Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 8, August 2014, pg.441 – 450

RESEARCH ARTICLE

AMENABILITY OF DISTRIBUTED DATA IN THE CLOUD BY MEANS OF CIA FRAMEWORK

[1]Shaik Mohammed Rayhan, [2]K.Bala Chowdappa

[1]PG Scholar, [2]Assistant Professor

[1,2]Department of CSE, G. Pulla Reddy Engineering College, A.P, India

[1] smdrayhan.mtech@gmail.com, [2] balak06@gmail.com

Abstract- Cloud Computing is a general term used to describe a new class of "network based computing" that takes place over the internet which facilitates highly scalable and 'anytime-anywhere access' services via web browser. The most important characteristic of cloud services is that any data may be processed remotely in various machines that users do not own or operate. Data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and these entities can also delegate the tasks to others and so on. Therefore data handling in the cloud goes through a complex and dynamic hierarchical service chain. So users' fears of losing control of their own data. This results in obstacle for the implementation of cloud services. To specify the existing criteria a "Highly Decentralized Cloud Information Accountability" framework is being proposed to keep track of usage of the user's data in the cloud with an attractive feature of Object Centered Approach that enables logging mechanisms to be enclosed with the Users' data and policies along with the distributed auditing mechanisms.

Index Terms— Cloud computing, accountability, data sharing

I. INTRODUCTION

Cloud computing paradigm makes huge virtualized compute resources available to users as pay-as-you-go style. Resource monitoring is the premise of many major operations such as network analysis, management, job scheduling, load balancing, event predicting, fault detecting, and fault recovery in Cloud computing. Cloud computing is more complicated than ordinary network owing to its heterogeneous and dynamic characteristics.

© 2014, IJCSMC All Rights Reserved

Hence, it is a vital part of the Cloud computing system to monitor the existence and characteristics of resources, services, computations, and other entities [1].

S. Zanikolas and R. Sakellariou present the taxonomy of existing Grid monitoring systems [2]. It indicates that some of the current Grid monitoring systems demonstrate high performance in specific contexts. For example, the widely-used distributed monitoring system, Ganglia [3], is such a tool that can monitor compute resources in Clusters and Grids.

To allay users' concerns, it is essential to provide an effective mechanism for users to monitor the usage of their data in the cloud. For example, users need to be able to ensure that their data are handled according to the service-level agreements made at the time they sign on for services in the cloud. Conventional access control approaches developed for closed domains such as databases and operating systems, or approaches using a centralized server in distributed environments, are not suitable, due to the following features characterizing cloud environments. First, data handling can be outsourced by the direct cloud service provider (CSP) to other entities in the cloud and these entities can also delegate the tasks to others, and so on. Second, entities are allowed to join and leave the cloud in a flexible manner. As a result, data handling in the cloud goes through a complex and dynamic hierarchical service chain which does not exist in conventional environments.

To overcome the above problems, we propose a novel approach, namely Cloud Information Accountability (CIA) framework, based on the notion of information accountability. Unlike privacy protection technologies which are built on the hide-it-or-lose-it perspective, information account-ability focuses on keeping the data usage transparent and trackable. Our proposed CIA framework provides end-to-end accountability in a highly distributed fashion. One of the main innovative features of the CIA framework lies in its ability of maintaining lightweight and powerful account- ability that combines aspects of access control, usage control and authentication. By means of the CIA, data owners can track not only whether or not the service-level agreements are being honored, but also enforce access and usage control rules as needed. Associated with the accountability feature, we also develop two distinct modes for auditing: push mode and pull mode. The push mode refers to logs being periodically sent to the data owner or stakeholder while the pull mode refers to an alternative approach whereby the user
(or another authorized party) can retrieve the logs as needed.
In summary, our main contributions are as follows:

We propose a novel automatic and enforceable logging mechanism in the cloud. To our knowledge, this is the first time a systematic approach to data accountability through the novel usage of JAR files is proposed. Our proposed architecture is platform independent and highly decentralized, in that it does not require any dedicated authentication or storage system in place. We go beyond traditional access control in that we provide a certain degree of usage control for the protected data after these are delivered to the receiver. We conduct experiments on a real cloud testbed. The results demonstrate the efficiency, scalability, and granularity of our approach. We also provide a detailed security analysis and discuss the reliability and strength of our architecture. We have made the following new contributions. First, we integrated integrity checks and oblivious hashing (OH) technique to our system in order to strengthen the dependability of our system in case of compromised JRE.

We also updated the log records structure to provide additional guarantees of integrity and authenticity. Second, we extended the security analysis to cover more possible attack scenarios. Third, we report the results of new experiments and provide a thorough evaluation of the system performance. Fourth, we have added a detailed discussion on related works to prepare readers with a better understanding of background knowledge. Finally, we have improved the presentation by adding more examples and illustration graphs.

© 2014, IJCSMC All Rights Reserved

II. RELATED WORK

In this section, we first review related works addressing the privacy and security issues in the cloud. Then, we briefly discuss works which adopt similar techniques as our approach but serve for different purposes.

2.1 Cloud Privacy and Security

Cloud computing has raised a range of important privacy and security issues. Such issues are due to the fact that, in the cloud, users' data and applications reside—at least for a certain amount of time—on the cloud cluster which is owned and maintained by a third party. Concerns arise since in the cloud it is not always clear to individuals why their personal information is requested or how it will be used or passed on to other parties. To date, little work has been done in this space, in particular with respect to accountability. Pearson et al. have proposed accountability mechanisms to address privacy concerns of end users [3] and then develop a privacy manager [4].Their basic idea is that the user's private data are sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is deobfuscated by the privacy manager to reveal the correct result. However, the privacy manager provides only limited features in that it does not guarantee protection once the data are being disclosed. In [5], the authors present a layered architecture for addressing the end-to-end trust management and accountability problem in federated systems. The authors' focus is very different from ours, in that they mainly leverage trust relationships for account- ability, along with authentication and anomaly detection.

Further, their solution requires third-party services to complete the monitoring and focuses on lower level monitoring of system resources. Researchers have investigated accountability mostly as a provable property through cryptographic mechanisms, particularly in the context of electronic commerce [6]. A representative work in this area is given by [7]. The authors propose the usage of policies attached to the data and present a logic for accountability data in distributed settings.

Similarly, Jagadeesan et al. recently proposed a logic for designing accountability-based distributed systems. In, Crispo and Ruffo proposed an interesting approach related to accountability in case of delegation. Delegation is complementary to our work, in that we do not aim at controlling the information workflow in the clouds. In a summary, all these works stay at a theoretical level and do not include any algorithm for tasks like mandatory logging. To the best of our knowledge, the only work proposing a distributed approach to accountability is from Lee and colleagues. The authors have proposed an agent-based system specific to grid computing. Distributed jobs, along with the resource consumption at local machines are tracked by static software agents. The notion of account-ability policies in [2] is related to ours, but it is mainly focused on resource consumption and on tracking of subjobs processed at multiple computing nodes, rather than access control.

2.2 Other Related Techniques

With respect to Java-based techniques for security, our methods are related to self-defending objects (SDO) [1]. Self-defending objects are an extension of the object-oriented programming paradigm, where software objects that offer sensitive functions or hold sensitive data are responsible for protecting those functions/data. Similarly, we also extend the concepts of object-oriented programming. The key difference in our implementations is that the authors still rely on a centralized database to maintain the access records, while the items being protected are held as separate files. In previous work, we provided a Java-based approach to prevent privacy leakage from indexing [9], which could be integrated with the CIA framework proposed in this work since they build on related architectures. In terms of authentication techniques, Appel and Felten [8] proposed the Proof-Carrying authentication (PCA)framework. The PCA includes a high order logic language that allows quantification over predicates, and

© 2014, IJCSMC All Rights Reserved

focuses on access control for web services. While related to ours to the extent that it helps maintaining safe, high-performance, mobile code, the PCA's goal is highly different from our research, as it focuses on validating code, rather than monitoring content. Another work is by Mont et al. who proposed an approach for strongly coupling content with access control, using Identity-Based Encryption (IBE) [6]. We also leverage IBE techniques, but in a very different way. We do not rely on IBE to bind the content with the rules. Instead, we use it to provide strong guarantees for the encrypted content and the log files, such as protection against chosen plaintext and cipher text attacks. In addition, our work may look similar to works on secure data provenance [5], [6], [15], but in fact greatly differs from them in terms of goals, techniques, and application domains. Works on data provenance aim to guarantee data integrity by securing the data provenance. They ensure that no one can add or remove entries in the middle of a provenance chain without detection, so that data are correctly delivered to the receiver. Differently, our work is to provide data accountability, to monitor the usage of the data and ensure that any access to the data is tracked. Since it is in a distributed environment, we also log where the data go. However, this is not for verifying data integrity, but rather for auditing whether data receivers use the data following specified policies.

III. PROBLEM STATEMENT

We begin this section by considering an illustrative example which serves as the basis of our problem statement and will be used throughout the paper to demonstrate the main features of our system.

Example 1. Alice, a professional photographer, plans to sell her photographs by using the SkyHigh Cloud Services.

For her business in the cloud, she has the following requirements:

. Her photographs are downloaded only by users who have paid for her services.

. Potential buyers are allowed to view her pictures first before they make the payment to obtain the download right.

. Due to the nature of some of her works, only users from certain countries can view or download some sets of photographs.

. For some of her works, users are allowed to only view them for a limited time, so that the users cannot reproduce her work easily.

. In case any dispute arises with a client, she wants to have all the access information of that client.

. She wants to ensure that the cloud service providers of Sky High do not share her data with other service providers, so that the accountability provided for individual users can also be expected from the cloud service providers.

With the above scenario in mind, we identify the common requirements and develop several guidelines to achieve data accountability in the cloud. A user who subscribed to a certain cloud service, usually needs to send his/her data as well as associated access control policies (if any) to the service provider. After the data are received by the cloud service provider, the service provider will have granted access rights, such as read, write, and copy, on the data. Using conventional access control mechanisms, once the access rights are granted, the data will be fully available at the service provider. In order to track the actual usage of the data, we aim to develop novel logging and auditing techniques which satisfy the following requirements:

1. The logging should be decentralized in order to adapt to the dynamic nature of the cloud. More specifically, log files should be tightly bounded with the corresponding data being controlled, and require minimal infrastructural support from any server.

© 2014, IJCSMC All Rights Reserved

2. Every access to the user's data should be correctly and automatically logged. This requires integrated techniques to authenticate the entity who accesses the data, verify, and record the actual operations on the data as well as the time that the data have been accessed.

3. Log files should be reliable and tamper proof to avoid illegal insertion, deletion, and modification by malicious parties. Recovery mechanisms are also desirable to restore damaged log files caused by technical problems.

4. Log files should be sent back to their data owners periodically to inform them of the current usage of their data. More importantly, log files should be retrievable anytime by their data owners when needed regardless the location where the files are stored.

5. The proposed technique should not intrusively monitor data recipients' systems, nor it should introduce heavy communication and computation overhead, which otherwise will hinder its feasibility and adoption in practice.

IV. CLOUD INFORMATION ACCOUNTABILITY

In this section, we present an overview of the Cloud Information Accountability framework and discuss how the CIA framework meets the design requirements discussed in the previous section.

The Cloud Information Accountability framework proposed in this work conducts automated logging and distributed auditing of relevant access performed by any entity, carried out at any point of time at any cloud service provider. It has two major components: logger and log harmonizer.

4.1 Major Components

There are two major components of the CIA, the first being the logger, and the second being the log harmonizer. The logger is the component which is strongly coupled with the user's data, so that it is downloaded when the data are accessed, and is copied whenever the data are copied. It handles a particular instance or copy of the user's data and is responsible for logging access to that instance or copy. The log harmonizer forms the central component which allows the user access to the log files. The logger is strongly coupled with user's data (either single or multiple data items). Its main tasks include automatically logging access to data items that it contains, encrypting the log record using the public key of the content owner, and periodically sending them to the logharmonizer. It may also be configured to ensure that access and usage control policies associated with the data are honored. For example, a data owner can specify that user Xis only allowed to view but not to modify the data. The logger will control the data access even after it is down-loaded by user X. The logger requires only minimal support from the server (e.g., a valid Java virtual machine installed) in order to be deployed. The tight coupling between data and logger, results in a highly distributed logging system, therefore meeting our first design requirement. Furthermore, since the logger does not need to be installed on any system or require any special support from the server, it is not very intrusive in its actions, thus satisfying our fifth requirement. Finally, the logger is also responsible for generating the error correction information for each log record and sends the same to the log harmonizer. The error correction information combined with the encryption and authentication mechanism provides a robust and reliable recovery mechanism, therefore meeting the third requirement.

© 2014, IJCSMC All Rights Reserved

The log harmonizer is responsible for auditing. Being the trusted component, the log harmonizer generates the master key. It holds on to the decryption key for the IBE key pair, as it is responsible for decrypting the logs. Alternatively, the decryption can be carried out on the client end if the path between the log harmonizer and the client is not trusted. In this case, the harmonizer sends the key to the client in a secure key exchange.

It supports two auditing strategies: push and pull. Under the push strategy, the log file is pushed back to the data owner periodically in an automated fashion. The pull mode is an on-demand approach, whereby the log file is obtained by the data owner as often as requested. These two modes allow us to satisfy the aforementioned fourth design requirement. In case there exist multiple loggers for the same set of data items, the log harmonizer will merge log records from them before sending back to the data owner. The log harmonizer is also responsible for handling log file corruption. In addition, the log harmonizer can itself carry out logging in addition to auditing. Separating the logging and auditing functions improves the performance. The logger and the log harmonizer are both implemented as lightweight and portable JAR files. The JAR file implementation provides automatic logging functions, which meets the second design requirement.

4.2 Data Flow

The overall CIA framework, combining data, users, logger and harmonizer is sketched in Fig. 1. At the beginning, each user creates a pair of public and private keys based on Identity-Based Encryption [4] (step 1 in Fig. 1). This IBE scheme is a Weil-pairing-based IBE scheme, which protects us against one of the most prevalent attacks to our architecture as described in Section 7. Using the generated key, the user will create a logger component which is a JAR file, to store its data items.

The JAR file includes a set of simple access control rules specifying whether and how the cloud servers, and possibly other data stakeholders (users, companies) are authorized to access the content itself. Then, he sends the JAR file to the cloud service provider that he subscribes to. To authenticate the CSP to the JAR (steps 3-5 in Fig. 1), we use OpenSSL-based certificates, wherein a trusted certificate authority certifies the CSP. In the event that the access is requested by a user, we employ SAML-based authentication [8], wherein a trusted identity provider issues certificates verifying the user's identity based on his username.

The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security (detailed discussion is in Section 7) without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption (step 7 in Fig. 1).

© 2014, IJCSMC All Rights Reserved

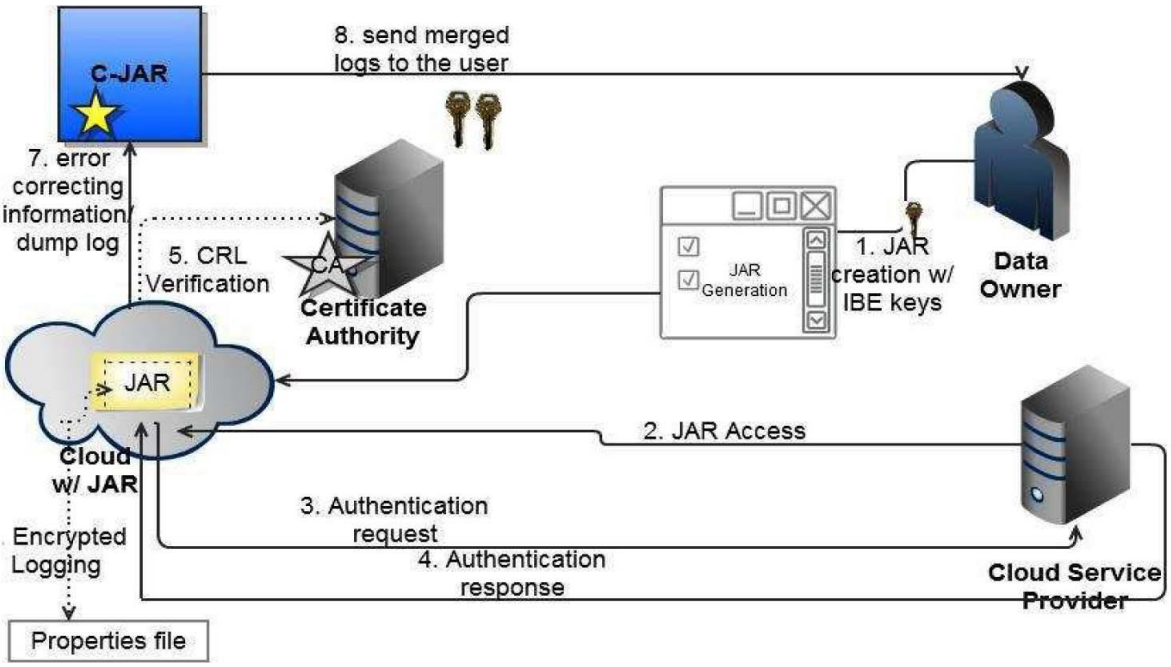


Fig.1. Overview Accountability Frame Work

Once the authentication succeeds, the service provider (or the user) will be allowed to access the data enclosed in the JAR. Depending on the configuration settings defined at the time of creation, the JAR will provide usage control associated with logging, or will provide only logging functionality. As for the logging, each time there is an access to the data, the JAR will automatically generate a log record, encrypt it using the public key distributed by the data owner, and store it along with the data (step 6 in Fig. 1). The encryption of the log file prevents unauthorized changes to the file by attackers. The data owner could opt to reuse the same key pair for all JARs or create different key pairs for separate JARs. Using separate keys can enhance the security(detailed discussion is in Section 7) without introducing any overhead except in the initialization phase. In addition, some error correction information will be sent to the log harmonizer to handle possible log file corruption (step 7 in Fig. 1). To ensure trustworthiness of the logs, each record is signed by the entity accessing the content. Further, individual records are hashed together to create a chain structure, able to quickly detect possible errors or missing records. The encrypted log files can later be decrypted and their integrity verified. They can be accessed by the data owner or other authorized stakeholders at any time for auditing purposes with the aid of the log harmonizer (step 8 in Fig. 1).

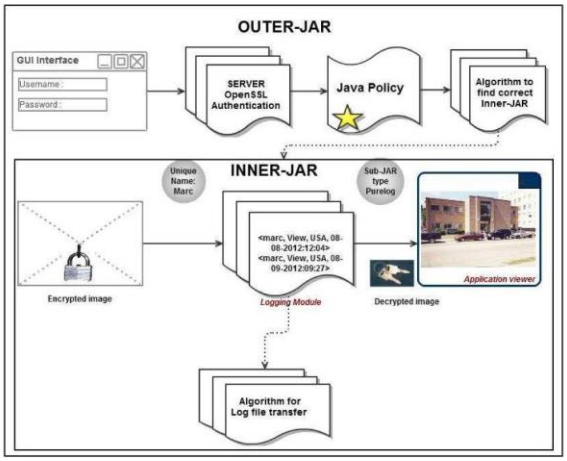
As discussed in Section 7, our proposed framework prevents various attacks such as detecting illegal copies of users' data. Note that our work is different from traditional logging methods which use encryption to protect log files. With only encryption, their logging mechanisms are neither automatic nor distributed. They require the data to stay within the boundaries of the centralized system for the logging to be possible, which is however not suitable in the cloud.

© 2014, IJCSMC All Rights Reserved

V. AUTOMATED LOGGING MECHANISM

In this section, we first elaborate on the automated logging mechanism and then present techniques to guarantee dependability. We represent the structure of the logging mechanism is as follows:

The Logger Structure



Weleverage the programmable capability of JARs to conduct automated logging. A logger component is a nested Java JAR file which stores a user's data items and corresponding log files. As shown in Fig. 2, our proposed JAR file consists of one outer JAR enclosing one or more inner JARs.

The main responsibility of the outer JAR is to handle authentication of entities which want to access the data stored in the JAR file. In our context, the data owners may not know the exact CSPs that are going to handle the data. Hence, authentication is specified according to the servers' functionality (which we assume to be known through a lookup service), rather than the server's URL or identity. For example, a policy may state that Server X is allowed to download the data if it is a storage server. As discussed below, the outer JAR may also have the access control functionality to enforce the data owner's requirements, specified as Java policies, on the usage of the data. A Java policy specifies which permissions are available for a particular piece of code in a Java application environment. The permissions expressed in the Java policy are in terms of File System Permissions. However, the data owner can specify the permissions in user-centric terms as opposed to the usual code-centric security offered by Java, using Java Authentication and Authorization Services. Moreover, the outer JAR is also in charge of selecting the correct inner JAR according to the identity of the entity who requests the data.

VI. PERFORMANCE STUDY

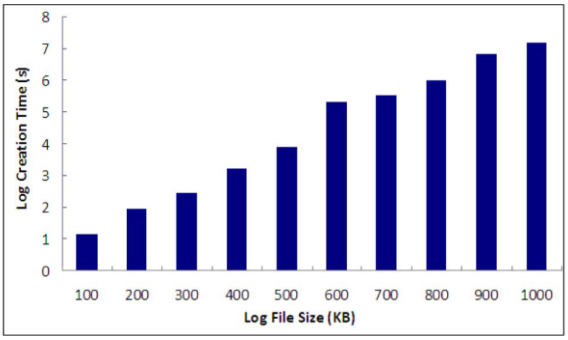
In this section, we first introduce the settings of the test environment and then present the performance study of our system.

6.1 Experimental Settings

We tested our CIA framework by setting up a small cloud, using the Emulabtestbed [42]. In particular, the test environment consists of several OpenSSL-enabled servers: one head node which is the certificate authority, and several computing nodes. Each of the servers is installed with Eucalyptus [41]. Eucalyptus is an open source cloud implementation for Linux-based systems. It is loosely based on Amazon EC2, therefore bringing the powerful functionalities of Amazon EC2 into the open source domain. We used Linux-based servers running Fedora 10 OS.

© 2014, IJCSMC All Rights Reserved

Each server has a 64-bit Intel Quad Core Xeon E5530 processor, 4 GB RAM, and a 500 GB Hard Drive. Each of the servers is equipped to run the OpenJDK runtime environment with IcedTea6 1.8.2.



6.2 Experimental Results

In the experiments, we first examine the time taken to create a log file and then measure the overhead in the system.

With respect to time, the overhead can occur at three points:

During the authentication, during encryption of a log record, and during the merging of the logs. Also, with respect to storage overhead, we notice that our architecture is very lightweight, in that the only data to be stored are given by the actual files and the associated logs. Further, JAR act as a compressor of the files that it handles. In particular, as introduced in Section 3, multiple files can be handled by the same logger component. To this extent, we investigate whether a single logger component, used to handle more than one file, results in storage overhead.

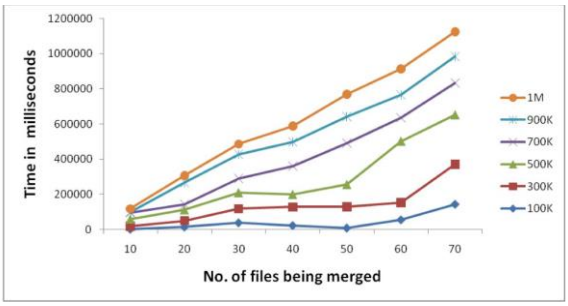
6.2.1 Log Creation Time

In the first round of experiments, we are interested in finding out the time taken to create a log file when there are entities continuously accessing the data, causing continuous logging. Results are shown in Fig. 5. It is not surprising to see that the time to create a log file increases linearly with the size of the log file. Specifically, the time to create a 100 Kb file is about 114.5 ms while the time to create a 1 MB file averages at 731 ms. With this experiment as the baseline, one can decide the amount of time to be specified between dumps, keeping other variables like space constraints or network traffic in mind.

6.2.2 Authentication Time

The next point that the overhead can occur is during the authentication of a CSP. If the time taken for this authentication is too long, it may become a bottleneck for accessing the enclosed data. To evaluate this, the head node issued OpenSSL certificates for the computing nodes and we measured the total time for the OpenSSL authentication.

© 2014, IJCSMC All Rights Reserved



Considering one access at the time, we find that the authentication time averages around 920 ms which proves that not too much overhead is added during this phase. As of present, the authentication takes place each time the CSPneeds to access the data. The performance can be further improved by caching the certificates.

The time for authenticating an end user is about the same when we consider only the actions required by the JAR, viz. obtaining a SAML certificate and then evaluating it. This is because both the OpenSSL and the SAML certificates are handled in a similar fashion by the JAR. When we consider the user actions (i.e., submitting his username to the JAR), it averages at 1.2 minutes.

REFERENCES

[1] P. Ammann and S. Jajodia, "Distributed Timestamp Generation in Planar Lattice Networks," ACM Trans. Computer Systems, vol. 11, pp. 205-225, Aug. 1993.
[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z.Peterson, and D. Song, "Provable Data Possession at UntrustedStores," Proc. ACM Conf. Computer and Comm. Security, pp. 598- 609, 2007.
[3] E. Barka and A. Lakas, "Integrating Usage Control with SIP-BasedCommunications," J. Computer Systems, Networks, and Comm., vol. 2008, pp. 1-8, 2008.
[4] D. Boneh and M.K. Franklin, "Identity-Based Encryption from theWeil Pairing," Proc. Int'l Cryptology Conf. Advances in Cryptology, pp. 213-229, 2001.
[5] R. Bose and J. Frew, "Lineage Retrieval for Scientific DataProcessing: A Survey," ACM Computing Surveys, vol. 37, pp. 1-28, Mar. 2005.
[6] P. Buneman, A. Chapman, and J. Cheney, "Provenance Management in Curated Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '06), pp. 539-550, 2006.
[7] B. Chun and A.C. Bavier, "Decentralized Trust Management and Accountability in Federated Systems," Proc. Ann. Hawaii Int'l Conf. System Sciences (HICSS), 2004.
[8] OASIS Security Services Technical Committee, "Security Assertion Markup Language (saml) 2.0," http://www.oasis-open.org/committees/tc home.php?wg abbrev=security, 2012.
[9] R. Corin, S. Etalle, J.I. den Hartog, G. Lenzini, and I. Staicu, "ALogic for Auditing Accountability in Decentralized Systems," Proc. IFIP TC1 WG1.7 Workshop Formal Aspects in Security and Trust,pp. 187-201, 2005.
[10] B. Crispo and G. Ruffo, "Reasoning about Accountability withinDelegation," Proc. Third Int'l Conf. Information and Comm. Security(ICICS), pp. 251-260, 2001.
[11] Y. Chen et al., "Oblivious Hashing: A Stealthy Software

© 2014, IJCSMC All Rights Reserved