Biomedical Research 2018; Special Issue: S88-S94 *ISSN 0970-938X*
www.biomedres.info

Provable dynamic auditing in mobile cloud computing for secure storage of e-health data.

Suguna M*, Mercy Shalinie S

Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, India

Abstract

The recent developments in mobile applications have met every need of the user. In spite of increasing usage, the smart mobile system face restrictions like resource scarcity, mobility and frequent disconnections. The emerging Mobile Cloud Computing (MCC) infrastructure exploits the services of Cloud Service Providers (CSP) by gathering the local data storage and computation to remote servers and thereby relieving the overhead incurred by mobile clients. There is a wide range of applications in MCC like m-Learning, m-Healthcare, etc. Though the restrictions are relaxed, the clients no longer have control over the privacy of outsourced data. A safe and secure remote access to highly sensitive private data like e-Health records has received the focus of research. It needs to ensure the integrity of outsourced health data along with its privacy preserved. The resource constrained mobile devices act as a thin client for the cloud server access and hence the computation for ensuring the proof of correctness for secure data storage proposed by existing solutions becomes an overhead. The proposed Provable Dynamic Data Auditing Protocol (PD-DAP) involves a trusted Third Party Auditor (TPA) who is in charge of blockless verification of data without retrieving the private health data. The usage of bilinear pairing and Merkle Hash Trees guarantees blockless verification by TPA. Also the proposed PD-DAP supports secure dynamic operations by the mobile user with the help of TPA. Hence, it saves the mobile user's computational resource, thereby achieving cost-effectiveness to gain trust in the Cloud Storage Server (CSS).

Keywords: Mobile cloud computing, e-Health, Privacy, Auditing, Bilinear pairing, Merkle hash tree.

Accepted on May 23, 2017

Introduction

Mobile devices play a vital role in people's routine life as a convenient and comfortable communication tool which can be used anywhere and anytime. However, they face some restrictions like CPU potential, memory, residual power and others like mobility and security [1]. Mobile device generally experience delay in its service due to limited resources. The mobile user can take up the advantages of upcoming cloud computing paradigm to improve its performance. Based on National Institute of Standards and Technology (NIST) definition [2], cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The start of such infrastructure widely reduces the cost of storage, computation, application hosting and delivery. Efficient and scalable data storage services are offered by CSPs to the users at a minimal cost compared to other traditional approaches [3]. With the incorporation of mobile computing and cloud computing arise a new paradigm shift from conventional mobile computing and wireless communications services to Mobile Cloud Computing (MCC) and services over wireless Internet in the real world [4]. Mobile cloud computing addresses an emerging infrastructure where both data storage and data processing happens outside the device. The local management of vast stored data in a smart mobile device is transferred to CSS, thereby relieving the storage and cost burden of the mobile user.

MCC provides a fast and better service provisioning to e-Health record in applications like remote monitoring and home care systems [5]. A large scale of medical data is involved in such applications. The overhead of storage and maintenance is delegated to the CSS. Fast access of e-Health data by smart mobile devices ensures the ease of anytime-anywhere-accessibility for the end users enabling them to reduce significantly the interruption in their day-to-day activities through remote monitoring.

However, the MCC standard for data hosting service presents new security challenges. The main concern is outsourcing private e-Health data to external cloud storage as it may lead to data and privacy loss [6]. As a solution to the stated problem, most of the online service providers, such as Dropbox, promise

to encrypt their client's data and store it at heavily secured locations. Nevertheless, recurring news reports [7] about security breaches, unauthorized access to private files, and other forms of data leakage have stated the trust in cloud service providers. It is of vital security measure for the mobile users to be equipped with certain provisions with which remote data correctness can be ensured periodically with minimum local computation overhead. The provisioning of data integrity and privacy in the proposed PD-DAP protocol finds application in the medical field where the patient record can be stored, updated and retrieved from cloud in a safe and secure manner that ensures the privacy of the e-Health data.

Related Work

Many recent research works have concentrated to propose a solution for the verification of the remote data possession without having the data locally stored. Deswarte et al. [8] and Filho et al. [9] have proposed protocols to check the correctness of files stored on remote server using RSA-based hash functions. The client can do the verifications for any number of times with the same data, which is not possible in other hash-based approaches. But, the computational complexity increases exponentially at the server side for entire file. This has been overcome by the two approaches Provable Data Possession (PDP) and Proofs of Retrievability (POR). Verification of data possession at untrusted server by the client without retrieving the entire file is the PDP approach proposed by Ateniese et al. [10]. It is limited by the issue of convincing the client that a file is verified in spite of partial or total data missing. Juels et al. proposed POR scheme [11] that enables archiving or backup service to produce a short proof data which is used by the client to audit the file and to retrieve the same. This scheme depends on the preprocessing steps the user has to perform before outsourcing the file to the CSS. The POR scheme is limited as it does not support data dynamic operations. The extended dynamic auditing protocol by Wang et al. [12] ensures privacy and also supports batch auditing. Also, the verifier is assumed to be stateless i.e., the verifier need not store the data block information in between the audits. Wang et al. proposed a dynamic auditing protocol [13] to enable verification of dynamic operations on data stored in remote storage servers. Bilinear aggregate signature was used for multiple auditing tasks and it also supports blockless verification i.e., the data blocks are not needed for the verifier and made impossible to access. Zhu et al. proposed Cooperative Provable Data Possession (CPDP) scheme [14] which supports batch auditing for multiple cloud. A preliminary PDP scheme IPDP [15] proposed by Zhu et al. concentrates on probabilistic query and verification at regular frequency to improve the performance of audit scheme. The secure dynamic auditing protocol proposed by Yang et al. [16] supported batch auditing for multiple owners and multiple clouds. Many proposed schemes considered various system and security models and have solved the problem of data integrity checking under different metrics. Even though widespread research is there, a security protocol for MCC storage auditing is not well proposed. All the existing methods for auditing incur heavy computational cost for the verifier, which may become a performance bottleneck for a mobile device. Since resource constrained mobile clients does not necessarily have the computation resource for verification, the proposed work introduces the public Third Party Auditor (TPA) which is an unbiased trusted server. The TPA helps to verify the semi trusted Cloud Storage Server on behalf of mobile device. The proposed PD-DAP protocol has two parts, Storage auditing and Data Dynamic operations.

The paper is organized as follows. The basic definitions and nomenclature used in the proposed work are presented in Section III. In the next section, the proposed provable dynamic data auditing protocol is discussed. The performance analysis based on security aspects and the performance evaluation of the proposed protocol is presented in Section V. Finally, the conclusion is given.

Preliminaries

The basic definitions of the proposed PD-DAP protocol are described as follows.

Basic definitions

Bilinear map: A bilinear map is a map e*:* $G \times G \rightarrow G_T$, where G and G_T are multiplicative groups of prime order with the following properties [17]:

(i) Computable: there exists an efficiently computable algorithm for computing *e*

(ii) Bilinear: for all $h_1, h_2 \in G$ and $a, b \in Z_p$, $e(h_1^a, h_2^b = e(h_1, h_2)^{ab}$

(iii) Non-degenerate: $e(g, g) \neq 1$, where g is a generator of G.

Merkle hash tree (MHT): It is a well-studied authentication structure [18], which is proposed to efficiently and securely prove that the elements are intact and unchanged. It is a binary tree with the hashes of the data values as leaves. While MHT is used in data values authentication, we apply the same to authenticate both the data values and also the positions of data blocks. The leaf nodes are considered in left-to-right sequence so that any data value can be uniquely identified and hence the root computation in the MHT.

System model

There are three major entities in the proposed electronic health record cloud storage system as shown in the Figure 1. Users collect the health related data from the health care devices. Medical technician is a clinical person who treats the patients. Both user and medical technician are referred as data owner and associated computing facilities. Data owner is in charge to store, retrieve and update the health records to CSS for storage and maintenance. The computing facilities are mainly smart mobile devices which are carried out. The storage servers managed by the CSP provisions storage infrastructure for the user's data and maintain the same.

The data owner depends on the trusted third party auditor, which has the skill to assess and detect risk of storage services given by CSP and specifies a dishonest entity upon request by the data owner.

Security requirements

In this work, we try to meet the following security requirements for Provable Dynamic Data Auditing Protocol (PD-DAP).

Data confidentiality-The data stored at CSS should not be a meaningful context for unauthorized parties who attack the cloud storage.

Data integrity-The highly sensitive data stored at third party storage servers protected from modification, insertion or deletion of data by unauthorized parties must be provable.

Data privacy-Even though TPA is trusted, the auditing process delegated must be a blockless verification i.e. only metadata is available for verification process.

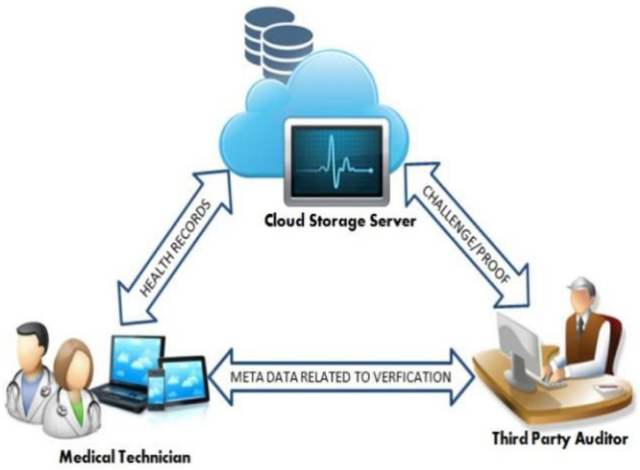

Figure 1. *System model for provable dynamic data auditing protocol.*

Provable Dynamic Data Auditing Protocol (PD-DAP)

The proposed storage auditing protocol is used by the data owner of electronic health records to ensure the data privacy of data stored at untrusted CSS. The TPA audits the outsourced file content on behalf of mobile data owner. Since the local data of patient health record are highly sensitive and private to the user, the main challenge of the protocol is ensuring data privacy against the auditor. In public file storage, the auditor may get access to the data by recovering it from the proof. Also the verifier may break the encrypted file through some process and will be able to succeed. The proposed protocol solves this data privacy problem by generating the encrypted proof using the bilinear pairing, such that the TPA cannot retrieve the data. The plain e-Health data file F is encrypted and then it is divided into n data block splits m_1, m_2, ..., m_n where $m_i \epsilon Z_p$ and p is a large prime. Each data split can be dynamically modified or deleted by the mobile user. Each and every data split m_i is divided further into k data blocks such that $F=\{m_{ij}\}$, $i \epsilon (1, n)$ $j \epsilon (1, k)$. The data tags are generated for every block split m_i $i \epsilon (1, n)$. The size of the data block is fixed by the selected parameter. Let a bilinear map $e: G \times G \rightarrow G_T$ and a hash function: H: $(0, 1)^* \rightarrow G$ be viewed as a random oracle model [19]. The proposed PD-DAP have two protocols for auditing and dynamic data update: (i) provable storage auditing protocol and (ii) dynamic data auditing protocol. The construction of these protocols follows in the subsequent sections.

Construction of provable storage auditing protocol

The remote data possession auditing protocol involves three phases: Initial set up, challenge-proof generation and proof verification. The algorithms involved in these phases are as follows:

Initial setup: The initial setup involves three computation steps at user side: key generation, tag generation and signature generation. The detailed description of the initial setup phase is as follows.

KeyGen (g) $\rightarrow$ *(pk, sk)*: At the mobile user side, the security parameter g taken as the only input and chooses a random $\alpha \epsilon Z_p$ and sets the secret key sk=α. It computes the *pk* using Equation 1 and sets as the public key v.

$$pk=g^{\alpha} \rightarrow (1)$$

TagGen (sk, F) $\rightarrow$ *(Γ, S (R))*: For the given file $F=m_1, m_2, ..., m_n$, with every data block split m_i $(i=1,..., n)$, the tag generation algorithm generates data tag t_i using the secret key *sk*. It selects k random numbers $u_{i\epsilon\{[1,k]} \epsilon G$, and computes the data tag t_i using Equation 2 and Γ the tag set using Equation 3.

$$t_i = \left(H(m_i) . \prod_{j=1}^{k} \mu_j^{m_{ij}} \right)^{sk} \rightarrow (2)$$

$$\Gamma=\{t_i\}_{i[1, n]} \rightarrow (3)$$

Next the user generates a root R based on the Merkle Hash Tree (MHT), where the leaf nodes are ordered set of hashes [20] of the file block splits $H (m_i)$ $i \epsilon [1, n]$.

Sign *(R, sk)* $\rightarrow$ *S (R)*: The data owner signs the root R using the secret key *sk* and generates *S (R)* using Equation 4.

$$S (R)= (H (R))^{sk} \rightarrow (4)$$

The data owner sends $\{F, \Gamma, S (R)\}$ to the storage server and gets relieved from the burden of local storage. Also the user sends F_{info} to the auditor for generating the Challenge *Chal*.

Challenge-proof generation: The challenge data *Chal* is generated by the TPA and sent to the storage server. Upon receiving the challenge information for a particular file, CSS computes the Proof data P and responds to the auditor by sending the same. The algorithms involved in this process challenge generation and proof generation are described here.

Challenge (F_{info}) $\rightarrow$ (*Chal*): The TPA generates the Challenge by taking the metadata of the file to be downloaded. The storage server is challenged with the generated information.

The auditor selects a random q element subset $Q=\{s_1, ..., s_q\}$ of set $\{1, ..., n\}$ such that $s_1 \leq ... \leq s_q$. The algorithm computes a random element $r_i \in Z_p$ for every $i \in Q$. Also, the position of block splits those to be verified are mentioned in *Chal*. The challenge is generated using Equation 5 and sent to the storage server.

$$Chal\{(i, r)\}_{s_1 \leq i \leq s_q} \rightarrow (5)$$

Proof (*F, Γ, Chal*) → (*P*): On receiving *Chal* the server computes μ_j for each $j \in [1, k]$ and σ using Equations 6 and 7 as follows.

$$\mu_j = \sum_{i=s_1}^{s_q} r_i m_{ij} \in Z_p \rightarrow (6)$$

$$\sigma = \prod_{i=s_1}^{s_q} t_i^{r_i} \rightarrow (7)$$

The cloud server also generates the set $\{\varphi_i\}_{s_1 \leq i \leq s_q}$, which are the list of sibling nodes from the specified leaves to root R of the MHT. Proof P generated using Equation 8 by the server is then sent back to the auditor.

$$P = \left\{\{\mu_j\}_{j \in [1,k]}, \sigma, \{H(m_i), \varphi_i\}_{s_1 \leq i \leq s_q}, S(R)\right\} \rightarrow (8)$$

Proof verification: The proof verification phase is executed by the auditor. The detailed description of the verification phase is described here.

VerifyProof (pk, Chal, P) → (0/1): he auditor on receiving the proof from the storage server computes the data R^* with {H (mi), i}$_{s1 \leq i \leq s\ q}$. The updated R^* is sent to the data owner for signing. After receiving the signed root from data owner, TPA authenticates the proof by verifying the bilinear property using Equation 9.

$$e(S(R^*), g) = e(H(R^*), g^{\alpha}) \rightarrow (9)$$

On verification, if the proof fails to satisfy the property, then he rejects by giving output as 0. Else, the hash challenge H_{chal} and Integrity Proof IP is computed using Equations 10 and 11 from the proof *P* contents received from the claimant (server) for verification.

$$H_{chal} = \prod_{i=s_1}^{s_q} H(m_i)^{r_i} \rightarrow (10)$$

$$IP = \prod_{j=1}^{k} \mu_j^{\mu_j} \rightarrow (11)$$

The verifier checks the correctness by the bilinear property verification using Equation 12 as follows:

$$e(\sigma, g) = e((H_{chal}.IP), v) \rightarrow (12)$$

Based on the conditional check, verifier outputs '1' if proved to be true to show that the content is not modified and the correctness of data are maintained by the server. Else, outputs '0'. Thus the provably secure verification is done.

Construction of provable dynamic data storage protocol

The fully dynamic data operations proposed in this section can maintain updation of outsourced data like modification, insertion and deletion efficiently. Before file updation or deletion, the File *F*, data tags Γ and the signed root *S (R)* are assumed to be already generated and maintained by cloud storage server. Figure 2 represents the dynamic data operations in a Hash Tree example.

Data block modification: In the cloud storage data maintenance, one of the frequently used functions is data modification. It refers to the process of replacing existing specified data split m_i, by a new one m'. As the data owner needs to update a data split, it generates the new tag t' and updated root R' by calling *UpdateFile (op, i, m', sk)* algorithm. Data owner chooses *op=M* for data split modification. Then, it prepares and sends the message {*F, Γ, i, m', t'*} to the server and the updated F_{info} to the auditor. The server on receiving the modify message computes ExecuteUpdate (*F, Γ, i, m', t'*). At the server side,

i). the block m_i is replaced with m' and the server generates F'

ii) the tag t_i is replaced with t' and the server generates Γ';

iii) the server replaces the respective node in MHT with *H (m')* and compute the updated R';

iv) the updated proof P_{update} is generated using Equation 13 and the server sends the same to the auditor for verification

$$P_{update} = (i, \{H(m_i)\}_{i\,[1,n]}, H(m')\ R') \rightarrow (13)$$

On receiving the updated proof the auditor first constructs the MHT and then replaces the node $H(m_i)$ with *H (m')*. Then, generates root R' and compare with received one to authenticate the modification at the server. The updated root is sent to the data owner for signing. The received signed root from him/her is sent for updation at the server by the auditor.

Data block insertion: The data file structure does not change in data modification. But in data insertion, new block split m' gets inserted after the specified block split m_i and hence there will be changes in the file structure. As the data owner needs to insert a new data split, it generates the new tag t' and updated root R' by calling *UpdateFile (op, i, m', sk)* algorithm. Data owner chooses *op=M* for data split insertion. Then, it prepares and sends the message {*F, Γ, i, m', t'*} to the server and the updated F_{info} to the auditor.

The server on receiving the insert message computes *ExecuteUpdate* (*F, Γ, i, m', t'*) and performs the following actions:

i) Insert the block m' after mi and compute F';

ii) Insert t' to the tag set and compute Γ';

iii) Add *H (m')* as a new leaf node and generate new root *R'*

iv) Generate the updated proof using Equation 14 and send it to the auditor for verification.

$Pupdate=(i, \{H(m_i)\}_{i[1,n]}, H(m'), R') \rightarrow (14)$

On receiving the updated proof the auditor authenticates and verifies the updated Proof.

Data block deletion: Upon receiving the data split deletion request of the particular block from the user, the server ensures all updates in server storage and gets authentication and verification by auditor similar to the data insertion function.

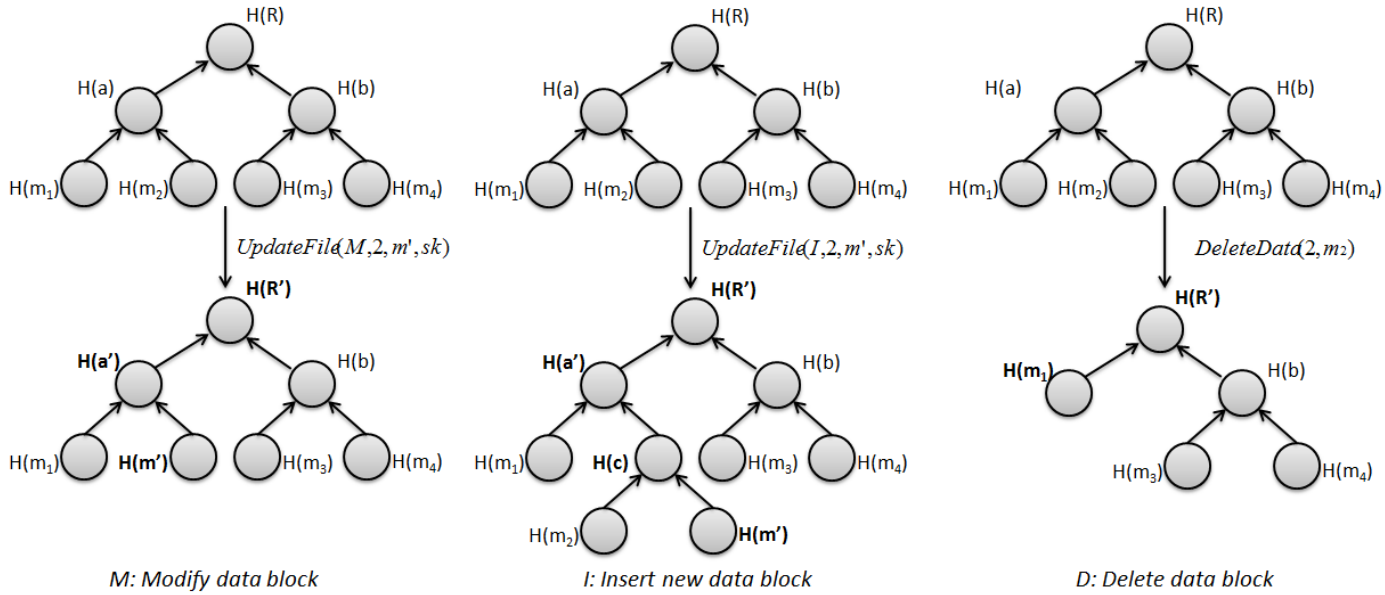

Figure 2. *Merkle hash tree example for: a) data block modification, b) data block insertion and c) data block deletion.*

Table 1. *Comparison of existing remote data integrity verification protocols and proposed PD-DAP scheme.*

Auditing Protocols	Computation		Communication	Privacy preserving	Dynamic data operations	Probability of detection
	Storage server	TPA				
PDP[10]	O(t)	O(t)	O(1)	Yes	No	1-(1-p)t
Audit[12]	O(t log n)	O(t log n)	O(t log n)	Yes	Yes	1-(1-p)t
IPDP[14]	O(tk)	O(t+k)	O(t+k)	Yes	Yes	1-(1-p)tk
DPDP[18]	O(t log n)	O(t log n)	O(t log n)	No	No	1-(1-p)t
CPDP[25]	O(t+k)	O(t+k)	O(t+k)	No	No	1-(1-p)tk
Proposed PD-DAP	O(t)	O(t)	O(t)	Yes	Yes	1-(1-p)tk

n: Number of data splits in the file; t: number of challenged data splits; k: number of blocks in every data split; p: Probability of block corruption.

Performance Analysis

Security analysis

The proposed work security audits the outsourced e-Health data by the mobile data owner at remote cloud storage with the help of a Trusted TPA. The MHT used for data integrity verification has the hashes of the data in the leaves of the tree structure. In this method of tree construction, all the blocks have to be sent by the server for authentication which will lead to overhead for verifier. Meanwhile, the privacy of user's data has to be maintained. Hence the original data files must not be sent to the verifier for the auditing process. To overcome this privacy issue, blockless verification is performed. Here, instead of data block authentication, the block tags are authenticated in the verification by TPA which is signed using Boneh-Lynn-Shacham (BLS) signatures.

The BLS signature method used is provably secure with an assumption of intractability of Computational Diffie-Hellman problem (CDH). The CDH assumption is, given g, ga, gb ϵ G, for unknown *a, b* ϵ Z_p, it is hard to compute g^{xy} [20]. If the signature scheme is existentially unforgeable and the CDH in bilinear groups is hard, no adversary against the soundness of the proposed scheme could cause to accept in a proof-of-retrievability protocol instance with non-negligible probability by the verifier, except by responding with correctly computed values. Given a fraction of the *n* blocks of an encoded file *F*,

recovering the original file *F* is possible with all but negligible probability.

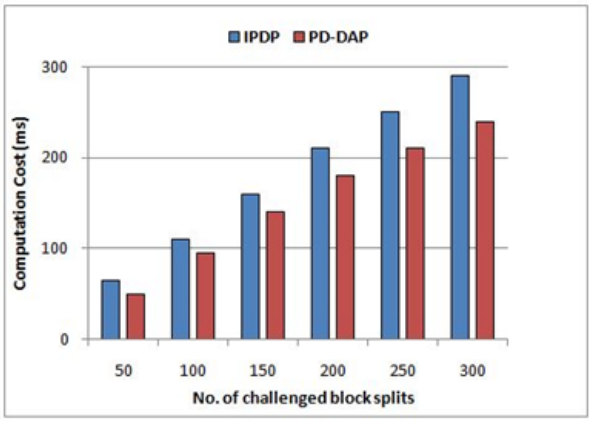

Figure 3. *Comparison of computation cost incurred by auditor.*

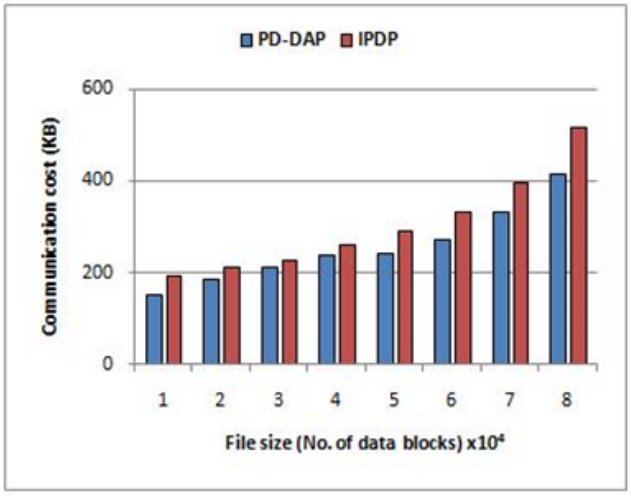

Figure 4. *Comparison of communication cost incurred by server.*

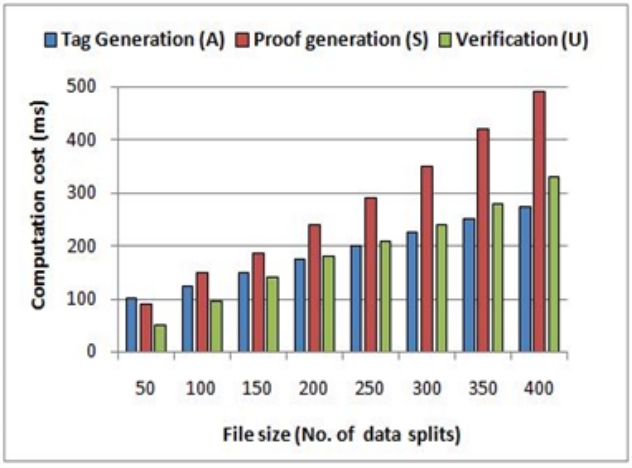

Figure 5. *Comparison of computation time of respective functions by Server (S), Auditor (A) and User (U).*

Performance evaluation

Generally storage auditing protocols incur high computation, communication and storage overhead. At storage server side, it has to store the necessary proof needed for every file during the verification process. For the auditor, the metadata from the data owner has to be stored, Challenge data to be generated and the proof sent by the server has to be verified. In the existing PDP schemes, the data owner acts as the verifier. But a mobile user cannot afford verification process because of its inherent resource problems. Hence an external trusted TPA is mitigated with the auditing process. Dynamic data auditing in MCC has not been solved effectively as in cloud computing. Hence the comparison of the proposed dynamic auditing protocol is made with audit protocols in general Cloud Computing shown in Table 1. The probability of a block getting corrupted with high detection probability is verified as $1-(1-p)^{tk}$. Let $q=(1-p)$, then the probability becomes $1-q^{tk}$. The table shows the computation and communication order of time for the proposed method which outperforms the existing methods. For experiment, we implemented our proposed scheme as follows. The auditing mechanism involves Cloud storage server, trusted third party auditor and the user who access to mobile devices. The Cloud computing process is implemented in Amazon EC2 with an instance of Linux with 8 core processor and 64 GB memory. The trusted TPA is simulated as a desktop machine running on Linux with 3.4 GHz Intel i7-3770 CPU and 32 GB memory. The mobile user is simulated as a mobile device which runs in Android 5.0.2 version running in 1.2 GHz quad-core Qualcomm Snapdragon 400 processor with 1 GB of RAM. We have used Java based Pairing Based Cryptography (jPBC) library version 0.5.12 for coding our scheme. The MNT elliptic curve with field size 159 bits is used. The Android studio environment is used for applications at user side manipulations. All the following results specified are the simulation results of mean over 20 trials.

Initially the process starts with the tag generation step at data owner side. Apart from the file, the respective metadata is also stored in the cloud. The metadata alone is stored at TPA for auditing purpose. Figure 3 shows the computation time incurred at the verifier side for the existing protocol IPDP [15] and the proposed method PD-DAP. The graph shows clearly that the proposed method consumes less time compared to existing, because the computation time at verifier includes only proof generation and challenge verification process only. But in the existing method the verifier is the actual data owner and hence the File blocks generation and the Tag generation process also performed by the verifier itself. Also, some of the verifier proof generation steps discussed in the existing method are delegated to the server here. This has increased the computation cost of storage server (Figure 3). The communication cost for sending the proof generated by the server to auditor for verification is represented in Figure 4 and compared with the existing scheme. Since the communication time of the auditor and the mobile user are negligible compared to the cost of Server communication it is not taken under discussion. The process of tag generation is performed by the user and Figure 5 compares the processing time at auditor and the server with the data owner job. As the file size increases, the tag generation time increased linearly. But the tag generation cost is high compared to the computational cost for

proof generation and verification for minimum file size. This is because the splitting and tag generation is heavy and it is efficient to use the protocol for large size files and not for smaller sized files uploaded from mobile devices. Thus the heavy overhead at the user side computation specified by existing method is reduced for mobile devices using the proposed method.

Conclusion

In this work, the problem of dynamic verifiability of e-Health data and auditing using TPA under user's trust domain is explored. The proposed Provable Dynamic Data Auditing Protocol (PD-DAP) for resource constrained mobile devices in the Mobile Cloud Computing architecture aims at ensuring the prominent features of the public integrity verification process for e-Health data with minimum overhead for the mobile user. In order to ensure the computation overhead limiting at the mobile user side, the public verifiability task is delegated to the Trusted Third Party Auditor (TPA) which in turn introduced data privacy issue of e-Health data. The blockless verification process designed in the proposed method ensures that the private data of the users are not revealed to the TPA during the audit process. Thus the Trusted Third Party Auditor (TPA) audits the service of Cloud Storage without the need for the data locally. Hence it is provably secure. Also, the usage of MHT enabled short signatures reduces the storage complexity at the verifier side. The detailed experimental analysis shows the validation for the performance of our proposed scheme.

References

1. Satyanarayanan M. Mobile computing: the next decade. Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond (MCS) 2010.
2. Mell P, Grance T. NIST definition of cloud computing. Nat Inst Stand Technol 2009.
3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. Comm ACM 2010; 53: 50-58.
4. Huang D, Zhang X, Kang M, Luo J. Mobicloud: A secure mobile cloud framework for pervasive mobile computing and communication. Proceedings of 5th IEEE International Symposium on Service-Oriented System Engineering 2010.
5. Tong Y, Sun J, Chow SS, Li P. Cloud-assisted mobile-access of health data with privacy and auditability. IEEE J Biomed Health Inform 2014; 18: 419-429.
6. Velte T, Velte A, Elsenpeter R. Cloud computing: a practical approach. McGraw-Hill, Inc. (1st edn.) New York NY USA 2010.
7. Soghoian C. How dropbox sacrifices user privacy for cost savings. Slight paranoia blog 2011.
8. Deswarte Y, Quisquater JJ, Saidane A. Remote integrity checking. Proc of Conference on Integrity and Internal Control in Information Systems (IICIS03) 2003.
9. Filho DLG, Baretto PSLM. Demonstrating data possession and uncheatable data transfer. IACR ePrint Archive 2006.
10. Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. Proc 14th ACM Conf Computer Comm Security (CCS07) 2007; 598-609.
11. Juels A, Kaliski BS. Pors: proofs of retrievability for large files. Proc 14th ACM Conf Computer Comm Security 2007; 584-597.
12. Wang C, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for data storage security in cloud computing. Proc IEEE Infocom 2010; 525-533.
13. Wang Q, Wang C, Ren K, Lou W, Li J. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distr Sys vol. 2011; 22: 847-859.
14. Zhu Y, Hu H, Ahn G, Yu M. Cooperative provable data possession for integrity verification in multi-cloud storage. IEEE Trans Parallel Distr Sys 2012; 23: 2231-2244.
15. Zhu Y, Wang H, Hu Z, Ahn GJ, Hu H, Yau SS. Dynamic audit services for integrity verification of outsourced storages in clouds. Proc ACM Symp Appl Comp 2011; 1550-1557.
16. Yang K, Xiaohua J. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans Parallel Distr Sys 2013; 24: 1717-1726.
17. Wang Q, Wang C, Li J, Ren K, Lou W. Enabling public verifiability and data dynamics for storage security in cloud computing. Cryptol ePrint Archive Rep 2009; 281: 2009.
18. Merkle RC. Protocols for public key cryptosystems. Proc IEEE Symposium on Security and Privacy, Oakland, California, USA 1980.
19. Shacham H, Brent W. Compact proofs of retrievability. Adv Cryptol Asia Crypt 2008; 90-107.
20. Boneh D, Lynn B, Shacham H. Short signatures from the Weil pairing. Adv Cryptol Asia Crypt 2001; 2248: 514-532.

*Correspondence to

Suguna M

Department of Computer Science and Engineering

Thiagarajar College of Engineering

India