra, Á., Benghazi, K., Noguera, M., & Garrido, J. L. (2013). Zappa: An open mobile platform to build cloud-based m-health systems. In *Ambient intelligence-software and applications* (pp. 87–94). Springer.
7. Bourouis, A., Feham, M., & Bouchachia, A. (2012). A new architecture of a ubiquitous health monitoring system: A prototype of cloud mobile health monitoring system. arXiv:1205.6910.
8. Li, Z.-R., Chang, E.-C., Huang, K.-H., & Lai, F. (2011). A secure electronic medical record sharing mechanism in the cloud computing platform. In *2011 IEEE 15th international symposium on consumer electronics (ISCE)* (pp. 98–103). IEEE.
9. Wu, R., Ahn, G.-J., & Hu, H. (2012). Secure sharing of electronic health records in clouds. In *2012 8th International conference on collaborative computing: Networking, applications and worksharing (CollaborateCom)* (pp. 711–718). IEEE.
10. Liu, D.-l., Chen, Y.-p., & Huai-ping, Z. (2010). Secure applications of RSA system in the electronic commerce. In *2010 International conference on future information technology and management engineering (FITME)* (Vol. 1, pp. 86–89). IEEE.
11. Gola, K. K., Gupta, B., & Iqbal, Z. (2014). Modified RSA digital signature scheme for data confidentiality. *International Journal of Computer Applications, 106*(13), 13–16.
12. Arora, R., Parashar, A., & Transforming, C. C. I. (2013). Secure user data in cloud computing using encryption algorithms. *International Journal of Engineering Research and Applications, 3*(4), 1922–1926.
13. Rahumed, A., Chen, H. C., Tang, Y., Lee, P. P., & Lui, J. C. (2011). A secure cloud backup system with assured deletion and version control. In *2011 40th International conference on parallel processing workshops (ICPPW)* (pp. 160–167). IEEE.
14. Xu, L., Wu, X., & Zhang, X. (2012). CL-PRE: A certificateless proxy re-encryption scheme for secure data sharing with public cloud. In *Proceedings of the 7th ACM symposium on information, computer and communications security, 2012* (pp 87–88). ACM.
15. Seo, S.-H., Nabeel, M., Ding, X., & Bertino, E. (2014). An efficient certificateless encryption for secure data sharing in public clouds. *IEEE Transactions on Knowledge and Data Engineering, 26*(9), 2107–2119.
16. Itani, W., Kayssi, A., & Chehab, A. (2010). Energy-efficient incremental integrity for securing storage in mobile cloud computing. In *2010 International conference on energy aware computing (ICEAC)* (pp. 1–2). IEEE.
17. Khan, A. N., Kiah, M. M., Madani, S. A., Ali, M., & Shamshirband, S. (2014). Incremental proxy re-encryption scheme for mobile cloud computing environment. *The Journal of Supercomputing, 68*(2), 624–651.
18. Chen, Y.-R., & Tzeng, W.-G. (2012). Efficient and provably-secure group key management scheme using key derivation. In *2012 IEEE 11th International conference on trust, security and privacy in computing and communications (TrustCom)* (pp. 295–302). IEEE.
19. Chen, Y.-R., Tygar, J., & Tzeng, W.-G. (2011). Secure group key management using uni-directional proxy re-encryption schemes. In *INFOCOM, 2011 proceedings IEEE* (pp. 1952–1960). IEEE.
20. Ali, M., Dhamotharan, R., Khan, E., Khan, S. U., Vasilakos, A. V., Li, K., et al. (2017). SeDaSC: Secure data sharing in clouds. *IEEE Systems Journal, 11*(2), 395–404.
21. Zhang, L., Wu, Q., Mu, Y., & Zhang, J. (2016). Privacy-preserving and secure sharing of PHR in the cloud. *Journal of Medical Systems, 40*(12), 267.
22. Rao, Y. S. (2017). A secure and efficient ciphertext-policy attribute-based signcryption for personal health records sharing in cloud computing. *Future Generation Computer Systems, 67,* 133–151.

V. Sri Vigna Hema Ph.D. Research Scholar in Information and Communication Engineering at Anna University Regional Campus—Tirunelveli, India. She had received a Master's in Computer Science and Engineering from the same university. Her main research interests are network security, pricing and security in cloud computing. She has published several papers in national conferences and international journal.

Dr. Ramesh Kesavan Ph.D. is working as Assistant Professor of Computer Application, Anna University Regional Campus—Tirunelveli, India. His area of research interest includes cloud computing, big data analytics, data mining, and machine learning. He has published numerous articles in referred journals, national and international conferences.