SSICC: Sharing Sensitive Information in a Cloud-of-Clouds

Rick Lopes de Souza, Hylson Vescovi Netto, Lau Cheuk Lung, Ricardo Felipe Custodio
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianopolis, Brazil
Email: {rick.lopes, hylson.vescovi, lau.lung, custodio}@inf.ufsc.br

***Abstract*—The need to share and manipulate sensitive data is a challenge for most content providers using the cloud for storage. The objective of this work is to propose an architecture to ensure a distributed access control to secure sharing of sensitive electronic documents in the cloud. This paper will explain the architecture of the model, details of the protocols, implementation, and analysis on security, usability, and performance. The main features of the proposed model are: the use of Identity-Based Encryption, byzantine fault tolerance, reliable integrity check, secure user revocation, low complexity in the management of cryptographic keys and secure sharing of sensitive electronic documents.**

Keywords-Privacy; Sensitive; Cloud.

I. INTRODUCTION

The possibility of decreasing investment in infrastructure for data storage become real through the use of cloud computing. A user can send, edit, save, and access his data in the cloud using any device. These characteristics are attractive for many corporations due to economic facilities provided. The cloud service provides savings in acquisition and configuration of hardware and software, making corporations to pay only for what they actually use.

The advantages of using cloud services bring with them the problem of ensuring the security of stored data. Typically, cloud services allocate more than one application or data in the same structure and, therefore, problems such as attacks coming from other corporations or even employees of cloud providers become real. Today, one of the main concerns of companies adopting cloud services is ensuring security and privacy of sensitive data. It is not difficult to find cases of data theft, as the case of SalesForce in 2007, where criminals have succeeded in stealing information about customers, such as e-mails and addresses [1]. In order to allow users to control the access to their sensitive data stored in a public cloud, a suitable access control is required. The access policies must restrict data access to only those intended by the data owners. These rules must be guaranteed by the cloud providers. If the system is allocated in just one cloud provider, the data owners have to assume that the cloud providers are trusted and they will prevent the access from unauthorized users. Thus, storing unencrypted data in public clouds can expose sensitive information to a malicious third party. To provide the necessary security, the cloud provider must not have access to unencrypted data. Therefore, the action of sharing a sensitive document to groups or roles is still considered a challenge.

Many techniques have been proposed to address these problems, but there is not a singular better solution. The traditional public key encryption techniques that uses public certificates with a public key infrastructure can not resolve all the problems involved in the sensitive document sharing and it is infeasible when the system grows in number of users. Another technique that can be used is called Identity Based Encryption (IBE) [2] and it was first introduced by Shamir in 1985. The IBE consists in three entities: sender, receiver, and a trusted authority. The sender of a message specifies an identity (a set of characters) such that only a receiver that matches that identity can decrypt and read it. The trusted authority is responsible for the authentication process and to supply the necessary private keys. These private keys are directly tied to the users identities.

The IBE trusted authorities can access the users private key and, for this reason, the IBE technique faces some resistance to be implemented in some systems. To overcome this limitation, multiple trusted authorities can be used, such that any of them can posses the users private key. The work of Aniket [3] proposes a multi authority to IBE systems. In his work, a set of authorities execute a modified algorithm of Joint Feldman Distributed Key Generator (JF-DKG) [4] to generate the master secret in a distributed manner. Users can contact a subset of authorities to request a part of the private key and then, reconstruct the entire user private key.

Another IBE limitation is the user revocation. Since one user has an identity 'ID', if the related key is compromised, there is no way to generate another private key to the same identity 'ID' without affecting other users. There are many proposes to overcome this problem. One of the proposals is to use Attributed Based Encryption [5]. This technique was proposed by Amit Sahai and Brent Waters and it posses the concept that every user has some attribute in a specific company or entity. The private keys are tied to users attributes that are given to then by a singular trusted authority. Recent works [6], [7], [8], [9], [10], [11] are based on multi-authorities but they still have some problems with user revocation due to re-encryption of documents and rekeying. Zhou et al. [12] proposes a modified algorithm of Boneh and Fraklin Identity Based Encryption (BF-IBE) [13] proposing a role based encryption. Our work has a similar idea of the Zhou et al. paper, using the concept of roles and groups to identify the users and encrypt sensitive documents.

This paper has as main focus to propose an architecture

 ISBN: 978-1-61208-319-3

to ensure secure sharing of sensitive documents in a cloud of clouds. To make it possible, this paper uses Identity Based Encryption with multi-authorities to manage the cryptographic keys. The main contribution of this paper is an architecture to store sensitive documents and share it in public or private clouds. This architecture makes practical the maintenance of users and groups, by using multi-authorities with IBE, secret sharing and erasure codes. This work touches the areas of Secure User Revocation, Reliable Integrity Check, Backward and Forward Secrecy, Byzantine Fault Tolerance, Storage Economy, and Efficient Document Sharing.

This paper is organized as follows. In Section 2, we present the math preliminaries and the related work. Section 3 presents the architecture of the proposed solution. Section 4 presents a detailed system model. In Section 5, we will analyze the security and performance of the system model. Finally, Section 6 presents the conclusions of the study and discussion of possible future works.

II. Preliminaries and Related Work

A. Bilinear Pairing

Let G_1, G_2 be additive groups and G_T a multiplicative group, all of prime order p. Let $P \in G_1, Q \in G_2$ be generators of G_1 and G_2 respectively. A pairing is a map: $e : G_1 \times G_2 \rightarrow G_T$ for wich the following holds:

- **Bilinearity:** $\forall a, b \in Z_p^* : e(aP, bQ) = e(P, Q)^{a,b}$
- **Non-degeneracy:** $e(P, Q) \neq 1$
- **Computability:** There is an efficient algorithm to compute $e(P, Q)$ for any $P \in G_1$ and $Q \in G_2$.

The IBC protocols used in this work have a special form of pairing called symmetric pairing which has: $e(P, Q) = e(Q, P)$. The security of the techniques used in this work are based on the Decisional Diffie-Hellman (DDH) problem and the Decisional Bilinear Diffie-Hellman (DBDH) problem. Our work rely on the assumption that no probabilistic polynomial-time algorithms can solve the DDH and DBDH problem with non-negligible advantage.

B. Distributed Key Generation

We use a completely distributed key generation based on the Joint-Feldman distributed key generator (JF-DKG), the distributed key generation proposed by Aniket [3]. The JF-DKG requires a number $n \geq 3t + 1$ nodes to run correctly, being the simplest and most efficient DKG. We use the BF-IBE technique due to it's simplicity of setup and methods. In BF-IBE Setup, a Private Key Generator (PKG) generates private keys (d) for clients using their known identities (ID) and master-key (s). We seek an (n, t) distributed key generation over an elliptic curve group G of order q and generator U, where n are the total nodes involved and $t + 1$ honest nodes are sufficient to generate it correctly. Let F(z) = $a_0 + a_1 z + ... + a_t z^t \in Z_q[z]$ be the current shared polynomial and s = a_0.

The protocol proposed by Aniket uses an improved version of Feldman Verifiable Secret Sharing (Feldman VSS) algorithm to generate in a distributed manner the master-key. It has a bulletin board that generates the public parameters of BF-IBE Setup, publishes the values and than initializes the A_k and A_{ik} values to zero, for i = 1, ..., n and k = 0 ... t, where $A_k = a_k U$ and $A_{ik} = a_{ik} U$. The master key is set to zero. The nodes initiates the Feldman VSS. After t + 1 nodes run their protocols successfully, the distributed shares are considered safe. These nodes are called qualified nodes. Here, we denote their set as ∂. The bulletin board computes and broadcasts the coefficients A_k (for k = 0... t) for the implied shared polynomial F(z).U as $A_k = \sum_{P_j \in \partial} A_{ik}$. After verifying the new A_k values, nodes send confirmation signatures to the bulletin board. On receiving t + 1 confirmation signatures, the A_k values are finalized. Each node then computes their secret share as $s_i = \sum_{P_j \in \partial} s_{ji}$. Mode details can be obtained in the work of Aniket [3].

C. Private Key Extraction

To extract the private key in a distributed way, the client must contact the nodes and send a specific ID. After receiving the ID, the PKGs $P_i \in O$ authentic and authorize the user and then returns a private-key share $S_i H(ID)$ over a secure channel. The H represents a hash function $H : (0, 1)^* \rightarrow G^*$. Upon receiving t + 1 correct shares of her private key, the client can reconstruct the private key D_{id} as $D_{id} = \sum_{P_i \in O} \lambda_i s_i H(ID)$, where the Lagrange coefficient $\lambda_i = \prod_{P_j \in O, j \neq i} \frac{j}{j-i}$.

D. Related Work

Recent works propose the use of privacy services [14], [15] to address the privacy documents storage problem, as well as some others works [16], [17] propose not to encrypt the files and just split it and then send it to different cloud providers. These works try to solve the problems involved to store sensitive documents in cloud, however, they can not provide all the necessary features to make a secure sharing. The work of Itani et al. [14] didn't provide a sharing scheme and a fault tolerance system. If the privacy service stops, the client can not encrypt or decrypt the files. The propose of Padilha [16] use the technique of homomorphism to modify the privacy parts of sensitive documents through the use of addictive functions, however, the techniques to provide secrecy using total homomorphism are theoretical and lack of pragmatic implementations. Practical implementations of total homomorphism to secrecy systems are still open issues research, therefore, are not applicable in the present circumstances of corporations.

Other line of work is to use Atributed Based Encryption (ABE), where each user receives credentials from the trusted authorities which releases the access to the sensitive documents. Some works as [6], [7], [8], [9], [10], [11] proposes the use of multi-authorities to generate the user private key, avoiding the cloud key escrow. These papers also works with a distributed manner of providing the attributes, supplying the needs of identity confidentiality. However, the solutions that involve ABE have the drawback of revocation. Once the key that encrypts one document is revoked, the system needs to re-encrypt the sensitive documents and then have to re-keying. Another peculiarity in almost all these related works is that they don't support fault tolerance. The proposes are based on multi-authorities just do extract the private keys, nevertheless, they store the files on just one cloud provider. If this cloud

 ISBN: 978-1-61208-319-3

provider fail for any reason, the user will not have access to his sensitive documents. Another issue due to these facts is that if an attacker can compromise one cryptographic key and have access to the cloud provider, he can obtain all the sensitive data.

Other line of work made by Bessani et al [18] uses concepts that we are going to use: symmetric encryption, Shamir secret sharing and erasure optimal code. Nevertheless the paper does not propose any mechanism to share the necessary keys to guarantee the integrity of the shares. It simply admits that there is a mechanism to share keys, however, this is one of the main challenges in sharing sensitive documents using encryption. Another downside is the read data algorithm that does not check integrity with a public key to verify the hash of the parties. If the cloud provider is a malicious attacker, it can modify the parts and provide false hashes for integrity check, thus compromising the system.

Our work resembles to the Zhou et al [12]. His work proposes the use of a modified IBE to enforce role-based access control, providing the possibility to encrypt a file to a single user or roles. The propose is efficient if it does not have many revocation operations, otherwise it will present high complexity. Zhou work does not solve the key escrow problem. The system administrator can access users keys in the extract operation. Another issue is the single point of failure and if the administrator fails, undertake part of the revocation system.

Based on earlier research, this article identified the main challenges to share sensitive documents in a secure manner. We have proposed an architecture to tackle all these challenges involving the use of IBE to provide the following features: Secure User Revocation, Reliable Integrity Check, Backward and Forward Secrecy, Byzantine Fault Tolerance, Storage Economy, and Efficient Document Sharing.

III. System Architecture

There will be two main types of components in the architecture for the solution. The first component represents public clouds that host the application servers and store sensitive documents. The second component is the end users that possess mechanisms to encrypt and decrypt sensitive data, as well as mechanisms to securely store cryptographic keys used.

The client side is responsible for editing, encrypting and decrypting files. This side also aims to define who are the custodians of sensitive data to be encrypted. These custodians are delimited by access rules specified for each application. The storage servers must be hosted on different cloud providers. The main features that are needed: the distribution of access control through state machine replication and the distributed manner of application servers spread across nodes.

Figure 1 illustrates how the architecture of secrecy works. Briefly, the architecture works as follows: N nodes (cloud providers) will conduct first a system setup, so they can share a private key. Each node shares access control and provides an application to users to share sensitive documents. Users who wish to share confidential documents must encrypt the document locally, perform operations of breaking and encoding then the document is sent and stored in the cloud providers. The document owner must inform who may have access to sensitive data. A user who wishes to obtain a document must authenticate to the cloud providers to obtain the required data to join, decode, and decrypt the document. In this work, it is used an abstraction called Data Block that store data about the sensitive documents. The Data Block contains five items: ID, part of the encrypted symmetric key, signed hash of the encrypted key, part of the encrypted document and signed hash of the encrypted document.

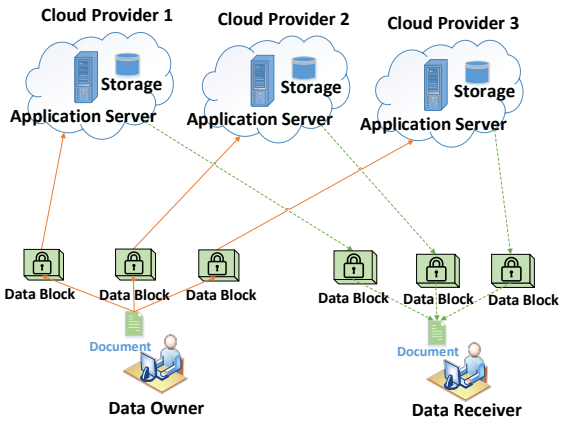

Figure 1. System Model - A Data Owner send N datablocks to cloud providers to share it. The Data Receiver must obtain a pre-defined number of datablocks to recompose the file.

The access control must be implemented in a distributed form. A cloud provider should not impersonate a user to obtain the other parts of the secret. Each cloud provider will posses an access control list and it will make the authentication and authorization of each user. As we consider the cloud as a semi-honest entity, the access control is considered reliable.

We focus on the architecture to provide distributed access control to documents and we will not discuss in details the access control. This proposal has the following assumptions:

- Every cloud provider has an access control;
- There is no concurrency for documents access;
- There is no concurrency for control access;
- The cryptography algorithms are resistant collision;
- The cloud providers are semi-trusted (They will behave correctly before the user requests, but can be curious and see the data stored);

IV. System Model

We use the secret sharing scheme to integrate the confidentiality and the availability. Since all encrypted keys are splitted in N pieces (N is the total number of cloud providers), the user needs a minimum of M parts (M is provided in the setup of the system) to recombine the encrypted key. In fact, we reuse the access control of an application to control which readers are able to access the stored data. We also use the mechanism of information-optimal erasure code [19], enforcing an economy in the cloud providers to store different versions of the same document. Otherwise, the costs would increase by a factor of n

 ISBN: 978-1-61208-319-3

if it was necessary to replicate it to n clouds. Each share has a reduced size by a factor of $\frac{n}{f+1}$ [20], considering f the number of faulty servers. Here we consider that the minimum number of parts to recover a Shamir secret sharing is directly related to the redundant parts of the information-optimal erasure code. For example, if we consider a total number $N = 4$ of nodes and a minimum number $M = 2$ of nodes to recover some key, the erasure code will consider a total of $T = N - M$ and a redundant number of $R = M$. It will be always necessary to gather at least two parts of the total to recombine the parts.

We use the BF-IBE scheme to encrypt symmetric keys building a specific ID containing the following information: Name of the Document, Group of Custodians, and Document Version. We use this specific ID due a set of characteristics that are necessary to share sensitive documents. The Name of the Document in ID is to generate a different key to each document stored in the cloud. The Group of Custodians is to limit the access control of the document. As we will be reusing the access control system, this group of custodians will be used to authenticate and allow access to sensitive documents. For every different group of custodians, there is a different key. The version number and the others elements has the intention to control the access for different versions of the documents and to guarantee the backward and forward secrecy. The Hess Signature technique [21] is used to sign the parts of encrypted symmetric key and encrypted data to guarantee the integrity. In this work, we use the same key pair to encrypt and sign, facilitating the key sharing and management.

Before users can use the system, it must be made the Private Key Generators (PKGs) setup. The protocol is described in the section II-C and it is responsible for the distributed generation of IBE master secret. After this step, reusing the system access control, the users can share sensitive documents through the use of the following methods: Write Data and Read Data. The Write Data algorithm is data owner responsibility and it encrypts, encode, and send all the encrypted and signed parts to the cloud providers. The Read Data algorithm is executed by the data receivers that wants to visualize the document content.

The Write Data (Figure 2) authenticates the user and asks the access control for the document metadata (line 4). The new document to be stored will have the last version found (line 5) plus one (line 6). A symmetric key is randomly created (line 7) to encrypt the document (line 9). An ID for the document is defined (line 10) and a public key based on this ID is created (line 11) using the BF-IBE technique. The symmetric key is encrypted with the public key (line 12) and then splitted into shares using Shamir secret sharing scheme (line 13). The encrypted document is encoded using an information-optimal erasure code algorithm (line 14), reducing the size of data that will be stored in the cloud providers. A private key is created, based on the ID (line 15) based on the procedure of the Session II-B. For each part of splitted encrypted key and encoded encrypted data, we provide hashes (lines 17 and 18) and then sign it (lines 19 and 20) using the Hess Signature scheme. A data block will be created, gathering all necessary information to store the document (line 21). The data block will be sent to the cloud providers (line 22) and the entry of this storing is sent to the access control (line 25).

The Read Data (Figure 3) first authenticate the user and

Algorithm 1 WRITEDATA($fileName, fileGroup, user$)

1: total $\longleftarrow n$
2: redundant $\longleftarrow m$
3: data_ver $\longleftarrow 0$
4: mt $\longleftarrow queryMetadata(fileName, fileGroup, user)$
5: data_ver $\longleftarrow max(mt[i].ver : 0 \leq i \leq n-1)$
6: new_data_ver $\longleftarrow data\_ver + 1$
7: ks $\longleftarrow generateSymKey()$
8: data $\longleftarrow openFile(fileName)$
9: e_data $\longleftarrow E(data, ks)$
10: id $\longleftarrow fileName + "/" + fileGroup + "/" + new\_data\_ver$
11: pubk_id $\longleftarrow generate\_pub\_key(id)$
12: e_ks $\longleftarrow E(ks, pubk\_id)$
13: enc_ks[0 ..n-1] $\longleftarrow split(e\_ks, total - redundant, total)$
14: enc_data[0 ..n-1] $\longleftarrow encode(e\_data, total - redundant, redundant)$
15: privk_id $\longleftarrow generate\_priv\_key(id)$
16: **for** $(0 \leq i \leq total - 1)$ **do**
17: data_hash $\longleftarrow H(enc\_data[i])$
18: ks_hash $\longleftarrow H(enc\_ks[i])$
19: data_signed_hash $\longleftarrow Sign(data\_hash, privk\_id)$
20: ks_signed_hash $\longleftarrow Sign(ks\_hash, privk\_id)$
21: dataBlock $\longleftarrow (id, enc\_ks[i], enc\_data[i], data\_signed\_hash, ks\_signed\_hash)$
22: ack $\longleftarrow sendStoreMessage(cloud_i, dataBlock)$
23: **if** $(ack =' ok')$ **then**
24: dataControlBlock $\longleftarrow (id, user, fileGroup)$
25: $sendAccessControlMessage(cloud_i, dataControlBlock)$
26: **end if**
27: **end for**

Figure 2. Write Data Algorithm - Executed by the data owner to encrypted and send the sensitive information to the cloud of clouds.

asks the access control for the document metadata (line 3). The last version is chosen (line 4) and the ID is composed (line 7). A key pair is generated from the ID using BF-IBE (lines 5-6) and the requests for data are started (line 8). The data blocks are requested from clouds (line 9), where each gotten data block is verified about its signatures and hashes using Hess Technique (line 12). After getting the necessary data blocks (line 19), the pieces of the encrypted document are joined using erasure code (line 23), the symmetric key is restored using secret sharing (line 24) and decrypted (line 25), and finally, the original document is decrypted (26).

V. IMPLEMENTATION

We have implemented the protocols in Java and C++. The implementation is divided into three parts: The PKGs Setup, The Distributed Access Control and the Algorithms of Write and Read Data. The PKGs Setup was implemented by Aniket in his work [3] using C++ and modifying the JF-DKG protocol. The work of Aniket uses the pairing-based cryptography library [22]. The necessary communication protocols to make a distributed access control were implemented in java using sockets to send and receive messages. The Write and Read Data algorithms was implemented in C++

 ISBN: 978-1-61208-319-3

Algorithm 2 READDATA($fileName, fileGroup, user$)

1: total $\longleftarrow n$
2: redundant $\longleftarrow m$
3: mt $\longleftarrow queryMetadata(fileName, fileGroup, user)$
4: data_ver $\longleftarrow max(mt[i].ver : 0 \leq i \leq n-1)$
5: privk_id $\longleftarrow generate\_priv\_key(id)$
6: pubk_id $\longleftarrow generate\_pub\_key(id)$
7: id $\longleftarrow fileName + "/" + fileGroup + "/" + data\_ver$
8: **while** $(i \leq n-1)$ **do**
9: temp_dataBlock $\longleftarrow cloud_i.getDataBlock(id)$
10: temp_eks $\longleftarrow temp\_dataBlock.return\_enc\_ks()$
11: temp_edata $\longleftarrow temp\_dataBlock.return\_enc\_data()$
12: rt $\longleftarrow Verify(temp\_dataBlock.ks\_signed\_hash_i, temp\_dataBlock.data\_signed\_hash_i, temp\_eks, temp\_edata, pubk\_id)$
13: **if** $(rt = true)$ **then**
14: $enc\_ks[i] \longleftarrow temp\_eks$
15: $enc\_data[i] \longleftarrow temp\_edata$
16: **else**
17: **return** $ERROR$
18: **end if**
19: **if** $(i > redundant - 1)$ **then**
20: $Break$
21: **end if**
22: **end while**
23: e_data $\longleftarrow decode(enc\_data, total - redundant, redundant)$
24: e_ks $\longleftarrow combine(enc\_ks, total - dedundant, total)$
25: ks $\longleftarrow D(e\_ks, privk\_id)$
26: **return** D(e_data, ks)

Figure 3. Read Data Algorithm - Executed by the data receiver to obtain and decrypt the sensitive that from the cloud of clouds.

using the following libraries: pbc library [22] to make the pairing based operations, the gfshare library to make Shamir secret sharing [23], jerasure to encode and decode files using information-optimal erasure code [19] and OpenSSL to make some traditional cryptographic operations [24].

Our implementation was made by modules and it was a proof of concept that could be optimized to achieve better results. We have used the AES algorithm to encrypt files with 128-bit key size. We also have used SHA-1 [25] for cryptographic hashes, BF-IBE [13] to encrypt symmetric keys, Hess Signature to sign the parts of the encrypted keys and data and we used a total of four nodes with a redundant number of two nodes (this number is tied to the PKG Setup of Aniket, which requires a minimum of 3f + 1 nodes, being f + 1 the quorum to recompose the master secret).

VI. ANALYSIS

This section presents quantitative and qualitative analysis about the algorithms and the security of the system. Due to the lack of space in this paper, we can not extend our analysis and make a deeper review of the benefits. We will focus on the main benefits and we will make a superficial analysis about the algorithms and what they solve.

A. Security Analysis

One of the problems pointed out by SP800-144 [26] is the vulnerability of internal attacks and lack of legal support in cases of intrusion due to the geographical location of the servers. To solve these problems, this paper proposes the use of distributed PKGs using the JF-DKG protocol to generate the master key without any of the trusted entities have ultimate control of it. By using this scheme, two parameters are set: t and N where N is the total number of PKGS and t represents the minimum number of parts that must be collected to recover a private key a user. Thus an agent discovering malicious need a total of t shares to recover the secrets, therefore decreasing the chances of success in an attack. Using the protocols of Aniket, we achieve a numerous benefits as: Distributed PKG Setup, Forward Secrecy, Availability of the Public Key, Periodic master-key modification, Secret Share Renewal, Secret Share Recovery, Group Modification and Threshold Modification. All the computation costs of these operations are $O(n^2)$.

To maintain the confidentiality of information, it is recommended to use more parameters to identify the public key of the IBE. One part of the solution is not to bind a key per user and rather bind user groups with documents, thus having a semantic key. This work proposes the use of identifiers containing access rules concatenated with the name of the document and a version of the same. The access rules are checked by the PKGs through access controls that must be done in a distributed and reliable manner . The document name binds the public key to a specific document. The version makes for each modification of this document to have a different public key. Thus, a member who was part of a group and obtained the private key to decrypt a document in a version X does not obtain a consequent deciphering key to a document with version $X + 1$ if he is no longer part of the group. If a user has already obtained the private key, this had access to document content in this version. Therefore, there is no need to reencrypt a document content that was already exposed. If a user has not obtained the private key and get out of a particular group, it will no longer have access to the private key, because the access control check if the member complies with the rules imposed on the classified document.

Using distributed PKGs, secret sharing and erasure codes we can ensure fault tolerance. Thus, this paper proposes protocols based on verifiable shared secret quorum, thus increasing the rigor of the checks widespread parts. The Byzantine fault tolerance and availability are provided with the property that a total of $3f + 1$ PKGs only f may fail and therefore should always consult a total of $f + 1$ responses to verify that the majority of obtained final secrets are equal.

We also use the Hess Signature Scheme to guarantee the integrity of the digest obtained from the encrypted symmetric key and data parties. The use of the same key pair to encrypt the symmetric keys and to sign the digests improve our architecture due to the key granularity and the simple and secure manner to share the IBE keys. The security of the Hess Signature follows from the security of the generic scheme in the random oracle model and is based on the Diffie Hellman Problem in the domain of the used pairing. Using IBE we also achieve chosen ciphertext security (IND-CCA) that is the

 ISBN: 978-1-61208-319-3

standard acceptable notion of security for a public encryption scheme.

TABLE I. COMPARISON OF DIFFERENT SOLUTIONS AND SOLVED PROBLEMS.

Procedure	**This Work**	Ruj et al. 2011	Zhou et al. 2011	Bessani et al. 2011
Custody of Keys	✓	✓	X	✓
User Revocation	✓	✓	✓	✓
Fault tolerance	✓	✓	X	✓
Reliable Integrity Check	✓	X	X	X

The work proposed here can solve some cloud privacy sharing problems in a simple and safe manner using a set of cloud providers, splitting techniques and as main tool the identity-based encryption. Table I compares the results of the related work to this article.

B. Performance Analysis

With our implementation we could evaluate the performance of the algorithms. We have chosen to evaluate the performance of the algorithms of Write Data and Read Data, mentioned in the Section IV. We did a sequence of simulations using the implementation with different file sizes. We have started with 1Kbyte files to 524288Kbytes (512 MBytes). The tests were executed in a computer with the following characteristics: Processor Intel I3, 4GB RAM with the operational system Linux Ubuntu. We have executed one week each algorithm and evaluated the standard deviation. Among 1 Kbyte and 16384 Kbytes times were unstable, surpassing the 5% standard deviation. However, this is because the size of the files were small. But even files with 1 Kbyte to 16384 Kbytes kept coming times of 300ms. From 16384 Kbytes files, the time began to grow linearly as doubling the size of the file. As shown in the graph of Figure 4, it can be noted that the increase was linear, demonstrating the stability of the algorithms for different file sizes.

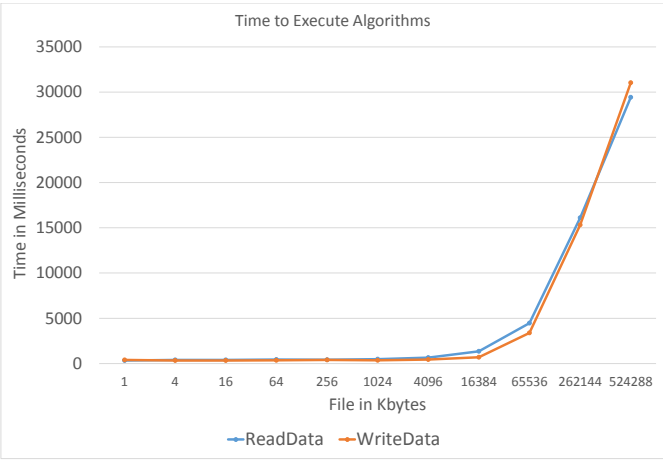

Figure 4. Graph illustrating the performance of Write Data and Read Data Algorithm.

Depending on the application that are using the algorithms, most of the files will be encrypted and sent to the clouds without major performance problems. For example, in applications sharing text files as PDFs with sizes less than 16Mbytes have good performance around 300ms, as in the case of courts of justice (e-justice and e-health). The cases where there is a greater need to share documents, such as medical imaging, where there is a need for high resolutions and video files starting with the 16Mbytes, have satisfactory performance due to file sizes.

VII. CONCLUSION AND FUTURE WORK

This paper proposed an architecture to share sensitive documents in clouds. Because the new trends of computerization of data, large corporations and government entities are increasingly investing resources in cloud computing that has demonstrated economically viable for data storage. These stored documents require encryption support to ensure the confidentiality of sensitive data that should not be accessible to unauthorized third parties.

The proposed model is based on the use of identity-based encryption and provides a secure manner to share sensitive documents between data providers and consumers of information. The main benefits that this work provides are: secure sharing of sensitive documents,reliable integrity check, reducing the custody of cryptographic keys and fault tolerance.

As future work, we suggest to improve the implementation to achieve better performance results and improve the work to provide identity privacy. It is also suggested the validation of protocols in a formal way so that they can be used in practice for large companies and government entities.

REFERENCES

[1] A. Greenberg, “Cloud computing’s stormy side,” Forbes Magazine, Last Visited: December, 2013. [Online]. Available: http://www.forbes.com/2008/02/17/web-application-cloud-tech-intel-cx_ag_0219cloud.html

[2] A. Shamir, “Identity-based cryptosystems and signature schemes,” in Advances in cryptology. Springer, 1985, pp. 47–53.

[3] A. Kate, Y. Huang, and I. Goldberg, “Distributed key generation in the wild.” IACR Cryptology ePrint Archive, vol. 2012, 2012, p. 377.

[4] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, “Secure distributed key generation for discrete-log based cryptosystems,” in Advances in Cryptology, EUROCRYPT, 99. Springer, 1999, pp. 295–310.

[5] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in Advances in Cryptology–EUROCRYPT 2005. Springer, 2005, pp. 457–473.

[6] S. Ruj, A. Nayak, and I. Stojmenovic, “Dacc: Distributed access control in clouds,” in Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on. IEEE, 2011, pp. 91–98.

[7] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, “Privacy preserving cloud data access with multi-authorities,” in IEEE INFOCOM, 2013, pp. 2625 – 2633.

[8] K. Yang, X. Jia, and K. Ren, “Dac-macs: Effective data access control for multi-authority cloud storage systems.” IACR Cryptology ePrint Archive, vol. 2012, 2012, p. 419.

[9] K. Yang, Z. Liu, Z. Cao, X. Jia, D. S. Wong, and K. Ren, “Taac: Temporal attribute-based access control for multi-authority cloud storage systems.” IACR Cryptology ePrint Archive, vol. 2012, 2012, p. 651.

[10] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, “A ciphertext-policy attribute-based encryption scheme with constant ciphertext length,” in Information Security Practice and Experience. Springer, 2009, pp. 13–23.

 ISBN: 978-1-61208-319-3

[11] S. Kamara and K. Lauter, “Cryptographic cloud storage,” in Financial Cryptography and Data Security. Springer, 2010, pp. 136–149.

[12] L. Zhou, V. Varadharajan, and M. Hitchens, “Enforcing role-based access control for secure data storage in the cloud,” The Computer Journal, vol. 54, no. 10, 2011, pp. 1675–1687.

[13] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in Advances in Cryptology - CRYPTO 2001. Springer, 2001, pp. 213–229.

[14] W. Itani, A. Kayssi, and A. Chehab, “Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures,” in Dependable, Autonomic and Secure Computing, 2009. DASC’09. Eighth IEEE International Conference on. IEEE, 2009, pp. 711–716.

[15] S. Pearson, Y. Shen, and M. Mowbray, “A privacy manager for cloud computing,” in Cloud Computing. Springer, 2009, pp. 90–106.

[16] R. Padilha and F. Pedone, “Belisarius: Bft storage with confidentiality,” in Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on. IEEE, 2011, pp. 9–16.

[17] Y. Singh, F. Kandah, and W. Zhang, “A secured cost-effective multi-cloud storage in cloud computing,” in Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on. IEEE, 2011, pp. 619–624.

[18] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, “Depsky: dependable and secure storage in a cloud-of-clouds,” in Proceedings of the sixth conference on Computer systems. ACM, 2011, pp. 31–46.

[19] J. S. Plank, S. Simmerman, and C. D. Schuman, “Jerasure: A library in c/c++ facilitating erasure coding for storage applications-version 1.2,” University of Tennessee, Tech. Rep. CS-08-627, vol. 23, 2008.

[20] M. O. Rabin, “Efficient dispersal of information for security, load balancing, and fault tolerance,” Journal of the ACM (JACM), vol. 36, no. 2, 1989, pp. 335–348.

[21] F. Hess, “Efficient identity based signature schemes based on pairings,” in Selected Areas in Cryptography. Springer, 2003, pp. 310–324.

[22] B. Lynn, “The pairing-based cryptography (pbc) library,” Last Visited: December, 2013. [Online]. Available: http://crypto.stanford.edu/pbc

[23] S. McVittie, “A secret sharing library - libgfshare,” Last Visited: December, 2013. [Online]. Available: https://launchpad.net/libgfshare

[24] E. A. Young, T. J. Hudson, and R. S. Engelschall, “Openssl,” Last Visited: December, 2013. [Online]. Available: http://www.openssl.org/

[25] D. Eastlake and P. Jones, “Us secure hash algorithm 1 (sha1),” RFC 3174, September, 2001.

[26] W. Jansen and T. Grance, “Nist sp 800-144 draft: guidelines on security and privacy in public cloud computing, security division,” Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, 2011, pp. 20 893–20 899.

 ISBN: 978-1-61208-319-3