International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 5 118 – 124

Securing Patient Data Access using Segmented Key Management Approach

Dr. M. Sughasiny
Assistant Professor, Department of Computer Science
SrimadAndavan Arts and Science College (Autonomous)
Trichy, Tamilnadu, India
sughasiny5.cs@gmail.com

C. Sengamalai
Research Scholar, Department of Computer Science
SrimadAndavan Arts and Science College (Autonomous)
Trichy, Tamilnadu, India
karthickkavian1968@gmail.com

Abstract—Cloud technology can be utilized to empower data sharing capacities, which can profit the client through more noteworthy efficiency and profitability. Nonetheless, the Cloud is defenseless to numerous security vulnerabilities and privacy, which thwarts the advance and wide scale reception of data sharing for the reasons for cooperation. Along these lines, there is a solid interest for data owners to not just guarantee that their information is kept private and secure in the Cloud, however to likewise have a level of control over their own particular data contents once they are imparted to data consumers. In particular, the principle issues for data sharing in the Cloud incorporate security attacks, key management and data owner access control. As far as key management, it is key that data should first be encrypted before storage in the Cloud, to prevent security breaches and privacy. In this paper, a segmented key management is proposed.

Keywords-*Key Management, RSA Encryption Public Key, Doubled Elgamal Private Key, AES Encryption,Heart Disease, Cloud Storage Provider*

*****

I. Introduction

Cloud technology can be utilized to empower data sharing capacities, which can profit the client through more noteworthy efficiency and profitability. Nonetheless, the Cloud is defenseless to numerous security vulnerabilities and privacy, which thwarts the advance and wide scale reception of data sharing for the reasons for cooperation. Along these lines, there is a solid interest for data owners to not just guarantee that their information is kept private and secure in the Cloud, however to likewise have a level of control over their own particular data contents once they are imparted to data consumers. In particular, the principle issues for data sharing in the Cloud incorporate security attacks, key management and data owner access control. As far as key management, it is key that data should first be encrypted before storage in the Cloud, to prevent security breaches and privacy. In this paper, a segmented key management is proposed.

There is right now a solid need to propel the field of health informatics [2]. As the total populace ages because of expanded future, this spots weight on the government to store going through related with the maturing populace, particularly as far as health burning through. Thus, the interest for cutting the cost of medicinal services has expanded, and there is presently a developing requirement for the remote care of patients at home, especially for the elderly and the physically handicapped. By utilizing the ability of mobile technology and in addition Cloud computing, one can then build up a health checking framework where the patient can be evaluated by specialists in a remote area, from the solace of their own home. There are a plentiful number of mobile applications accessible today, for mobile telecare [3].

Lately, there has likewise been a developing requirement for the sharing of health information between social insurance groups that incorporate specialists, medical attendants and relatives. A few advantages of sharing health information incorporate more secure and better health results for the patient, as the health proficient gets a more entire medicinal history. This is predominantly due to not repeating the therapeutic history each time a health expert is counseled, and furthermore not any more superfluous tests. Sharing health data is likewise key to bringing down social insurance costs [4]. Notwithstanding, the principle issue with sharing health data is the protection and security dangers related with it.

Expanding on advances in Cloud computing, we look to go past the portable health applications, to empower the protected sharing of telecare information in the Cloud. The Cloud, as an empowering agent for mobile telecare, can give the successful treatment and care of patients because of its advantages, for example, on-demand gets to anyplace and whenever, high elasticity and low expenses. Be that as it may, the Cloud is defenseless to security attacks and privacy, a hefty portion of which happen from inside the Cloud suppliers themselves [5], as they have direct access to stored information.

II. Related Works

TABLE 1: Related Works on the Health Monitoring System using Cloud Computing

Authors Name and Application Name	**Paper Title**	**Explanation**
G. Fortino, M. Pathan, and G. Di Fatta [6]	Bodycloud: Integration of cloud computing and body sensor networks	Presented the Body Cloud design, which empowers the management and observing of body sensor information by means of the Cloud. It gives the usefulness to get and oversee sensor information consistently from a body sensor Network (BSN)
F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, and M. Sgroi [7]	Spine: a domain-specific framework for rapid prototyping of wbsn applications	It have introduced the SPINE system. This open-source structure permits designers to quickly model and oversee BSN applications
G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari [8]	Enabling effective programming and exible management of efficient body sensor network applications	There are two principle segments of the SPINE system: the coordinator side, which is executed on a PC or cell phone, and the BSN hub side. On the coordinator side, SPINE furnishes application

		engineers with an instinctive interface to the BSN, while on the hub side, SPINE gives designers deliberations of hardware resources, for example, sensors, and an engineering to tweak and extend the structure to support new physical stages and management
S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya [9]	An autonomic cloud environment for hosting ecg data analysis services	Incorporates mobile and Cloud technologies with electrocardiogram (ECG) sensors, to empower the remote observing of patients with heart related issues, for example, cardiovascular arrhythmias. The patient interfaces the sensors to their body and after that run an application on a cell phone. The application associates with the sensors through Bluetooth. The application will then intermittently transfer information to the Cloud. The client can then download charts from the Cloud, which speak to the client's health status. The plan likewise actualizes middleware in the Cloud
Alivecor Application [10]	Alivecor (tm) mobile ecg device for heart rhythm monitoring now available by prescription	It is a remote application based ECG checking framework. The framework is like our framework in that it permits a patient to screen their ECG on their iPhone and furthermore share their ECG information to whomever the patient needs. It gives a heap of valuable elements, for example, recording, showing, exchanging and storing great ECG information. Notwithstanding, the framework was not created because of security; in this manner, it is conceivable that an interloper will have the capacity to take health information with a specific measure of exertion
Cardiocomm solutions [11]	Cardiocomm solutions, inc. reveals a new remote mobile ecg monitoring solution at medica	CardioComm Solutions have likewise exhibited their remote patient ECG observing management, Heart Check Smart Monitoring. The framework takes into account quick get to and for doctors to better audit the ECG information so as to survey how the patient ought to be dealt with. Be that as it may, it doesn't particularly concentrate on security angles and privacy, for example, the classification of information as it is being transmitted to the Cloud or as it is put away inside the Cloud
S. Gradl, P. Kugler, C. Lohmuller, and B. Eskofier [12]	Real-time ecg monitoring and arrhythmia detection using android-based mobile devices.	It have likewise built up an Android-based application that takes into consideration the ongoing checking of ECG information, like our model, and in addition mechanized arrhythmia recognition. Be that as it may, the application additionally does not concentrate on security and privacy viewpoints
H. Xia, I. Asif, and X. Zhao [13]	Cloud-ecg for real time ecg monitoring and analysis	It addresses the helpfulness of ECG information gathered from patients themselves utilizing cell phones, and the issues that this presents. They don't concentrate on the security angles related with sending information to the Cloud

Wherever Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts if possible. True-Type 1 or Open Type fonts are preferred. Please embed symbol fonts, as well, for math, etc.

III. PROPOSED SEGMENTED KEY MANAGEMENT APPROACH

The primary thought is that the information is encrypted utilizing any AES symmetric encryption algorithm. The encrypted data is then stored to the Cloud. The symmetric key used to encrypt the information is then encrypted utilizing the RSA public key. Consequently, the best way to decrypt the symmetric key is by utilizing the ElGamal private key.

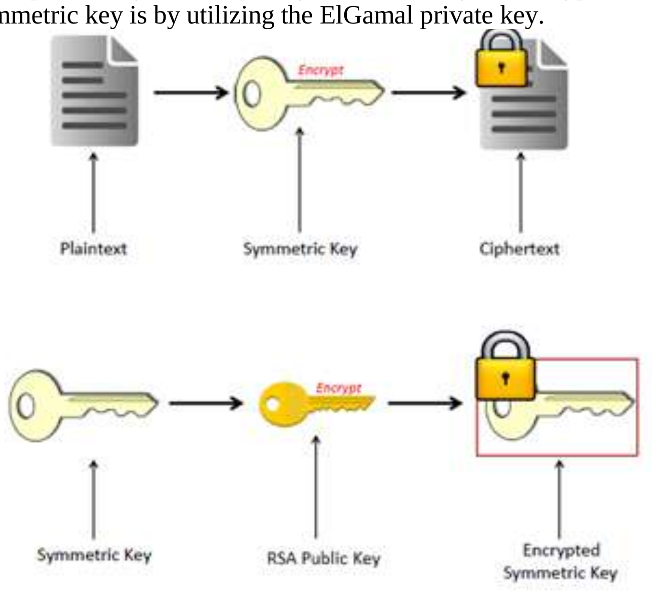

Figure 1: Encrypted Data and Key

The data is first encrypted with a symmetric key and that symmetric key is then encrypted using the RSA public key of the data owner. That is,

$$E_k(d) = C$$
$$G_{pub}(k) = K$$

where k is the symmetric key, E is the symmetric encryption operation, C is the ciphertext, G is the RSA encryption operation, pub is the RSA public key of the information owner and K is the encrypted symmetric key.

Since the ElGamal algorithm speaks to its private keys and public keys as huge numbers, this makes key partitioning possible and subsequently, fractional decryption is additionally conceivable. In this way, if we somehow happened to partition the ElGamal private key C into two sections A and B with the end goal that A + B = C, the symmetric key could be somewhat decrypted utilizing A and the in part decrypted key can then be completely decrypted utilizing B. Our joined symmetric and asymmetric encryption plan is highlighted in the following diagram:

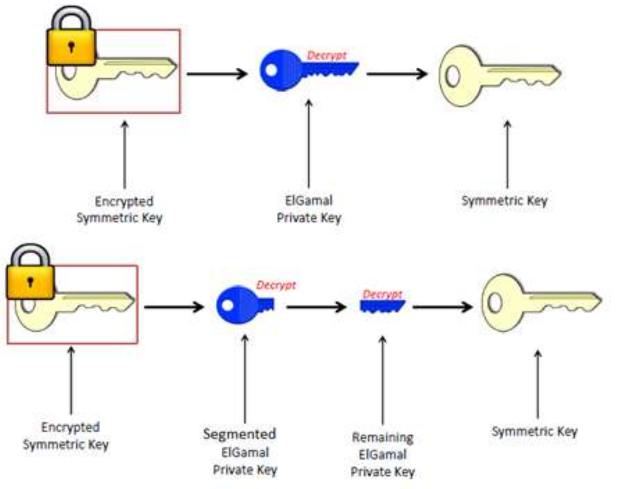

Figure 2: Proposed Segmented Key Management Approach

IV. HEALTH MONITORING SYSTEM USING PROPOSED APPROACH

Since the Cloud is at the front line of numerous security attacks and privacy, and numerous security attacks originate from inside the Cloud Service Provider (CSP) itself, as insiders ordinarily have direct access to information and may steal information to pitch to third parties with a specific end goal to pick up benefit, the whole database in the CSP should be encrypted. This implies the information should be encrypted before sending the information “over-the-wire." This will keep any noxious outsiders, and in addition the CSP itself, from increasing any helpful information without the decrypting key.

Since our attention is on data sharing to specialists and medical caretakers, basic encryption strategies are insufficient. As talked about, on the off chance that we have many nurses and doctors authorized to see the patient's information, and the patient chooses to renounce a particular specialist's get to rights to their information, the patient needs to re-encrypt their information utilizing another key and send the new key to the various specialists and attendants. This is computationally wasteful and places a weight on the patient to re-encrypt and distribute new keys, each time they renounce a specialist or medical caretaker's get to rights. It additionally puts a weight on the rest of the individuals from the gathering, as they always need to refresh their key set so as to keep up-to-date with the greater part of their patient's information. The primary motivation behind why the patient would need to re-encrypt the information with another key is that the renounced specialist still holds the key can in any case hypothetically get to the information, regardless of the possibility that he is not permitted to. It can't be expected that the specialist or medical attendant will never see the patient's information or that they will dependably keep the key a mystery. For instance, in the health area, there are standards, for example, HRIPA which is followed in NSW, Australia [14] or HIPAA (Health Insurance Portability and Accountability Act) which is followed in the US [15]. These guidelines plan to secure and uphold the classification of a patient's health related information and keep the information secret from anybody unless authorized by the patient. As it were, any element ought not get to a patient's health data without the patient's consent. Thus, clinics and health associations are hesitant to receive Cloud technology as a security rupture can destroy, particularly as far as cost [16]. In our work, we give an answer which uses the Cloud to help guarantee health information is kept private and secure.

A. *Data Model*

Figure 3 depicts the eHealth monitoring framework, by including a security layer that empowers secure and efficient data sharing. The first web service now speaks to the Cloud Data Service (CDS), and we include another web benefit called the Data Sharing Service (DSS) that handles the data sharing parts of the framework. We accept that the DSS is completely trusted.

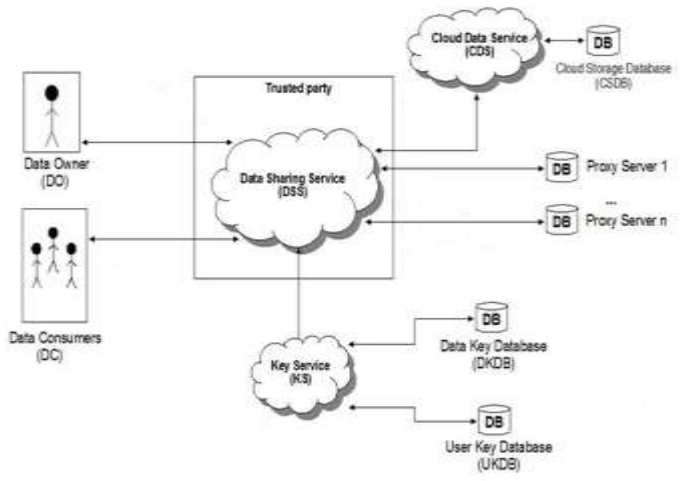

Figure 3: Data Sharing Model of Health Monitoring System

Notwithstanding, this makes it especially powerless against attacks; thusly, the DSS itself should be ensured. Keeping in mind the end goal to accomplish this, the DSS can be demonstrated as a trusted private Cloud supplier that is secured utilizing traditional mechanisms, for example, Internet firewalls. There are likewise various proxy services to store key pieces for individuals from the gathering, and a Key Service (KS) to store the encrypted keys of the health information and the keys of the data customers (DC).

To quickly condense how the model functions, we expect that every client in the gathering, including the Data Owner (DO), has a key that can decrypt the fitting keys in the Data Key Database (DKDB). Be that as it may, their keys are partitioned into n+ 1 part, where n parts are stored in every proxy and the client keeps the additional part. Along these lines, none of the clients know the full key required to decrypt the keys in the Data Key database. At the point when the client requires information get to, they call the DSS. The DSS then decrypts the key in the DKDB utilizing the greater part of the key pieces in the proxy database that compare to the calling client. The key is then used to decrypt the information in the Cloud. At the point when the data owner asks for that a client's get to is disavowed, their key pieces in the proxies are just expelled and the first information require not be re-encrypted, nor there any re-distribution of keys to outstanding clients. None of the other data customers will be influenced by the renouncement, since their relating key pieces still stay in place in the proxies and furthermore with themselves.

B. *Protocol*

We now talk about our data sharing protocol in detail. The protocol has four stages: customer revocation, initialization, authorized data access and customer authorization. It is additionally essential to note that we expect the DSS to be completely trusted, in that it will dependably sincerely take after the protocol we make utilization of ElGamal encryption in our work and expand upon the work of Tran et al. [17] to give a more secure stage to data sharing. The following table contains brief meanings of the truncations utilized as a part of our protocol.

Table 2: List of abbreviations and Explanation

Definitions	Abbreviations	Explanation
Data Owner	DO	The owner decides who can access the data and give permission to the data.
Data Consumer	DC	Any user who has permission to access data given by the DO
Data Sharing Service	DSS	In the protocol, the most functionality of the data-sharing is carried out in this trusted service.
Cloud Data Service	CDS	The call to be constructed to the Cloud Storage is allowed by this service.
Key Service	KS	The administration that permits calls to be made to the Cloud key service, to acquire and store encryption keys
Cloud Storage Database	CSDB	The database containing encrypted information
Data Key Database	DKDB	The database that stores encryption keys which are they encrypted.
User Key Database	UKDB	The database that stores all clients, including DC and DO private keys.

1) Initialization

Step 1: Key Request DO->DSS (To upload data to the cloud, first a request is send to the DSS by DO)
Step 2: Generate Key x DSS $b=c^x$ mod p (A random private key is generated by DSS also its appropriate public key {p,b,c} is also generated by RSA Encryption)
Step 3: DSS Generate $x_1+ x_2+x_3+ \ldots+x_n+x_{n+1} = x$. Generate u_{DO} (The DSS then partitions x into n + 1 pieces. The DSS also generates new user identification for the data owner)
Step 4: for (all proxy i) (stores each piece in each of the n proxy servers)
DSS -> proxy i { u_{DO} , u_i }
Step 5: DSS->DO { u_{DO} , x_{n+1} , {p,b,c}} (The DSS then sends the user identification, the remaining partitioned key piece, and the public key, to the DO)
Step 6: DO Generate symmetric key k $E_k(m)$, generate r; = c^rmod p (The DO then generates a random symmetric key k and encrypts his data with it. The symmetric key is then encrypted itself by the DO, using the public key {p,b,c} generated by the DSS)
$E_{\{p,b,c\}}(k) = (c^r, c^{rx}.k \bmod p)$
Step 7: { u_{DO} , $E_k(m)$, $E_{\{p,b,c\}}(k)$ } (The DO then sends his user identification, the encrypted data and the encrypted key, to the DSS)
Step 8: Generate d_m (The DSS generates a data identification for the data)
Step 9: { u_{DO} , d_m, $E_k(m)$ } (The DSS then sends the data identification and the encrypted data to the CDS (9) for storage.)
Step 10: { u_{DO} , d_m, $E_{\{p,b,c\}}(k)$ } (The DSS finally sends the data identification and the encrypted key to the KS).

2) Consumer Authorization

Step 1: {access_request, d_m} DC to DO (When a DC wishes to access the DO's data m, he sends an access request to the DO along with the data identification of the data he wishes to gain access)
Step 2: addUser(u_{DO} , d_m, x_{n+1}) DO to DSS (Assuming the DO approves, he sends a request to the DSS and sends the request along with his user identification, the data identification, and key piece)
Step 3: Verify u_{DO}, d_m exists. If not, exit here. (The DSS then verifies whether the data identification and data owner identification exist, with a call to the CDS). If the CDS returns false, then the DSS notifies the DO that the data does not exist and exits the protocol.
Step 4: for (all proxy i) (If the CDS returns true, the DSS then retrieves the DO's key pieces from the proxy)
{ u_{DO} , key_piece_request)} DSS to proxy i
x_i proxy i to DSS
Step 5: Compute $x_1+x_2+x_3+\ldots+x_n+x_{n+1} = x$ (computes the secret key x by adding all the key pieces together)
Generate $x_{u1}+x_{u2}+x_{u3}+\ldots+x_{um}+x_{u(n+1)} = x$ (The DSS will then generate new key pieces for the new DC that, when combined, are equivalent to the secret key x)
Generate u_{DC}(The DSS will also generate a random user identification as well as a public/private key pair, using ElGamal encryption for the DC)
Generate {p_{DC}, b_{DC}, c_{DC}}, x_{DC}
Step 6: The DSS will then send the DC's user identification, the public key and identifiers such as the DO user identification and data identification, to the KS. The KS will then store this in the UKDB
{ u_{DC}, u_{DO}, d_m, { p_{DC}, b_{DC}, c_{DC} } } DSS to KS then KS to UKDB ()
Step 7: The newly generated key pieces corresponding to the DC are then stored in each of the proxy servers
for (all proxy i)
{ u_{DC}, u_{DO}, d_m, x_{ui}} DSS to proxy i
Step 8: the remaining piece is sent to the DO along with the private key of the DC. The DO finally sends this to the DC in a secure manner.
{ u_{DC}, $u_{u(n+1)}$, x_{DC}} DSS to DO then DO to DC

3) Authorized Data Access

Step 1: When a DC wishes to access data, he sends his key piece to the DSS along with identifiers to the data.
{ u_{DC} , u_{DO} , d_m , $x_{u(n+1)}$ }
Step 2: The DSS obtains the encrypted key from the DKDB via a call to the KS.
getKey (u_{DO} , d_m)
Step 3: $E_{\{p,b,c\}}(K) = (c^r, c^{rx}.k \bmod p)$
Step 4: The DSS then calls each proxy server to obtain the corresponding key piece of the DC.
getKeyPiece(u_{DC})
Step 5: x_{iu}
Step 6: decrypts the encrypted key using each key piece.
$D_{x_{iu}}(E_{\{p,b,c\}}(K)) = (c^r , (c^r))^{-xiu} . c^{rx}.k \bmod p)$

$= (c^r, (c^r)^{x-xiu}$ .k mod p)

Step 7: Repeat the step 4-6 for proxies 2..n

Remaining ciphers: $(c^r, (c)^{r^{x}} {}^{-x_{1u}-x_{2u}-\cdots-x_{nu}}$ .k mod p)

Step 8: The DSS then uses the DC's key piece from step (1) and decrypts the remaining encrypted key to reveal the full key

$$D_{x_{u+1}}((c^r, (c^r)^{x-x_{1u}-x_{2u}-\cdots-x_{nu}} .\text{k mod p}))$$

$$(c^r, (c^r)^{-x_{u(n+1)}}.(c^r)^{x-x_{1u}-x_{2u}-\cdots-x_{nu}} .\text{k mod p}))$$

$$(c^r, (c^r)^{x-x_{1u}-x_{2u}-\cdots-x_{u(n+1)}} .\text{k mod p})$$

$(c^r, k\ mod\ p) since$ x= $x_1 + x_2 + x_3 + \cdots + x_n + x_{n+1}$

Step 9: The DSS then obtains the encrypted data from the CSDB via calls to the CDS

getData(u_{DO}, d_m)

Step 10: $E_k(m)$

Step 11: The encrypted data is then decrypted with the full key, to reveal the full plaintext. The DSS then generates another arbitrary symmetric key and encrypts the data with this key.

$D_k(E_k(m)) = m$

Generate k1

$E_{k1}(m)$

Step 12: The DSS obtains the corresponding DC's public key from the UKDB.

getUserKey(u_{DC})

Step 13: { p_{DC}, b_{DC}, c_{DC} }

Step 14: Encrypts the symmetric key using the public key

Generate $r_{DC}, \gamma_{DC} = c_{DC}{}^{r_{DC}}$ mod p_{DC}

$E_{\{\text{pDC,bDC,cDC}\}}(k1) = c_{DC}{}^{r_{DC}}, c_{DC}{}^{r_{DC} x_{DC}}$. k1 mod p

Step 15: The encrypted data and the encrypted key are sent to the DC.

$\{E_{\{\text{pDC,bDC,cDC}\}}(k1), E_{k1}(m)\}$

Step 16: The DC can then decrypt the key using his earlier distributed private key. Once the key is decrypted, the DC can then decrypt the data itself, to reveal the full plaintext.

$D_{xDC}(E_{\{\text{pDC,bDC,cDC}\}}(k1), E_{k1}(m))$

$= (c_{DC}{}^{r_{DC}}, (c_{DC}{}^{r_{DC}})^{-x_{DC}} c_{DC}{}^{r_{DC} x_{DC}}$. k1 mod p

$= (c_{DC}{}^{r_{DC}}, k1\ mod\ p)$

$D_{k1}(E_{k1}(m))$ = m

4) *Consumer Revocation*

Step 1: When the DO decides to revoke a user's access rights to data, he simply calls the DSS to request the revocation of the user's rights to the specified data.

removeUser(u_{DO}, u_{DC}, d_m)

Step 2: The DSS will then remove the corresponding key pieces of the user in each of the proxy databases. Note that the data does not need to be re-encrypted and none of the other users will be affected, since only the key pieces corresponding to the user are removed. All other key pieces corresponding to other users remain in the proxy database. Since the data does not need to be re-encrypted, nor does there need to be any key re-distribution, the model is efficient and has a runtime of O(n), where n is the number of proxies.

For (all proxy i)

removeKeyPiece(u_{DO}, u_{DC}, d_m)

proxy i Remove x_{uDCi}

V. PERFORMANCE TEST AND DISCUSSIONS

We did various execution tests on our protocol, basically the downloading and uploading of ECG information. The motivation behind the execution test was to test whether such a protocol will be attainable for use by regular individuals. Each of the execution tests were done on 10 seconds of ECG information, or 3,000 examples of ECG data points.

For the uploading tests, we measured to what extent it would take for a patient to upload their ECG information to the Cloud. We measured the time it took from the minute the patient presses the upload button on their application, to the storage of information in the database. We did 10 test cases and for each experiment, we measured the time it took for the patient demand to achieve the Cloud administration and after that from the support of Cloud stockpiling. We likewise completed the experiments utilizing the safe information sharing convention and after that once more, without the protected information sharing convention.

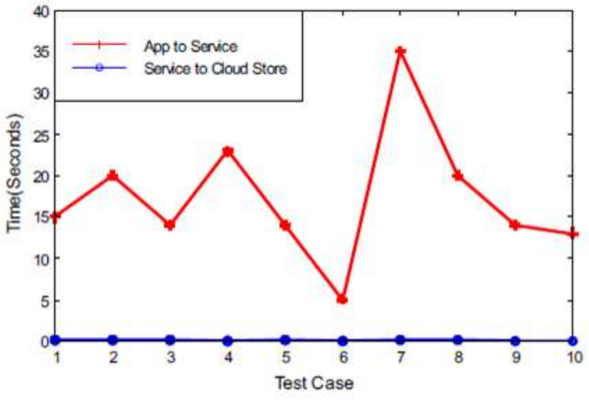

Figure 4: Uploading times with Security Protocol

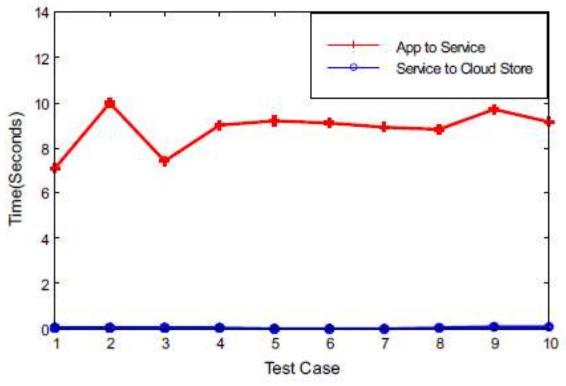

Figure 5: Uploading times without security protocol

We did various execution tests on our framework, basically the transferring and downloading of ECG information. The motivation behind the execution test was to test whether such a framework will be attainable for use by regular individuals. Each of the execution tests were done on 10 seconds of ECG information, or 3,000 examples of ECG information focuses.

For the transferring tests, we measured to what extent it would take for a patient to transfer their ECG information to the Cloud. We measured the time it took from the minute the patient presses the transfer catch on their application, to the capacity of information in the database. We did 10 test cases and for each experiment, we measured the time it took for the patient request to achieve the Cloud service and after that from the support of Cloud storage. We likewise completed the experiments utilizing the safe data sharing protocol and after that once more, without the secure data sharing protocol.

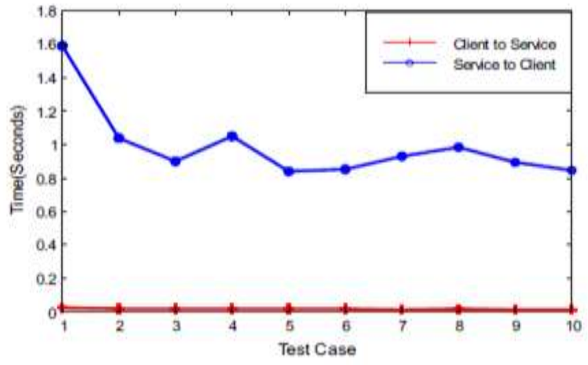

Figure 6: Downloading times with Security Protocol

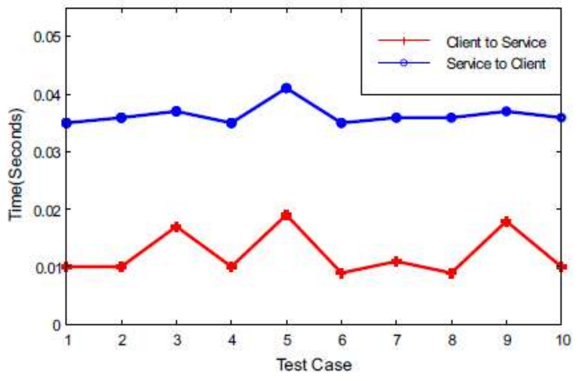

Figure 7: Downloading times without security protocol

From our outcomes, we found that this time, it was substantially quicker for the patient request to achieve the Cloud service. In spite of the fact that the time taken to recover information from Cloud storage and return it to the client was marginally more, despite everything it gave back the outcomes in under one moment.

The total downloading and uploading times are highlighted in the accompanying figures. We measure the overhead presented with the security protocol set up, contrasted with a framework with no security measurement at all.

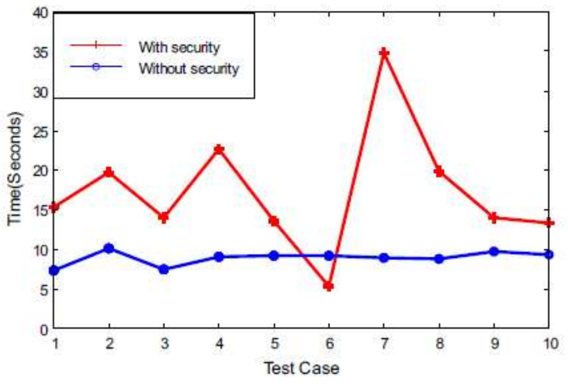

Figure 8: Uploading Overhead

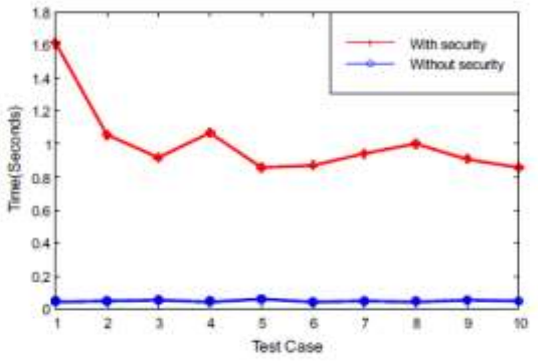

Figure 9: Downloading Overhead

From our outcomes, the security protocol introduced fundamentally more overhead contrasted and without the security protocol. We likewise found that uploading times by and large took a great deal longer, contrasted with download times when the security protocol was set up. Without the security protocol, the distinction was unimportant. The normal time taken to upload information to the Cloud was 17.27 seconds, with a standard deviation of 7.40 seconds, with the security protocol set up. Without the security protocol set up, the normal time was 8.89 seconds, with a standard deviation of 8.92 seconds. With the security protocol set up, recovery times took around 1-2 seconds. With the security protocol set up, the normal download time was 1.00 second, with a standard deviation of 0.21 seconds; without the security protocol, the normal was 0.05 seconds, with a standard deviation of 0.01 seconds.

VI. CONCLUSION

We displayed a key-partitioning system that would permit proficient key management. To quickly depict the key-partitioning strategy, the encrypted information key is partitioned into two (or potentially more) parts. The Cloud provider keeps one segment and the data consumer keeps the other. At the point when data consumer request information get to, the Cloud provider halfway decrypts the information with the key, and sends this to the information customer. The information consumer then completely decrypts, utilizing the remaining key partitioning. This guarantees neither the Cloud provider nor the data consumer knows the completely decrypted key. We introduced our thought through an application situation including the observing of patients health and furnishing them with criticism. In this situation, the execution overhead to store and recovering health data through our created models was significant and along these lines achievable to use in a true situation.

REFERENCES

[1] Lorenz, Klara, et al. "Technology-based tools and services for people with dementia and carers: Mapping technology onto the dementia care pathway." *Dementia* (2017): 1471301217691617.

[2] Moghaddasi, Hamid, and Alireza Tabatabaei Tabrizi. "Applications of Cloud Computing in Health Systems." *Global Journal of Health Science* 9.6 (2016): 33.

[3] Lee, Ming-Huei, et al. "Multidisciplinary self-management telecare system may improve quality o life in patients with interstitial cystitis/bladder pain syndrome (IC/BPS)–A randomized controlled study." *Urological Science* 27.2 (2016): S15.

[4] Susanto, Heru, and Chin Kang Chen. "Information and Communication Emerging Technology: Making Sense of Healthcare Innovation." *Internet of Things and Big Data Technologies for Next*

Generation Healthcare. Springer International Publishing, 2017. 229-250.

[5] Mohit, Prerna, et al. "A Standard Mutual Authentication Protocol for Cloud Computing Based Health Care System." *Journal of medical systems* 41.4 (2017): 50.

[6] G. Fortino, M. Pathan, and G. Di Fatta. Bodycloud: Integration of cloud computing and body sensor networks. pages 851-856, Dec 2012.

[7] F. Bellifemine, G. Fortino, R. Giannantonio, R. Gravina, A. Guerrieri, and M. Sgroi. Spine: a domain-specific framework for rapid prototyping of wbsn applications.Software: Practice and Experience, 41(3):237-265, 2011.

[8] G. Fortino, R. Giannantonio, R. Gravina, P. Kuryloski, and R. Jafari. Enabling effective programming and flexible management of efficient body sensor network applications. IEEE Transactions on Human-Machine Systems, 43(1):115-133, Jan 2013.

[9] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya. An autonomic cloud environment for hosting ecg data analysis services. Future Generation Com- puter Systems, 28:147-154, 2012.

[10] BusinessWire. Alivecor(tm) mobile ecg device for heart rhythm monitoring now available by prescription. Business Wire, 2013.

[11] BusinessWire. Cardiocomm solutions, inc. reveals a new remote mobile ecg monitoring solution at medica 2011. Business Wire, 2011.

[12] S. Gradl, P. Kugler, C. Lohmuller, and B. Esko_er. Real-time ecg monitoring and arrhythmia detection using android-based mobile devices. pages 2452-2455, Aug 2012.

[13] H. Xia, I. Asif, and X. Zhao. Cloud-ecg for real time ecg monitoring and analysis. Computer Methods and Programs in Biomedicine, 110:253-259, 2013.

[14] H. Xia, I. Asif, and X. Zhao. Cloud-ecg for real time ecg monitoring and analysis. Computer Methods and Programs in Biomedicine, 110:253-259, 2013.

[15] U.S. Department of Health and Human Services. Hipaa privacy. U.S. Department of Health and Human Services Website, 2012.

[16] J. Saarinen. Uk health trust fined for privacy breach. Itnews Technology News, 2012.

[17] S. Geoghegan. The latest on data sharing & secure cloud computing. Law & Order, pages 24-26, 2012.