

Secured Federated Data Management in Distributed Mobile Cloud Computing

Abubakar Usman Othman¹; Muhammad B. Abdullahi²; Moses Timothy³

Graduate Student, Department of Computer Science, Federal University of Technology Minna, Nigeria¹;

Lecturer, Department of Computer Science, Federal University of Technology Minna, Nigeria²;

Lecturer, Department of Computer Science, Federal University Wukari, Nigeria³
Othman80s@yahoo.com¹; el.bashir@futminna.edu.ng²; visittim@yahoo.com³

ABSTRACT

With the development of wireless access technologies such as 3G/4G, mobile devices and sensors are used as information collection nodes for cloud which offers immense benefit for mobile users. Federating two or more cloud service providers and caching frequently accessed data drive is an effective way to seamlessly access data in mobile cloud computing environment thereby overcoming server failover, frequent disconnection of database server, lock-in by single service provider and data location management issues. Privacy and security of mobile users' data however, remain the challenges that outweigh the colossal success factor of a federated data management in Mobile Cloud Computing. This is owing to the fact that data are hosted on public clouds owned by commercial service providers. Thus, new secured service architecture is required to provide security to the federated data in the distributed mobile cloud computing environment. This work proposes an efficient and secured federated data management to ensure the privacy, confidentiality and integrity of mobile user data in a distributed mobile cloud computing.

Keywords: Cloud Computing, Data Management, Distributed, Efficient, Federated, Mobile Cloud, Mobile Support Station, Surrogate Object.

1. INTRODUCTION

Cloud computing provide the use of computing resources to deliver information technology (IT) as a service over the internet. The cloud computing objective is to optimize capacity and functionalities during runtimes and not employing new infrastructures, software, or new hands. Cloud computing allow users make use of cloud services on a pay-as-you-go principle in the web. The services include infrastructure as a service (IaaS), data

storage as a service (DaaS), software as a service (SaaS), security as a service (SecaaS), communication as a service (CaaS), platform as a service (PaaS), and business as a service (BaaS) (Armburst et al, 2009). Various layered architecture are available to provide these services as a utility for cloud computing. This layered architecture include the cloud backbone layer which consist of servers and switches, the supervisor layer which hosts software that manage the cloud infrastructure resources, software infrastructure layer which is responsible for handing over network resources to upper layer that provides a building block for an emerging computing paradigm which delivers IT as a service, the platform infrastructure layer provides application development platform and some set of application programming interfaces (API), the application layer provides platform for users to access different applications contained in the cloud provider's files centres using web (Sotomayor et al, 2009).

Resource constraint mobile devices depend on cloud for computation, information sharing and upload to perform computationally intensive operations such as data storage. The availability of such enabling environment for ubiquitous wireless infrastructure and resource constraint mobile devices to perform computation operations provide a platform for emerging computing paradigm known as mobile cloud computing (MCC) (Abdul et al, 2012). Mobile devices cannot perform computationally intensive and storage demanding operations because mobile devices being battery powered have minimal processing power, less energy, limited storage capacity and with less security. Moreover, if a cloud service provider lock-in or the database disconnect, mobile cloud users cannot access data which brings the need for federating multiple clouds. So federating with multiple clouds provides an effective way to seamlessly access data, but there is concern of mobile user's data being exposed to commercial

cloud service providers due to lack of confidentiality and integrity (Ravimaran et al, 2012). This challenge outweighs the colossal success factor of a federated data management in distributed mobile cloud computing. Federated data needs to be efficient and secured to ascertain the integrity of mobile user's data files uploaded in cloud storages to avoid being exposed to the commercial service providers (Jian et al, 2011).

Federation in its simplest form means two cloud service providers sharing some data. The data could be files, resources, users' identity, resource information or any other useful information that may demand constant use in the future. Surrogate objects enable seamless data access for users. Surrogate objects are software entities that are hosted on some mobile support station (MSS) and act on behalf of mobile devices as well as the middleware component between the mobile user and cloud service provider (CSP) in the cloud. Federation increases data availability and avoids vendor lock-in as data are replicated to more surrogate objects to provide seamless data access and efficient data management in mobile cloud systems (Ravimaran et al, 2012).

2. RELATED LITERATURES

To solve the problem of securing federated data files and the resource constraint of mobile devices, many schemes aim at providing solutions to the arising problem. Canepa and Lee (2010) proposed that storage-demanding operations should be moved to the cloud. Kumar and Lu (2011) suggested that to offload storage operations on cloud storage servers, the internet condition and the communication overhead should be taken into consideration to make the offloading beneficial for mobile users. A security framework for efficient and secured data storage services in mobile cloud computing (MCC) was proposed by Zhibin and Huang (2011). In their proposed scheme, Privacy Preserving Ciphertext Policy Attribute-Based Encryption scheme was used to ensure the privacy of mobile user's data in the cloud. Since the intensive operation of encryption and decryption are outsourced to a third party, the resource constraint of mobile devices is relieved of the burden. The problem with this technique is that the ciphertext increases linearly with the number of ciphertext attributes.

Deswart et al (2003) proposed remote integrity checking. They used RSA-based functions for hashing entire data files in cloud storages for every verification challenge sent by the verifier. Their work failed to handle large data files that require longer time to efficiently compute and move hash

values. Public Provable Data Possession technique proposed by Jian et al (2011) for resource-constrained mobile devices ensures the privacy, confidentiality of mobile user's data files stored in cloud storage. In their scheme, a trusted third party does the encryption/decryption, encoding/decoding, signature generation, and verification of data files for mobile users. Though offloading saves energy, the addition of users brings degradation in output.

Amritha and Saravana (2013) also proposed a security model that does not allow data leakage in distributed mobile systems using surrogate objects. To secure the data on the object during transaction execution and reconciliation, the users need to be authenticated. There is efficiency in the proposed system since the surrogate objects are made to act on behalf of the server which drastically reduces the communication overhead. It also ensures confidentiality of transaction due to the presence of an encrypted tunnel between database server and the objects. The author used unique identification assigned to mobile users and its respective surrogate objects assigned to it when it enters or registers in the cloud to enable it to access the cloud. Non-registered mobile users will be denied access into the cloud if they try to access the cloud. Peng and Yanping (2013) proposed a cloud-based storage archive for Mobile Computing (CS-Mobile) that addresses the challenges of data storage in MCC. The framework provides an easy-to-use file navigation service and also provides a mechanism for users to verify their data with minimal burden. The aim of the author's work was to provide a lightweight storage system to resource-constrained mobile users. The framework composed four entities including mobile user, federated cloud storages, independent security server and CS-Mobile. While the scheme considers federated cloud storages, it does not support dynamic data operation in mobile cloud computing.

Itani et al (2010) proposed an energy-efficient framework for limited energy mobile devices to ascertain the correctness of data stored on cloud through incremental cryptography and trusted computing. The system is composed of three entities including mobile user, service provider and TTP. They discussed data upload, data insertion, data deletion, and verification for files in MCC. The authors described the use of incremental cryptography with respect to integrity verification. When uploading data on the cloud storage server, the mobile user generates an Incremental Message Authentication Code using a secret key. The problem with their system is that the privacy of mobile user's data is overlooked. The trusted coprocessors can handle a particular number of mobile users which did not give room for scalability. An increase in the

number of mobile users results to degradation in performance. Hsueh et al (2011) proposed a scheme for smartphones to provide security and verify data uploaded on cloud servers. They introduced a mechanism to authenticate users who uploaded data on cloud storage. The proposed system consists of the mobile device that utilizes the services provided by CSP and the certification authority authenticates each device; the telecommunication module which is responsible for generating passwords and also keeps record of information about mobile device to use the services of the cloud. In their work, the authors failed to address the processing and storage capability of the mobile device. Also, the encryption, decryption and hash computations performed by the mobile user through mobile devices consumed considerable amount of energy. Again, the advisory can impersonate the mobile user by utilizing the user's credentials since the certification authority is left with whole responsibility having access to users' credentials.

Jia et al (2011) developed a network model which is composed by data owner (DO), data sharer, and Cloud Service Provider. Information and security management are outsourced on cloud and the content of data is hidden to the cloud by using proxy re-encryption and identity base encryption schemes. The authors used a semi trusted proxy to transform information enciphered with user A's key into another information enciphered with user B's public key and the identity based encryption technique is used on bilinear mapping. The system failed to address the computationally intensive operations performed by mobile users. Also, performing cryptographic operations consume a lot of energy. A bilinear mapping is a mapping function.

3. METHODOLOGY

The proposed architecture for this work is presented in Figure 1. The system architecture incorporates the solutions to the problems identified. The system comprises three main entities; mobile user, surrogate object and storage service provider. The *mobile user* uploads large amounts of data files on cloud and has the permission to access, do transaction on uploaded data and can request for trusted validation of stored data. The second entity is *Surrogate Object* which is the trusted third party in this system. It has the ability and capability that the mobile user did not have. It is trusted with the computational task of data encryption, decryption, encoding, decoding and authentication to relieve resource constraint mobile user of the task on behalf of the mobile user. The third entity is *storage service provider* with

enormous amount of storage space to provide storage for mobile users and provides proof of data possession when requested. In mobile cloud computing paradigm, data are stored by cloud storage service provider thereby relieving mobile users with resource constraint mobile devices the burden of intensive computation and storage. As mobile user's data are been stored in the cloud storage, it becomes necessary for them to ensure that their data are stored properly and correctly maintained.

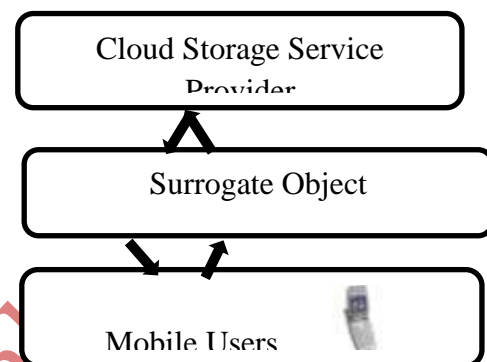


Figure 1: Architecture of the Proposed System

3.1 Application Scenario

When a mobile user joins the cloud and is registered with the Mobile Support Station (MSS). The MSS issued a unique identity as MID to the mobile user and assigned a surrogate object with a unique surrogate object identification number (SOID) for the mobile user. The mobile user sends message to the storage service provider (SSP) through its surrogate object. The computationally intensive operations of encryption, decryption, encoding and decoding is outsourced to a service provider (SP) which is the surrogate object of respective mobile devices. Messages are encrypted and encoded by SO which is the encryption and encoding service provider (ESP and EnSP) before sending to the SSP. Each surrogate object does the encryption and encoding operations for its particular mobile user. Mobile user requests for data access through its respective SO. The decryption service provider provides decryption services to the mobile device by decrypting the encrypted messages. The decoding service provider (DeSP) provides decoding service to the mobile device by decoding the encoded messages encrypted and encoded by the ESP and DeSP. Whenever mobile user wants to request for data verification, it does so through its SO. The SO provides or sends the verification request while the cloud storage provides proof of possession. Here we assume that the mobile devices are from MD_1 to MD_n

as in $MD_1, MD_2, MD_3, \dots, MD_n$. Each mobile device has an assigned surrogate object with a unique identification number. $SO_1, SO_2, SO_3, \dots, SO_n$. The surrogate object SO_1 does the encryption, decryption, encoding and decoding operations for mobile device MD_1 , SO_2 for MD_2 and so on up to SO_n for MD_n .

To prevent the system from any threat, the mobile user and corresponding surrogate object needs to establish a secured link of communication for preventing an intruder from interfering. This secured communication is achieved through a cryptographic scheme; Diffie-Hellman Protocol, which utilizes the discrete logarithmic problem to generate and safely exchange symmetric key for data exchange.

3.2 Remote Identification and Setup

In this subsystem, the mobile user and the SO familiarized with each other as registration is already done when the mobile user entered the cloud. The mobile user is issued with a mobile identification number (MID) and unique surrogate object identification as (SOID) by the mobile support station and then passes the information to the cloud. Once the mobile user enters the cloud, the unique MID and the unique SOID are identified. The mobile user generate a pair of symmetric key (pk, dk) as public key and private key. Through the Diffie-Hellman protocol, the pair of keys is exchanged between the mobile user and the SO. The mobile user initiates the process of the key exchange by choosing a number such as prime value Q , with an element s which is contain in q . The mobile user then compute Q depending on the generated secret value v , and the shared asymmetric key values x and y . Mobile user calculate Q as x to the power of v mod y .

$$Q = x^v \text{ mod } y \dots \dots \dots (1)$$

The result obtain from equation 1 above is sent to the SO while the value of v is kept as private. The SO calculate the value of R , from equation 2, depending on the privately generated value r and the shared secrete asymmetric keys x and y and send the output R , back to the mobile user. Mobile user calculate R as x to the power of w mod y .

$$R = x^w \text{ mod } y \dots \dots \dots (2)$$

Mobile user can only calculate the value of M if the SO send the value of R to mobile user. It is also not possible for the SO to calculate the value of M without knowing the value of Q . So M can be calculated depending on the output Q which has been received from the mobile user in the request message. Algorithm for the process is as follows:

Algorithm 1: Key Exchange

$$MU \rightarrow SO: x, x^v$$

$$SO \rightarrow MU: x^w$$

$$MU \rightarrow SO: \{F\}_{x^{vw}}$$

Table 1. Parameters used

S/no	Symbol	Uses
1	Φ	A generated signature collection
2	Φ'	Hash signature collection
3	F	Original data file
4	F'	Encrypted data file
5	K	A place holder for alphabet used for cryptosystem
6	μ	A place holder that holds part of the proof generated by CSP
7	β	A symmetric key
8	α	A symmetric key
9	ω	A place holder that holds part of the proof generated by CSP
10	τ	Parameter used in storing cumulative proofs
11	m_i	This refers to the i^{th} data block
12	dk	It is a key use in decrypting encrypted data
13	pk	A key use in encrypting data
14	γ	A place holder for accumulative proofs
15	M	Message file
16	v	Encryption exponent
17	w	Decryption exponent
18	Q	A place holder that holds results of computation
19	R	A place holder that holds results of computation

Signature and Parameters Generation

$$MU_{prk_SigGen}(F') \rightarrow Sig_{prk}(H(R)) \dots \dots \dots (3)$$

$$SO_{prk_SigGen}(F') \rightarrow Sig_{prk} \Phi \dots \dots \dots (4)$$

$$proofGen(chal, F', Sig_{prk}(H(R)), \Phi) \rightarrow (P) \dots (5)$$

Equation (3) is run by the mobile user and takes the hash value of root of merkle hash tree as input and generates its signature as metadata. Equation (4) is executed by surrogate object that used m_i as input from each data block of sealed file F' that is sealed, and generate the signature collection $\Phi = \{\pi_i\}$ on $\{m_i\}$. The cloud storage service provider run equation (5) that uses the generated verify challenge

information “chal” as data to be inputted by surrogate object, the data file F stored in the cloud storage, the signature for the metadata and the signature \emptyset , and output the information as proof of possessing the data P.

3.3 Privacy and Confidentiality

To ensure privacy and confidentiality of mobile users' data file stored on cloud, data are encrypted using cryptographic method and encoded before they are being stored in the cloud storage.

$$Encap_{ek}(F) \rightarrow F'$$

Encap is the encrypting parameter. The SO utilises this parameter to encrypt mobile user's raw data file with seal key ek to encode the encrypted data file with erasure codes using the rotated reed Solomon code. It generate sealed encrypted and encoded file F' . The cryptosystem consist of an alphabet K containing all characters that can be used in messages such as numerals, punctuation marks, letters, and blank spaces and so on. It also consists of bijections as:

$$\alpha : K \rightarrow P \text{ and } f : P \rightarrow P.$$

For this study, the author assumed that messages are all in form of alphabet

Algorithm 2: Encryption Operation

Encrypt ()
 Begin ()
 Set $f(x) = 2x \text{ mod } 26$
 $K = A - Z$
 $\alpha(A) = 0, \alpha(B) = 1, \dots, \alpha(Z) = 25$
 perform f
 inverse α
 end.

Encoding of data and description

$$EncodAl(F') \rightarrow F''$$

The encoding service provider invokes the above parameter and runs the encoding algorithm to encode the encrypted mobile user data file into F'' using erasure codes.

Decoding of data and description.

To decode the encrypted and encoded mobile user's data file stored in cloud storages, the data file needed to be retrieved from the cloud storage before decoding operation algorithm is invoke. The parameter used in decoding is as follows:

$$K = \{A, B, \dots, Z\}, \text{ taking } P = Z_{26}$$

$$\alpha(A) = 0, \alpha(B) = 1, \alpha(C) = 2, \alpha(D) = 3, \dots, \alpha(Z) = 25$$

The author now uses $f(x) = 2x \text{ mod } 26$ to encrypt the message THESIS WORK.

The procedure is as follows:

T	H	E	S	I	S
	W	O	R	K	

Using equation (4);

Where T = 19, H = 7, E = 4, S = 18, I = 8, S = 18, W = 22, O = 14, R = 17 and K = 10

Therefore, $\alpha = 19, 7, 4, 18, 8, 18, 22, 14, 17$ and 10

And $f = 12, 12, 18, 10, 10, 10, 18, 2, 8$ and 6.

$\alpha^{-1} = M, M, S, K, K, K, S, C, I$ and G. So the

message THESIS WORK has the corresponding ciphertext MMSKKKSCIG.

The algorithm of the encryption operation is as follows.

$$DecodAl(F'') \rightarrow F'$$

Decryption of data and description

To decrypt the data file, the decryption algorithm is run by the surrogate object. The above data file can be decipher by repeating the same procedure as in encryption algorithm using the inverse($f(x)$) function as follows:

$$Decap_{dk}(F') \rightarrow F$$

$Decap_{dk}$ is the decrypting data parameter. The mobile user or data owner submit the request for database access or to extract the data file F through its corresponding SO, the SO retrieved the corresponding sealed encrypted and encoded data file F' from storage service provider, decodes and decrypt the file to obtain F using decryption key (dk), and forward F to mobile user.

3.4 Integrity Verification

$$verify(P, chal) \rightarrow (0/1)$$

This phase begins with the mobile user who issues a command to the SO to send a verification challenge to cloud storage to ascertain the

correctness of outsourced file. Before challenging, the SO use public key pk to identify signature or marks generated and included in the data file. If verification fails, deny generating 0; else, return 1. According to proof, storage service provider who acts as a prover, compute the proof of verification and send the proof generated to the verifier (SO). The verifier then verifies the proof sent by the prover and sends the outcome to the mobile user. The algorithm for the integrity verification is shown in Algorithm 3.

Algorithm 3: Integrity Verification

Chal()
SO send chal → CS
CS uses chal
CS gen proof($sig_{pk}(H(R)), \phi$) → SO
SO ver($\alpha\beta\pi$) → CS, where chal = (0/1)

3.5 Data file retrieval

Symmetric keys have already been exchanged using Diffie-Hellman protocol. The mobile user request for data file retrieval through its respective surrogate object. The SO then requests for data file retrieval from the cloud storage. Mobile user send the decryption key (dk) enciphered by symmetric session key to SO. The SO then runs the decryption algorithm to decrypt the data file as in $Decap_{dk}(F') \rightarrow F$.

Algorithm 4: Data File Retrieval

The algorithm for data file retrieval is as follows:
Retrieval ()
Mobile user request(F) → SO
MU → dk
SO request(F') → CS
CS send F' SO
SO decrypt(F') → MU

3.6 Update of Records

$PerfUpdate(F, \phi, Update) \rightarrow (F', \phi', Pupdate)$

PerfUpdate is the perform update of data parameter. The algorithm is executed by server. It

uses file F as input, signatures Φ , and information request “update” from users. It gives the result F’ as an updated file, updated signatures Φ' and a proof DoUpdate for computation performed.

Data Modification:- To modify data in stored data file (F’), mobile user initiates the request for data modification by sending its SO by calling the update data function $PerfUpdate(F, \phi, Update) \rightarrow (F', \phi', Pupdate)$ specifying the data block to be updated such as update m_i , with m'_i illustrated using MHT in Figure 2.

To modify $h(n_2)$, replace the block by $h(n'_2)$.

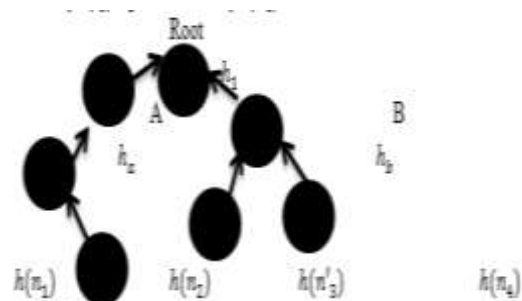


Figure 2: Data Modification

Figure 2: Data Modification

Data Inclusion:- Data are inserted into the data file through the data inclusion operation. The data file has to be retrieved from the cloud storage before data are inserted into some marked locations in data file F. mobile user sends the update request: $update = (I, i, m^*, \omega^*)$ and send the request message to the cloud storage. The cloud storage runs the $DoUpdate(F, \phi, update)$ algorithm and sends the output of the update to the mobile user through its SO. The data inclusion operation is illustrated using MHT as in Figure 3.

Include $h(n^*)$ after $h(n_2)$

cloud4 and assume all the clouds are federated. User's data in the cloud can be replicated and distributed as data cache and stored in SO of cloud 1, 2, 3, and 4.

SO executes data access request submitted by its corresponding mobile user if it holds data cache otherwise MU_1 to MU_5 request are forwarded to SSP1 in cloud1, MU_6 to MU_7 requests are forwarded to SSP4 in cloud2, MU_8 to MU_{10} requests are forwarded to SSP5 in cloud3 and MU_{11} to MU_{13} requests are forwarded to SSP7 in cloud4 through a distributing manager located in MSS-1.

Furthermore, whenever SSP1 receives more than two requests at a time from say MU_1 to MU_5 , it allowed only two requests to be processed. Maximum load is set to two. Since cloud1 to cloud4 are federated, the distributing manager distributes the request of MU_1 to MU_5 to the SSP across the boundaries of cloud2, cloud3, and cloud4 or to the SO. The resources of the clouds are shared or used on rental basis to satisfy the MU_1 to MU_5 requests. The same goes for all other requests in federated mobile cloud storages.

To provide seamless data access even when cloud1 is located out of the communication bound due to unforeseen reasons, cloud1 immediately consult with cloud2, cloud3 and cloud4 since data are replicated and synchronized well in advance by SO in the other clouds. Therefore, the service requests directed to cloud1 are distributed to cloud2, cloud3, or cloud4 to proceed with the request disconnected in cloud1. At the end of the processing of the request, the requested data is retrieved and forwarded to the mobile users' corresponding SO for effective and efficient delivery.

4. RESULTS

To evaluate the performance of the author's proposed system, a simulation experiment was conducted in the Aneka Cloud through the Aneka Management Studio 3.0 on a system having an intel Pentium (R) Dual Core processor running at 2.20 GHz, 4 GB installed memory (RAM), and on a 32bit windows7 operating system. The Aneka cloud is a simulator and a .Net based platform. The service management platform (Aneka Cloud) provides basic parameters and functions required for mobile nodes in cloud platform and infrastructure that comprises of service authentication, verification (auditing), and possession, authorization, forming and sending messages. The Aneka cloud configuration wizard was used to create the needed simulation prototype for the proposed system. The Aneka cloud

contained infrastructure comprising of; repositories, machine credentials, installed machines, uninstalled machines and access denied machines, and the container which comprises of; master containers, provisioned containers, quarantined containers and orphan containers. The node selection module, data store settings, security module, advance setting module and other modules such as service setting modules are used to create 20 nodes in the Aneka cloud for the evaluation of the proposed system. It also used to configure, monitor and manages the nodes created in the simulation prototype. The various nodes are sets into different clouds; cloud 1, cloud 2, cloud 3 within the federated clouds. Specifically, the advance setting module in the Aneka cloud configuration wizard was used to create the various nodes and clouds as the simulation prototype. A surrogate object is a software entity hosted on mobile support station and used in this study as a service provider. It is created in the various nodes using data base setting and management tools.

The simulation in Aneka cloud setup was done and tested using the size of data files as stated in table 2. The data size for table 2 and table 3 was obtained from the data files of the mobile user. The measurement is from the memory of the computer system used.

Table 2: The data size and storage space

l	Data File Size (MB)	Storage Space
2^8	8	8
2^9	16	16
2^{10}	32	32
2^{11}	64	64
2^{12}	128	128
2^{13}	256	256
2^{14}	512	512
2^{15}	1024	1024
2^{16}	2048	2048
2^{17}	4096	4096
2^{18}	8192	8192

Table 3: The computation speed and storage space

l	Data File Size (MB)	Storage Space	Verifier Comp. Time	Prover Comp. Time (μ s)
2^8	8	8	501	700
2^9	16	16	1020	1,200
2^{10}	32	32		2000
2^{11}	64	64	2034	2800
2^{12}	128	128	2501	3,400
2^{13}	256	256	3032	4000
2^{14}	512	512	3501	4200
2^{15}	1024	1024	4056	4800
2^{16}	2048	2048	4512	5,100
2^{17}	4096	4096	5030	5,900
2^{18}	8192	8192	5506	7,000

From the table above, it shows that larger values for l requires more time for computation due to the complexity nature of the algorithm and the parameters involved. To deploy a verification system using the proposed work, time taken to access a data file must be considered to be as minimal as possible having as low access time as 1 Megabyte to be read in as small time as some few nanoseconds.

Figures 5 show the time taken for a verifier to computes the proof of possession send to it by the prover and verify that the data are correct. The computational time depends on the size of the data. In figure 6, measurement of time taken by the prover to computes and send responds to the verifier that it is actually in possession of the data file is shown.

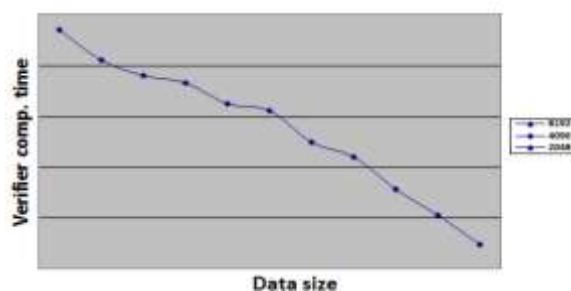


Figure 4.1: Time taken by a verifier to compute and verify a data size of a respond

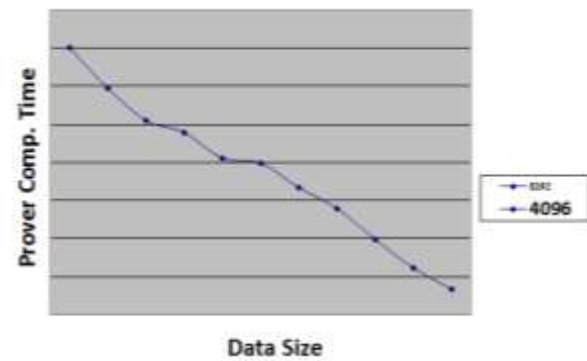


Figure 4.2: Time spent by a prover to compute and send proof of data possession

Figure 5 and Figure 6 shows scalability in the computational time of both the prover and the verifier. From Figure 5 and Figure 6, it shows that the higher the size of the data file the longer time it took the prover and the verifier in computing and providing proof of data possession and verification of the provided proof of possession.

5. CONCLUSION

The computing and storage intensive operations that ensure privacy and confidentiality are encryption, decryption, encoding and decoding which are handled by surrogate objects to relieve resource constraint mobile devices of the burden of this workload. Data are encrypted and encoded before storing them. The computation of proof of data possession and data verification is achieved through cloud storage (prover) and surrogate object. Mobile users need to register and assign with unique identity and surrogate object. A verification mechanism is built on top of the surrogate objects to ensure the integrity of stored data in distributed mobile cloud environment. The verification mechanism is built on communication services to provide an authentication structure to efficiently and securely prove that the stored data file is undamaged and correct. The author's system also ensures confidentiality of data as data files are encrypted and encoded with erasure codes. The cumulative proof ensures that all the data file in the federated clouds are verified. The security analysis of the system shows that using the Diffie-Hellman key exchange protocol communication between two entities are more secured. Simulation was conducted and the result shows considerable and acceptable scalability measures of the proposed solution.

The review of existing system has revealed that mobile devices have limited resources to perform computationally intensive operations in cloud computing environment. Mobile users who

employed federated data management are faced with the problem of insecurity and data integrity. Most of the proposed security framework for cloud federation and traditional cloud are not for federated mobile cloud computing. The author in this study presented federated data management system that is secured in a distributed mobile cloud. Surrogate object was used to act on behalf of mobile devices to perform computationally intensive operation and also to calculate and store through its cache, the cumulative proofs.

6. ACKNOWLEDGMENTS

Our thanks staff of Federal University of Technology Minna, Niger State, whose guidance has made this research work a success.

7. REFERENCES

- [1] N. K. Abdul, K.M.L. Mat, U.K. Samee and A.M. Sajjad, Towards secure mobile cloud computing: A survey. *Future Generation Computer Systems*, doi:10.1016/j.future.2012.08.003. Available at www.elsevier.com/locate/fgcs. Retrieved on June 21st, 2014.
- [2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Kartz, A. Konwinski and M. Zaharia, Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/Eecs-2009-28, Eecs Department, University of California, Berkeley, 2009.
- [3] S. Amritha, and S. K. Saravana, Threshold proxy re-encryption scheme and decentralized erasure code in cloud storage with secure data forwarding. *Journal of Computer Engineering*, 9 (5), 27-31, 2013.
- [4] H. Canepa, and D. Lee, A virtual cloud computing provider for mobile devices, *ACM workshop on Mobile Cloud Computing & services Social Networks and Beyond*, MCS '10, San Francisco, USA, 2010.
- [5] Deswarte, Y., Quisquater, J. J., and Saidane, A. (2003). Remote Integrity Checking. *Proceedings of Conference on Integrity and Internal Control in Information Systems*. IICIS' 03, San Francisco, USA.
- [6] Hsueh, S. C., Lin, J. Y., and Lin, M. Y. (2011). Secure cloud storage for conventional data archive of smart phones, in: *Proceedings of 15th IEEE International Symposium on Consumer Electronics, ISCE '11*, Singapore.
- [7] Itani, W., Kayssi, A., and Chehab, A. (2010). Energy-efficient incremental integrity for securing storage in mobile cloud computing, in: *Proceedings of Internal Conference on Energy Aware Computing, ICEAC '10*, Cairo, Egypt.
- [8] Jia, W., Zhu, H., Cao, Z., Wei, L., and Lin, X. (2011). SDSM: a secure data service mechanism in mobile cloud computing, in: *Proceedings of IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPs*, Shanghai, China.
- [9] Jian, Y., Haihang, W., Jian, W., Chengxiang, T., & Dingguo, Y. (2011). Provable data possession of resource constraint mobile devices in cloud computing. *Journal of Networks*, 6 (7), 1033-1040.
- [9] Kumar, K., and Lu, Y.H. (2010). Cloud computing for mobile users: Can offloading computation save energy? *IEEE Journal of Computer*, 43 (4), 51–56.
- [10] Peng, X., and Yanping Z. (2013). CS-Mobile: A cloud-based distributed storage middleware for mobile devices. *International Journal of Smart Home*, 7(1), 87-98.
- [11] Ravimaran, S., Muhammed, M., and Sharief, A. (2012). Federated data management in distributed mobile cloud. *European Journal of scientific Research*, 86 (4), 482-492.
- [12] Sotomayor, B., Montero, R. S., and Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, 13 (5), 14-22.
- [13] Zhibin, Z., and Huang, D. (2011). Efficient and secure data storage operations for mobile cloud computing, *IEE Journal of Computer*, 46 (7), 49-55.