Int J Software Informatics, Vol.4, No.4, December 2010, pp. 351–365 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
 Tel: +86-10-62661040

Confidentiality Protection in Cloud Computing Systems

Stephen S. Yau and Ho G. An

(Information Assurance Center, and School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, Arizona, USA)

Abstract Current cloud computing systems pose serious limitation to protecting users' data confidentiality. Since users' sensitive data is presented in unencrypted forms to remote machines owned and operated by third party service providers, the risks of unauthorized disclosure of the users' sensitive data by service providers may be high. Many techniques for protecting users' data from outside attackers are available, but currently there exists no effective way for protecting users' sensitive data from service providers in cloud computing. In this paper, an approach is presented to protecting the confidentiality of users' data from service providers, and ensures that service providers cannot access or disclose users' confidential data being processed and stored in cloud computing systems. Our approach has three major aspects: 1) separating software service providers and infrastructure service providers in cloud computing, 2) hiding information of the owners of data, and 3) data obfuscation. An example to show how our approach can protect the confidentiality of users' data from service providers in cloud computing is given. Experimental results are presented to show that our approach has reasonable performance.

Key words: data confidentiality; cloud computing system architecture; data obfuscation

Yau SS, An HG. Confidentiality protection in cloud computing systems. *Int J Software Informatics*, **2010, 4(4): 351–365. http://www.ijsi.org/1673-7288/4/i68.htm**

1 Introduction

Cloud computing systems provide various Internet-based data storage and services. Due to its many benefits, including cost effectiveness and high scalability and flexibility, cloud computing has gained significant momentum recently as a new paradigm for distributed computing for various applications, especially for business applications. With the rapid growth of the Internet, service-oriented architecture (SOA) and virtualization technologies, cloud computing leads to the vision of "Internet as a supercomputer." This vision incorporates the concepts of "software as a servic", "platform as a service", and "infrastructure as a service." However, cloud computing has a major limitation to be broadly adopted due to the serious barrier that current cloud computing systems cannot protect the confidentiality of users' data from service providers[1]. A recent survey[2] shows that most of cloud users fear the

* Corresponding author: Stephen S. Yau, Email: yau@asu.edu
Received 2010-08-31; revised 2010-12-15; accepted 2010-12-25.

leakage of their sensitive data in the cloud because their data is processed and stored on remote machines owned and operated by various service providers, which the users do not have any control. Since users' data is processed and stored in cloud computing systems in unencrypted form in current cloud computing systems, there are serious risks of unauthorized uses of the users' data by service providers.

There exist many techniques for confidentiality protection in various computing systems for access control, identity management, end-to-end data confidentiality and integrity assurance, etc. However, these techniques cannot be applied to cloud computing systems for confidentiality protection because they were developed only for protection from malicious third parties outside the systems. Since cloud computing systems have service providers inside the systems as new threat on data confidentiality, the basic ideas about data confidentiality must be changed and an effective new technique for confidentiality protection from service providers of cloud computing systems is needed.

In this paper, we will present an approach to protecting confidentiality of users' data in cloud computing systems. In our approach, we do not consider confidentiality protection from attackers outside cloud computing systems. Instead our approach focuses on protecting confidentiality of users' data from service providers inside the cloud computing systems. We assume that cloud service providers may not be trustworthy and may collect users' sensitive data in their systems. Our approach has the following specific goals:

- Service users can make sure that their sensitive data, which the users specify not to be shared with their service providers is not disclosed to their service providers even if there is no cooperation from their service providers
- Our approach does not cause much overhead on service performance.

This paper is organized as follows. In Section 2, we will articulate users' data confidentiality protection from service providers in cloud computing systems. This concept will be used to determine whether our approach can protect users' confidential data from service providers. In Section 3, the problems of current cloud computing architecture in terms of protection of users' data confidentiality will be discussed. In Section 4, we will present our approach, including a new architecture for cloud computing system and using data obfuscation to achieve the above both goals of our approach. In Section 5, our data obfuscation and de-obfuscation developed for protecting user's data confidentiality in cloud computing will be discussed. In Section 6, we will present an example to show how the confidentiality of user's data can be protected by our approach in cloud computing systems. In Section7, experimental results are presented to show that our approach has reasonable performance. In Section 8, we will discuss the advantages and limitations of our approach as well as future work.

2 Protection of Users' Data Confidentiality from Service Providers in Cloud Computing Systems

In Ref.[3], confidentiality is defined as the assurance that sensitive information is not disclosed to unauthorized persons, processes, or devices. Hence, we must make sure that the users' confidential data, which the users do not want to be accessed by

service providers is not disclosed to service providers in the cloud computing systems, including applications, platforms, CPU and physical memories.

It is noted that users' confidential data is disclosed to a service provider only if all of the following three conditions are satisfied simultaneously:

Condition 1) The service provider knows where the users' confidential data is located in the cloud computing systems.
Condition 2) The service provider has the privilege to access and collect the users' confidential data in the cloud computing systems.
Condition 3) The service provider can understand the meaning of the users' data.

This is due to the following reasons: In order to collect users' data, the service provider must know the location of the data in cloud computing systems and have the privilege to access the data. Even if the service provider can collect users' data successfully, the service provider may not be able to understand the meaning of the data unless the service provider has at the least some of the following information to understand the meanings of the data: *i)* types of data, *ii)* functionalities and interfaces of the application using the data and *iii)* format of the data

Hence, if we can prevent the service providers from satisfying all the above three conditions, we can protect the confidentiality of users' data in cloud computing systems from the service providers.

3 Problems with the Current Cloud Computing Architecture

The current cloud computing system consists of three layers: software layer, platform layer and infrastructure layer, as shown in Fig.1. The software layer provides the interfaces for users to use service provider's applications running on a cloud infrastructure. The platform layer provides the operating environment for the software to run using system resources. The infrastructure layer provides the hardware resources for computing, storage, and networks. Platforms or infrastructures can be provided as virtual machines. The following are the major problems of current cloud computing systems:

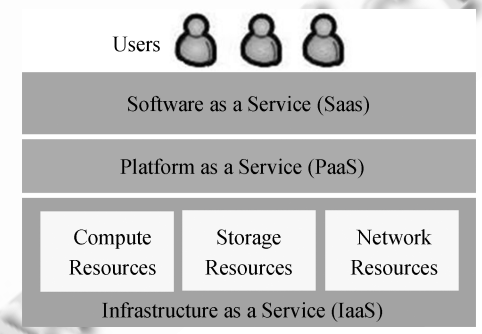

Figure 1. Current cloud computing architecture

- Each service provider has its own software layer, platform layer and infrastructure layer. When a user uses a cloud application from a service provider, the

user is forced to use the platform and infrastructure provided by the same service provider, and hence the service provider knows where the users' data is located and has full access privileges to the data.

- The user is forced to use the interfaces provided by the service provider, and users' data has to be in a fixed format specified by the service provider, and hence the service provider knows all the information required for understanding users' data.

Therefore, we cannot prevent service providers from satisfying all of the three conditions in Section 2.

4 Overview of our Approach

Our approach can be depicted in Fig.2. In our approach, we have the following seven entities to protect the confideitiality of data processed and stored in cloud computing systems: Software Cloud, Infrastructure Cloud, Software Service Broker, Infrastructure Service Broker, Software Service Attestation Authority, Data Obfuscator and Data De-obfuscator[16]. The Software Cloud and Infrastructure Cloud have the same features of the software layer in the ordinary cloud computing architecture discussed in Section 3. However, in our approach, we require that the software layer and infrastructure layer are not managed by the same service provider. The Software Service Brokers and Infrastructure Service Brokers have the same functionality of the service brokers in service-oriented architecture (SOA), but they have the additional function for identity-anonymization. The Software Service Attestation Authority, Data Obfuscator and Data De-obfuscator are additional entities introduced in our approach. Our approach makes sure that any of these entities in a cloud computing system does not satisfy the three conditions simultaneously in Section 2. We will describe each of these entities below:

- **Software cloud**: A *Software Cloud* provides the software as a service upon users' requests. Each software cloud may contain multiple software services, and each software service can be discovered and accessed by users through Software Service Broker. An authenticated user with proper credentials[4] can request and acquire a service instance from the software cloud. A *service instance* is a piece of compiled executable code. The executable code can be deployed and run on any Infrastructure Cloud. In order to protect the intellectual property of the software, the code is compiled using various code obfuscation technologies[5,6] so that reverse engineering on the service instance is computationally infeasible. This implies that infrastructure service providers cannot understand the functionality and algorithms of the service instance by examining at the code when the service instance is running on *Infrastructure Cloud*.
- **Infrastructure cloud**: An *Infrastructure Cloud* provides virtualized system resources, such as CPU, memory, and network resources. An authenticated user can request a virtual machine on which the user can deploy any platform or operating system to execute a software service instance.
- **Software service broker**: A *Software Service Broker* has two major functions. First, it helps users automatically discover and access available software services.

Second, it helps users hide their identities from software cloud service providers. A Software Service Broker provides identity anonymization service, by which users can use pseudonyms instead of their true identities so that the users can acquire service instances without revealing their identities. The anonymization of user identities is very important for protecting the confidentiality of users' data because the information about the data owners may reveal a lot of sensitive information regarding the data.

- **Infrastructure service broker**: An *Infrastructure Service Broker* has two major functions similar to a Software Service Broker. It helps users automatically discover and use available infrastructure services. It also provides identity anonymization service to prevent the system from revealing users' true identities.
- **The software service attestation authority (SSAA)**: The *SSAA* is a third party authority to verify that a service instance does not perform any malicious activity that may disclose users' confidential data. For example, a software service developer may have injected a hidden function on the software service which transmits user's confidential data to an unauthorized third party during its processing without the user's consent. Since users do not know whether a service instance will act as described in the service description, the SSAA needs to help users test the service instance before users using it. When a service instance is deployed on the Service Testing Platform of SSAA, the SSAA tests whether the service instance performs exactly what the service provider claims, and whether the service instance may transmit users' data to any unauthorized entity.
  The testing can be done by *i)* verifying whether the service instance satisfies the web service description language (WSDL) specification of the service, and *ii)* monitoring all the network traffics the software service produces during its processing. An approach to automated web service testing based on syntactic and semantic analysis of WSDL specification was presented in Ref.[7]. After completing the testing of the service instance, SSAA issues a digital certificate for the tested service instance. A certificate is attached to the service instance so that users will know whether the service instance has passed the SSAA's testing.
- **Data obfuscator**: A *Data Obfuscator* is a middleware provided by a user that can be deployed on a virtual machine in an Infrastructure Cloud. The service instance in an Infrastructure Cloud can use system resources only through the interfaces of the Data Obfuscator. The Data Obfuscator has the following security functionality to ensure that users' confidential data is not disclosed to infrastructure service providers:
  i) Sets up an encryption key with the user. The key is chosen by the user and not revealed to any process, platform or device of the cloud computing system.
  ii) Encrypts any data being stored in the physical storage of the cloud computing system or being transmitted through the network.
  iii) Obfuscates users' sensitive data being processed in the service instances. The obfuscated data cannot be de-obfuscated in the platforms or physical devices of an Infrastructure Cloud so that its infrastructure provider cannot understand

the meaning of the users' sensitive data. The details of the data obfuscation will be discussed in Section 5.

- **Data De-obfuscator**: A *Data De-obfuscator* de-obfuscates obfuscated data so that a user can see the plain data. A Data De-obfuscator remains in the user's personal computer all the time.

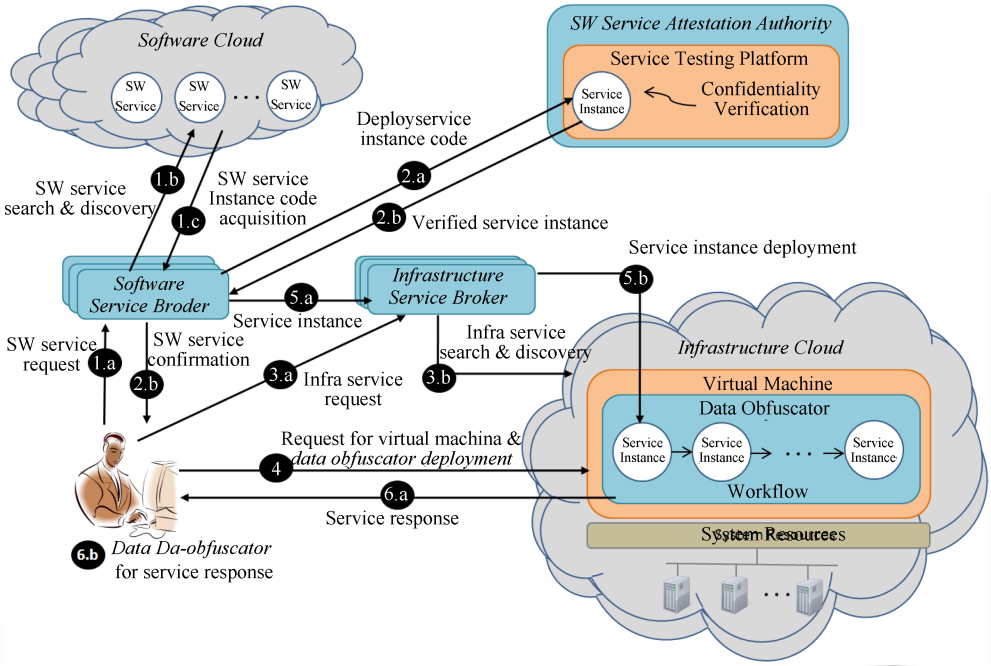

Figure 2. Our approach to protecting confidentiality of users' data in cloud computing. The numbers in the figure correspond to the steps of our approach described at the end of Section 4.

Our approach is based on three features: *1)* separation of software service providers and infrastructure service providers, *2)* hiding information about the owner of data and *3)* data obfuscation. With these three features, our approach can ensure that at least one of the three conditions in Section 2 is not satisfied in each of the seven entities in cloud computing system due to the following reasons:

- Although a software service provider knows the functionality of a service instance and the data format of users' input and can satisfy *Condition 3)*, since software service instances are deployed to Infrastructure Cloud through its Software Service Broker and Infrastructure Service Broker, the software service provider does not know where users' data is located, and does not have the privilege to access the data. Thus, the software service provider cannot satisfy *Conditions 1)* and *2)*.
- Although an infrastructure service provider knows the locations of users' specific confidential data and has the privilege to access the data being processed and stored in the infrastructure cloud, the infrasture service provideer cannot understand the meaning of the data because *i)* he/she does not know the functionality of the deployed software instance and the format of the users' specific confidential data, *ii)* the deployed software instance cannot be reverse-engineered, *iii)*

the infrastructure service provider does not know the identity of the data owners, and *iv)* the users' specific confidential data is obfuscated while processed and stored in the infrastructure. Hence, the infrastructure service provider cannot satisfy *Condition 3)*.

- Because the Software Service Broker does not know where users' data is located, and does not have the privilege to access user's data, the Software Service Broker cannot satisfy *Conditions 1)* and *2)*.
- Because an Infrastructure Service Broker does not know the functionality of deployed software instance, the data format of user's data, and does not have the privilege to access the data, the Infrastructure Service Broker cannot satisfy *Conditions 2)* and *3)*.

Our approach is depicted in Fig.2, where the numbers are corresponding to the steps. It can be summarized as follows:

S1) *i)* A user requests any Software Service Broker to find a software service by providing the specification of the software service[8]. *ii)* The Software Service Broker performs automatic service discovery[9] to find a service instance in the Software Cloud that satisfies the user's requested service requirement specification. *iii)* The Software Service Broker acquires the discovered software instance using an anonymous credential[10].

S2) *i)* The Software Service Broker deploys the acquired service instance to the testing platform of a SSAA. The SSAA verifies whether the service instance performs according to its description, and the service instance does not transmit users' specific confidential data to any unauthorized entity. *ii)* After the verification procedure, the software service instance is sent back to the Software Service Broker.

S3) *i)* The user asks the Infrastructure Service Broker to find an infrastructure service compatible to the service instance. *ii)* The Infrastructure Service Broker discovers an infrastructure service provider, who has the capability to execute the acquired software service instance.

S4) The user requests the infrastructure service provider to set up a virtual machine and then deploys the Data Obfuscator on the virtual machine using the Agent Deployment Plans (ADPs), for automated middleware deployment and migration in service-based systems according to Ref.[11].

S5) *i)* The service instance acquired in *S1)* is sent to the Infrastructure Service Broker. *ii)* The service instance is deployed on the workflow of the Data Obfuscator set up in *S4)*.

S6) *i)* The user sends his/her data to the workflow to process, including the obfuscated data. During the processing of users' input data, the user's specific condential data is obfuscated so that the infrastructure service provider cannot understand the meaning of user's data. After completing the processing, a service response of the workflow is sent to the user indicating that the processing of the user's input data has been completed. *ii)* The service response is de-obfuscated to plain data in the user's computer.

5 Data Obfuscation in Infrastructure Clouds

Data obfuscation is the process of transforming the format or structure of data to

hide the meaning of data. The major difference between encryption and obfuscation is that encrypted data cannot be processed until it is decrypted, but obfuscated data can be processed without de-obfuscation. In our approach, data obfuscation is used to process users' data in an infrastructure cloud without revealing any users' specific confidential data to the infrastructure service providers.

An approach to obfuscating data, which is transmitted from a user to software layer of a cloud computing system for protecting user's privacy is available[12]. However, this approach is limited in the use of data obfuscation because the obfuscated data must fit into the user interfaces provided by service providers. In our approach, data obfuscation takes place between the software layer and the infrastructure layer as shown in Fig.2 so that the use of data obfuscation is not constrained by the user interfaces provided by the service providers.

The general algebraic description of data obfuscation is as follows. Suppose a user wishes to use an infrastructure cloud service to process a function F on the user's input x without revealing the meaning of x to the infrastructure service provider. Obfuscation function O and de-obfuscation function D have following properties:

- $D(F(O(x,k),k) = F(x)$, where k is an obfuscation key unknown to the infrastructure service provider
- The infrastructure service provider cannot understand the meaning of x by examining $O(x,k)$
- D or k cannot be derived from O
- $O(x,k)$ and $D(F(O(x,k),k)$ can be calculated in polynomial time

When a data obfuscator is deployed by a user on a virtual machine in the Infrastructure Cloud, the data obfuscator obfuscates the user's input data x using one of the following three methods:

Method A) **Take dummy input data from a user and generate arbitrary dummy outputs.** In our approach, a user's input data is entered to a software service instance through the data obfuscator. Since the infrastructure service provider does not know the input data format of the service instance, the data obfuscator can take any number of dummy input data from the users. Only the user's inputs to be processed are given to the software service instance, and the data obfuscator generates arbitrary dummy outputs on the user's dummy inputs. The outputs generated by the service instance from the user's inputs, and the dummy outputs generated from the user's dummy inputs by the data obfuscator are mixed and encrypted together, and sent back to the user. The dummy outputs are marked so that data de-obfuscator in the user's computer can recognize the dummy outputs after the all the outputs of the service instance are decrypted.

For example, given user's input data x, a software service function F and a secret integer key k, $O(x,k)$ generates dummy input data a_1, a_2... a_n, where $n > k$. For each dummy input data generated, the data obfuscator processes dummy functions $f_1(a_1)$, $f_2(a_2)$... $f_n(a_n)$ while the software service instance processes $F(x)$. Then, an array of output data $(f_1(a_1), f_2(a_2)...f_{k-1}(a_{k-1}), F(x), f_k(a_k)...f_n(a_n))$ is sent to the user. The data-de-obfuscator at the user's computer de-obfuscates the array of output data

using the secret key k. Hence, $D((f_1(a_1), f_2(a_2)...f_{k-1}(a_{k-1}), F(x), f_k(a_k)...f_n(a_n)) = F(x)$.

Method B) **Use a file system not known to all the infrastructure providers.** A file system is a method of storing and organizing files and data into computer memories and storage devices, such as hard disks or CD-ROMs. If an infrastructure service provider can understand the file system structure of a user's operating system running on the infrastructure cloud, the service provider may be able to locate and extract users' data from memories or storage devices. Data obfuscator uses a file system which is not known to any of the infrastructure providers so that the infrastructure service providers cannot extract meaningful data from the memory or storage devices of the cloud computing system.

For example, data obfuscator can change the number of disk sectors in a cluster or the format of the file allocation table in the file system so that the infrastructure service provider cannot understand the file system structure. In order to prevent the service providers from extracting the users' specific confidential data from network traffics, the user's specific confidentail data should be encrypted in the user's computer before being sent to service provider's infrastructure. The encrypted data is decrypted and processed on the data obfuscator's file system.

Method C) **Transform users' input data.** Data obfuscator transforms the format or value of users' input data so that the infrastructure service provider cannot understand the meaning of the data. All the operations associated with the original data must also be applicable to the transformed data so that the software service instance is able to process the obfuscated data.

For example, the data obfuscator transforms a regular string encoded with ASCII code into a different string format using a secret string encoder. Then, the string encoded with the secret stirng encoder must support all the operations associated with string data, such as concatenation, split, length calculation, comparison, uppercase, lowercase or character replacement.

The general algebraic description of data transformation for obfuscation is as follows. Given input data x, a set of associated operations $\{P_1, ..., P_n\}$, an obfuscation function O, a secret key k, and de-obfuscation function D:

$$P_i(x) = D(P_i(O(x,k)),k), \text{ where } 1 \leqslant i \leqslant n$$

Both numerical data and text data can be obfuscated using the data transformation method.

For numeric data, an arithmetic operation is performed on user's input data with secret integer k to transform the value of the data. For example, suppose a user wish to process a function $F(a,b) = a + b$, where a and b are integer values, in an Infrastructure Cloud without revealing the value of a and b to the infrastructure service provider. In order to obfuscate a and b, both a and b are multiplied by k. Then $F(ak, bk) = (ak+bk)$ is processed in the infrastructure cloud. The infrastructure service provider cannot find the value of a and b without knowing k. The user can de-obfuscate $(ak+bk)$ by dividing it by k and then get the value of $a+b$. Table1 shows obfuscation and de-obfuscation functions for each arithmetic operation for numeric data.

Table 1 Obfuscation and de-obfuscation functions for each arithmetic operation

Operation	User input	Obfuscation function	Obfuscated data	Deobfuscation function	Deobfuscated data
Addition	a, b	$O(x,k) = xk$	ak, bk	$D(x,k) = x/k$	$a+b$
Subtraction	a, b	$O(x,k) = xk$	ak, bk	$D(x,k) = x/k$	$a-b$
Multiplication	a, b	$O(x,k) = xk$	ak, bk	$D(x,k) = x/k^2$	$a \times b$
Division	a, b	$O(x,k) = x^k$	a^k, b^k	$D(x,k) = \sqrt[k]{x}$	a/b

For text data, a secret encoder is used to transform the format of the text data. A secret encoder can be generated by randomly shuffling the order of a standard text encoding table, such as ASCII code table. Once a secret encoder is generated by a user, the data obfuscator transforms each character of the user's text data using the secret encoder. Since the transformation is done character by character, any operation associated with original text data can be applied to the transformed data. After the transformed text data is processed in the infrastructure cloud, the data is sent to the user and de-obfuscated in the user's computer. During the de-obfuscation, each character of the transformed text data is mapped to the original ASCII code. For example, suppose a user wants to concatenate two strings "ab" and "cd" in an infrastructure cloud. The user generates a secret encoder by randomly shuffling the ASCII code table. Table 2 shows the standard ASCII code and a secret code generated by the user.

Table 2 The standard ASCII code and a secret string encoder

Character	Standard ASCII code	A secret code generated by a user
a	01100001	01111011
b	01100010	00100111
c	01100011	10110000
d	01100100	01111011

Data obfuscator transforms "ab" and "cd" to "0111101100100111" and "1011000 001111011" respectively using the secret code. The transformed strings are concatenated into "01111011001001111011000001111011" in the infrastructure cloud and sent to the user. Data de-obfuscator in the user's computer performs the mapping of the transformed text data to the original ASCII code so that the de-obfuscated data is mapped to "01100001011000100110001101100100", which means "abcd" in the original ASCII code.

ASCII code has 127 characters in its encoding table, and each character is represented by an 8-bits binary number. By shuffling the order of the 127 characters in the table, we can generate the factorial of 127 ($\approx 3 \times 10^{213}$) secret encoding tables. Thus, it is computationally infeasible for infrastructure service providers to find the meaning of the transformed text data using brute forcing attack.

It is noted that we discussed three methods for obfuscating and de-obfuscating data due to the following reasons: *Methods A)* and *C)* are used for obfuscating and de-obfuscating users' specific confidential data during the data processing time. *Method C)* is suitable for processing numerical and text data, and *method A)* is suitable for other types of data, such as image and vedio data. *Method B)* is used for obfuscating and de-obfuscating any types of data during the storing time.

6 An Illustrative Example

In this section, we present an example to illustrate how our approach can protect users' specific confidential data in cloud computing systems. Consider a group of users in different locations, who would like to have an online conference using cloud computing shown in Fig.3. They must be able to communicate with each other through voice, video, and/or instance messaging. They also need to share files with each other. To achieve this, they need to use the five services: Voice Communication Service, Video Communication Service, File Sharing Service, Instant Messaging Service, and Conference Controller. Since the voice data, video data, messages, or files may contain the specific confidential information which the users do not want the service providers to know, the users require the protection of such specific confidential data from the service providers. The virtual machine for online conferencing can be set up according to our approach as follows:

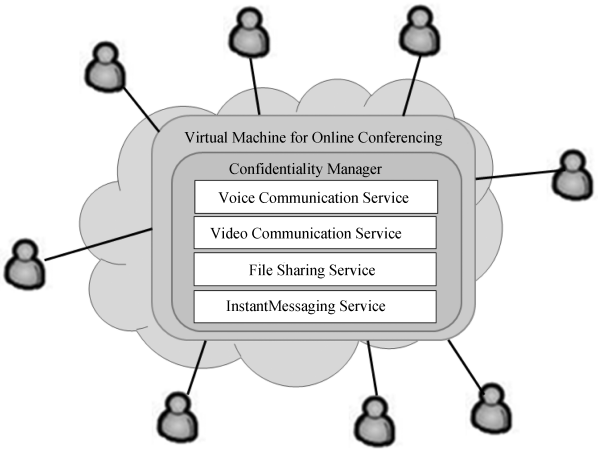

Figure 3. An example of online video conferencing to illustrate our approach

S1) *i)* The leader of the group requests a Software Service Broker to find the five software services: Voice Communication Service, Video Communication Service, File Sharing Service, Instant Messaging Service and Coference Controller. *ii)* The Software Service Broker discovers the services. *iii)* The Software Service Broker downloads the service instances of the five software services.
S2) *i)* The Software Service Broker deploys the service instances to the testing platform of an SSAA. *ii)* The SSAA verifies the software service instances.
S3) *i)* The leader of the group requests an Infrastructure Service Broker to find an infrastructure service compatible to the service instances. *ii)* The Infrastructure Service Broker discovers an infrastructure service.
S4) A virtual machine is set up in the Infrastructure Cloud. The leader of the group deploys the Data Obfuscator on the virtual machine. Obfuscation keys and encryption keys unknown to infrastructure service providers are sent to all other users.
S5) *i)* The service instances are sent to the Infrastructure Service Broker. *ii)* The service instances are deployed on the Data Obfuscator. The five service instances are composed to a workflow. The workflow provides all the functionality for online conferencing.

S6) *i)* The users of the group send their input data to the workflow to process. During the processing of the users' input data, the users' specific confidential input data is obfuscated. After completing the processing, a service response of the workflow is sent to all the users of the group that the processing of their input data has been completed. *ii)* The service response of the users' specific confidential input data is de-obfuscated.

The data obfuscation during the online conferencing is done as follows:

- All users' names are transformed to pseudonyms in the Infrastructure Cloud. The pseudonyms are de- obfuscated to the plain names in users' personal computers.
- All users' voice data is transformed using a secret audio encoder.
- All users' video data is transformed using a secret video encoder.
- All users' text messages are transformed using a secret string encoder.
- Dummy input data is generated frequently and sent to all the users.

In this example, at least one of the three conditions in Section 2 is not satisfied in each entity in the cloud computing system due to the following reasons:

- Software service providers do not know where users' data is located, and do not have the privilege to access the data. Hence, *Conditions 1)* and *2)* cannot be satisfied by the software service providers
- The infrastructure service provider cannot understand the meaning of users' specific confidential input data because *i)* the infrastructure service provider does not know the functionality of the deployed software instances and the data formats of the users' specific confidential input data, *ii)* the deployed software instance cannot be reverse-engineered, *iii)* the infrastructure service provider does not know the identities of the data owners, and *iv)* users' specific confidential input data is obfuscated. Hence, *Condition 3)* cannot be satisfied by the infrastructure service provider.
- The Software Service Broker does not know where users' data is located, and does not have the privilege to access the data. Hence, *Conditions 1)* and *2)* cannot be satisfied by the Software Service Broker.
- The Infrastructure Service Broker does not know the functionality of the deployed software instances, the format of users' specific confidential input data, and does not have the privilege to access such data. Hence, *Conditions 2)* and *3)* cannot be satisfied by the Infrastructure Service Broker.

Hence, the confidentiality of users' specific confidential data is protected in this example.

7 An Experiment on Performance of Data Obfuscator and Data De-obfuscator

We implemented a cloud service using C# Microsoft .NET framework for collaborative online documentation. Using this cloud service, multiple users can collaboratively create and edit a document within a cloud computing system. Since the

document may contain confidential information, the cloud service provider must not be able to understand the meaning of the document while the document is processed in the cloud computing system.

We also implemented *Method C)* of Data Obfuscator and De-obfuscator discussed in Section 5. A virtual machine was set up using Microsoft Hyper-V Manager 6.0, which had 0.7 GHZ CPU, 256 MB memory and Windows Server 2003 OS. The collaborative online documentation cloud service and data obfuscator are deployed on the virtual machine, and the data De-obfuscator is deployed on the user's computer as shown in Fig.4.

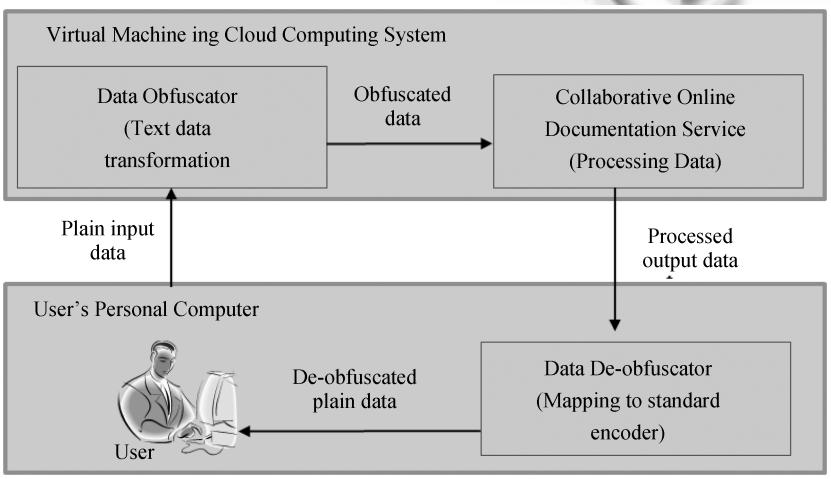

Figure 4. An example of collaborative online documentation service with Data Obfuscator and Data De-obfuscator

In order to measure the performance of *Method C)* of our data obfuscator and de-obfuscator, we ran two sets of experiments. In both sets of experiments, a set of input texts with various sizes are sent to the cloud service for processing and the service response time for each input was measured. In the first set of experiments, we used the cloud service without using the Data Obfuscator and De-obfuscator. In the second set of experiments, we used the cloud service along with *Method C)* of Data Obfuscator and De-obfuscator. 100% of the users' input text was specified as confidential data so that all of the input data is obfuscated and de-obfuscated during the service processing. The measured service response time for each set of experiment is shown in Fig.5.

Our experimental result shows that the service response time always increases linearly as the size of the input text increases in both sets of experiments. This means that the additional overhead for data obfuscation and de-obfuscation appears to be propotional to the size of the input text specified as confidential. Hence, our data obfuscation and de-obfuscation methods can perform in polynomial time and scalable.

In our approach, performing data obfuscation and de-obfuscation is the steps producing most overhead. Since the experimental results show that the data obfuscation and de-obfuscation do not cause much overhead, and hence our approach has reasonable performance.

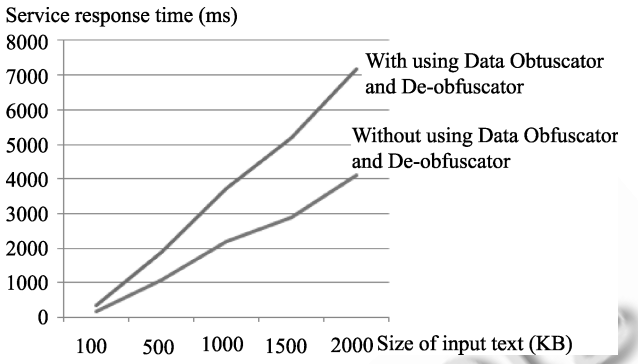

Figure 5. Service response time of our collaborative online documentation service with various sizes of input texts

8 Conclusion and Future Research

In this paper, we have presented an approach to protecting users' confidential data in cloud computing from cloud service providers. Our approach is based on new cloud system architecture, which has three features: *1)* separation of software service providers and infrastructure service providers, *2)* hiding information about the owner of data and *3)* data obfuscation. Our cloud system architecture ensures that cloud service providers cannot know location of the users' data, access the user's data, or understand the meaning of the user's data simultaneously. Our experimental results on the performance of our data obfuscation and de-obfuscation show that the overhead for data obfuscation and de-obfuscation aapears to increase linear with the size of input data. Hence, our approach is scalable with size of input data.

Our future research includes incorporation of the dynamic resource allocation[17] in our cloud computing architecture to support multiple QoS, including security aspects.

References

[1] Horrigan J. Use of cloud computing applications and services. Pew Internet and American Life Project Memo. 2008.

[2] Heiser J, Nicolett M. Assessing the security risks of cloud computing. Gartner Report, 2009, http://www.gartner.com/DisplayDocument?id=685308.

[3] DoD Trusted Computer System Evaluation Criteria, http://csrc.nist.gov/publications/history/dod85.pdf

[4] Iwaihara M, Murakami K, Ahn GJ, et al. Risk evaluation for personal identity management based on privacy attribute ontology. Proc. 27th Int'l Conf. on Conceptual Modeling (ER 2008). 2008. 183–198.

[5] Mateas M, Michael N. A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics. Proc. 6th Digital Arts and Culture Conference. 2005. 144–153.

[6] Ertaul L, Venkatesh S. Novel obfuscation algorithms for software security. Proc. Int'l Conf. on Software Engineering Research and Practice. 2005. 209–215.

[7] Dong W, Yu H. Web service testing method based on fault-coverage. Proc. 10th IEEE Int'l Enterprise Distributed Object Computing Conference Workshops. 2006. 43–49.

[8] Gibson J. Developing A requirements specification for a web service application. Proc. 12th IEEE Int'l Conf. Requirements Engineering. 2004. 340–344.

[9] Kona S, Bansal A, Gupta G, et al. Web service discovery and composition using USDL. Proc.

3rd IEEE Int'l Conf. E-Commerce Technology. 2006. 65–67.

[10] Damodaram A, Jayasri H. Authenticatio without identification using anonymous credential system. Int'l Jour. Computer Science and Information Security (IJCSIS), 2009, 3(1): 34–37.

[11] Yau SS, Zhu L, Huang D, Gong H. An approach to automated agent deployment in service-based systems. Proc. 10th IEEE Int'l Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC). 2007. 257–264.

[12] Mowbray M, Pearson S, A client-based privacy manager for cloud computing. Proc. Conf. Communication System Software and Middleware. 2009. 138–145.

[13] Iwaihara M, Murakami K, Ahn GJ, et al. Risk evaluation for personal identity management based on privacy attribute ontology. Proc. 27th Int'l Conf. Conceptual Modeling (ER 2008). 2008. 183–198.

[14] Yau SS, Yin Y. A privacy preserving repository for data integration across data sharing services. IEEE Trans. Services Computing, 2008, 1(3): 130–140.

[15] Yau SS and Huang J. A user-centric approach to assessing confidentiality and integrity of service-based workflows. Proc. 3rd Int'l Conf. Human-centric Computing (HumanCom). 2010. 89–94.

[16] Yau SS, An HG. Protection of users' data confidentiality in cloud computing. Proc. 2nd Asia-Pacific Symposium on Internetware. 2010. 32–37.

[17] Yau SS, An HG. Adaptive resource allocation for service-based systems. International Journal of Software and Informatics, 2009, 3(4): 483–499.