How Not to Be Seen in the Cloud: A Progressive Privacy Solution for Desktop-as-a-Service

D. Davide Lamanna, Giorgia Lodi, and Roberto Baldoni

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti"
Sapienza University of Rome
{lamanna,lodi,baldoni}@dis.uniroma1.it

Abstract. In public clouds, where data are provided to an infrastructure hosted outside user's premises, privacy issues come to the forefront. The right to act without observation becomes even more important in Desktop-as-a-Service (DaaS) environments. This paper describes the design, implementation and preliminary experimental evaluation of a progressive privacy solution for a DaaS system. Progressive privacy is a privacy preserving model which can be configurable (possibly on-demand) by a user not only quantitatively but rather qualitatively, i.e., the user is allowed to discriminate what type of information must be preserved and to what extent, according to her/his desired profiles of privacy. To this end, a lightweight client-side proxy named Hedge Proxy has been designed such that non-intelligible user contents and non-traceable user actions are guaranteed by enabling homomorphic encryption, oblivious transfer and query obfuscation schemes in the proxy. The paper also proposes an implementation and evaluation of the Hedge Proxy based on a specific DaaS environment developed at the University of Rome and called Virtual Distro Dispatcher (VDD). Preliminary results of such evaluation are presented and aim at assessing the performances experienced by users of VDD against the progressive privacy achievements that can be obtained. As expected, the perceived client performances when using VDD highly decrease when augmenting the level of privacy protection (e.g., using large key encryption size, high obfuscation density). Nevertheless, experiments show that for light encrypted data streams the system can reach fair level of privacy with small keys without significantly deteriorating user experienced performances.

Keywords: Privacy, Desktop-as-a-Service, thin client, visualization, homomorphic encryption, oblivious transfer, query obfuscation.

1 Introduction

Public clouds, where software applications and data are stored in off-premises data centers, are increasingly used for running large-scale distributed computations on data coming from heterogeneous sources. Data intensive elaborations are enabled through virtualization techniques that provide users with large volume of computational resources and storage, made available only when required.

Users can send their own data to the cloud and obtain in return a number of services (e.g., email services, financial services, online repositories, desktops).

Despite the high potential offered by this dynamic data processing and resource management model, cloud computing adoption is still limited. People, and also different types of organizations (e.g., financial stakeholders), are reluctant to provide their data to an external infrastructure, for which they do not have any control on where the data are located and how are processed. With the cloud model, the risks that privacy of data (even sensitive such as credit reports or heath care data) can be compromised by unauthorized parties and external threats augment [17].

As reported by the recent Berkley's vision of the cloud [3], one of the obstacles to the growth of cloud computing concerns the level of privacy that can be guaranteed when storing and processing the data in the cloud. This obstacle can be twofold; there exists an issue of data retention: where (in which country) data in the cloud are processed and to which legislation they are subject need to be addressed in order to respect the correct countries' legislations. Secondly, there exists a clear issue concerning the privacy of data which can be compromised by other users of cloud resources as well as by cloud providers themselves that actually perform computations on those data. Moreover, up-to-now privacy in today's Internet is mainly based on all-or-nothing trust relationships [6]. In contrast, a privacy model should be user-centric and configurable at the finest level of granularity as possible. For example a user should be able to specify different privacy policies for distinct actions (e.g. execution of applications) and/or user data.

This paper describes an architectural framework and related mechanisms capable of preserving the privacy of either (i) sensitive and personal user data, (ii) or the sequence of actions executed by users or (iii) both of them when users interact with cloud services hosted at cloud providers' premises. Among all available cloud services, the paper concentrates its attention on Desktop-as-a-Service (DaaS), hosted in public clouds.

In DaaS, on-demand user-configured desktops are directly access in the network by users who perform activities with them via end devices such as robust and low-maintenance thin clients. In DaaS there are two kinds of interactions: (i) the thin client receives a continuous stream of data (i.e., the graphic stream) and (ii) the client invokes commands in the form of *action*(*data*) where user data can be personal photos, files etc., and actions performed on those data can be visualize photos, store files, etc.; typically, the execution of these commands influence the graphic stream exchange. The commands and streams if not adequately protected, can be easily seen and traceable by cloud providers and other users sharing the same resources and service in the cloud, thus allowing them to infer personal identifiable information.

In order to preserve privacy in such a context, the paper proposes the so-called *progressive privacy* cloud solution. Progressive privacy is a privacy preserving model which can be fully configurable not only quantitatively but qualitatively; that is, users are allowed to discriminate what type of information must be privacy-preserved and to what extent, according to user desired profiles of privacy. The solution consists in installing at client terminals a small software program that acts as proxy (i.e., the Hedge Proxy): it intercepts all user actions carried out with DaaS and, according to desired privacy profiles, it enables the proper privacy policies capable of enforcing such profiles. Four policies have been defined; these policies progressively increment the offered privacy protection through state of the art encryption schemes that ensure strong security guarantees. The schemes are homomorphic encryption, oblivious transfer and query obfuscation, which are used for preserving privacy of data, actions, and data and actions together.

In order to show the feasibility and effectiveness of the proposed approach, the model has been implemented in the context of a DaaS service developed at University of Rome and named Virtual Distro Dispatcher (VDD) [4],[9],[16]. A preliminary experimental evaluation of that implementation has been carried out. Specifically, it has been evaluated the client perceived response time when using VDD in case homomorphic encryption is enabled by the Hedge Proxy for preserving privacy of personal data. The response time has been assessed by varying the dimensions of the data to be protected and the level of guaranteed privacy, measured as key encryption size. Costs estimation of the encryption schemes used for hiding the actions performed on data are also discussed. In general, it emerged that the performances of VDD highly decreased when both the size of data and the level of protection (key encryption and obfuscation density) augmented. In other words, there exists a trade-off between the privacy level and performances: the more robust required privacy profile, the strongest enabled privacy policy (large key size and high obfuscation density), the higher performances experienced by users when they interact with VDD. However, despite this tradeoff, experiments pointed out that a user can achieve a fair level of privacy with a reasonable overhead in terms of operation latency, when "light" data streams are exchanged (i.e., graphic streams with low resolution frames encrypted with keys up to 128 bits). This latter result is in accordance with other performance analysis in the field [18].

The rest of this paper is structured as follows. Section 2 discusses a number of research works for preserving data privacy in cloud systems. Section 3 introduces personal identifiable information in DaaS systems and it discusses a taxonomy of the types of data to be maintained private, the possible computations on those data, and privacy requirements that should be met. Section 4 describes the proposed progressive privacy cloud solution and in particular the used encryption schemes, the architecture of the solution, and the principal privacy policies that are supported by the solution. Section 5 discusses the implementation of the progressive privacy in VDD and presents the main results of an experimental evaluation of it. Finally, Section 6 concludes the paper and outlines future works.

2 Related Work

Providing privacy preserving mechanisms in public clouds is still an open challenge. However, a number of research works recently appeared in the literature.

These works are here assessed by considering three different points of view; namely, server versus client side solutions and privacy customization solutions.

Server-Side Solutions. In server-side solutions, privacy mechanisms are enabled by both Cloud Providers and Third Parties. Specifically, in [5] the authors propose an approach that separates competences. Hence, there exist Identity Providers (IdPs), Cloud Service Providers (CSPs), Registrars and final clients. CSPs provide access to data and software that reside on the Internet; IdPs issue certified identity attributes to clients and control the sharing of such information; Registrars are components that store and manage information related to identity attributes. A drawback of this system emerges in case multiple transactions are carried out by the same client with the same CSP. This CSP can easily link transactions and determine the common origin, thus profiling the client. On top of that, different CSPs may also collude and determine a profile of the transactions carried out by the same client. Such information, when combined with other available client information, may lead to disclosing the actual client identity, thus violating her/his privacy.

In [28] a Third Party Auditor (TPA) is used for verifying service quality from an objective and independent perspective. However, introducing a third-party just postpones the problem (e.g. High-Assurance Remote Server Attestation [8]. Delegate to TPAs might be reasonable from a technical point of view, as clients could not able to commit necessary computation resources; nevertheless, this imposes to presume trustworthiness of such services, which can not necessarily be expected.

Client-Side Solutions. In contrast to the previous systems, client-centric approaches to distributed privacy are proposed in which the clients themselves act as privacy managers. To this end, there exist research works based on P2P systems, where the content is shared only with peers and through trusted proxies under client's control. For example, [19] presents a mechanism to protect privacy in dynamic communication environments, in which the association between the identity of the user and data (s)he is interested in is hidden. Such mechanism is based on proxies used as shields and collaboration among peers. In the cloud computing context, P2P-based systems for privacy preserving can be ideal for Communication-as-a-Service (CaaS), where the scope of the service is the interaction among clients, and client relationships can be unmediated. However, in DaaS the client does not share her/his data with entities different than her/his service provider.

In addition, most of the client-side solutions may require to enable cloud computing clients to be their own privacy managers [6]. This may bring to a higher processing power necessary at the endpoints in order to carry out privacy management functionalities. In DaaS, this may be a concern due to the limited resources available at the client side. There is usually a tradeoff between the extent to which data are kept private and the set of applications that can be effectively used, unless the service provider cooperates in guaranteeing privacy of sensitive data [20]. When limited resources are available at the client side, soft obfuscation techniques can be employed in order not to require high processing power [22].

Privacy Customization. In [14], three levels of privacy for different types of data are discussed; namely, *full trust*, where insensitive data can be safely stored and processed in clear on the cloud; *compliance-based trust*, where data need to be stored encrypted (using a provider-specific cryptographic key) in order to support legal compliance regulations; and *no trust* where highly-sensitive customer data should be stored encrypted using customer-trusted cryptographic keys and processed in isolated cryptographic containers in the cloud.

In addition, user-centric identity management can be required [7]. In the Privacy Manager introduced in [23] the user can choose between multiple personae when interacting with cloud services. In some contexts, a user might want to be anonymous, and in others (s)he might wish for partial or full disclosure of her/his identity.

Another example of privacy customization can be drawn from Terra [11], a trusted virtual machine monitor (TVMM) that partitions the platform into multiple, isolated VMs. Each VM can customize its software stack to its security and privacy requirements.

Let us finally remark as the objectives of this paper are complementary to the ones pursued in [2] where the authors target organizations moving to the cloud, or outsourcing their services, that wish to access or allocate virtual resources anonymously. In this paper we wish the provider is not able to retrieve data and the application used by a client while it is allowed to detect who has assigned a resource.

3 Personal Identifiable Information in DaaS

In DaaS, on-demand user-configured desktops are hosted in public clouds. Users can directly access their desktops and personal data in the network, and perform computations with them via end devices such as robust and low-maintenance thin clients.

When refer to a person, data *content* and *computation* could produce information useful to identify, contact or locate a single individual. This information is usually referred to as *Personal Identifiable Information.*

In a public cloud context, the risks that such information can be inferred by cloud providers augment as clients do not have any control on where and how the data are stored and processed at cloud premises.

The next subsections discuss data typologies and data computations in the DaaS context, and a number of requirements that once met can guarantee cloud clients that their personal identifiable information will be kept private.

3.1 Data Typologies

Data generated, consciously or not, by a person, and hence identifying and qualifying her/him, can be classified in the following typologies. Note that these typologies are sufficiently general to be defined in any type of systems, not necessarily DaaS systems.

User Registration Data (URD) it is a type of data that can be requested by a service provider and produced (provided) at a certain instant on demand, often in a short time. Examples of URD can be name and address provided by clients to cloud providers when they register to a DaaS service;

User Generated Content (UGC) it is a type of data produced by a person at a certain time instant and for a given purpose; it is usually saved (intentionally) in a (private) storage system. In DaaS, a typical example of UGC can be a personal file which is saved in a desktop folder and contains a text document or a photograph;

User Configuration Data (UCD) it is a type of data that can be consciously produced by a person every time (at least once) (s)he wishes to adapt the system service to her/his personal needs or preferences. In DaaS, clients can personalize their desktops with specific configurations: for instance, they can use their favorite colors for the layout of the folders, or they can use a personal photo as background image. These data are typically considered UCD.

User Log Data (ULD) it is a type of data that can be unconsciously produced by a person while using system services, and can be maintained for audit purposes. In DaaS, it can be the set of logs maintained by cloud providers, used for tracking all the actions clients perform when they use their desktops.

3.2 Data Computation

The earlier introduced data are elaborated by both cloud providers and clients in order to provide and use cloud services, respectively.

In DaaS, URD can be usually *sent* by clients to cloud providers and *stored* by cloud providers for administrative purposes. UGC, such as a personal file created by a client, can be *stored* by a cloud provider and *open* by that client when (s)he uses the desktop. UCD, such as a personal photo used as background of a customized desktop, can be *stored* by a cloud provider and *visualized* at the client side when the client accesses her/his desktop. Finally, ULD that keep track of all the computations the clients perform with their desktops can be *registered* by cloud providers and *saved* on stable storage for auditing purposes.

3.3 Privacy Requirements

In order for PII not to be inferred from any of the above data typologies and data computations, and thus to preserve privacy of the clients when they use DaaS services, the following requirements should be effectively met:

- UGC must not be seen or reconstructed by any cloud provider; however clients need to freely perform a number of computations on them. For instance, the client wishes to open her/his file which contains her/his clinical records; however, the cloud provider is not able to read the information included in that file.
- UCD must not be seen or reconstructed by any cloud provider; however clients need to freely perform a number of activities on them. The client visualizes her/his personal photo as background of the desktop; however, the cloud provider is not able to see that photo.
- ULD must not be reconstructed by any cloud provider.
- URD can be seen by a cloud provider provided that they are not associated with the previous data typologies. For instance, the cloud provider can see that "Joan Smith" is registered to the DaaS service; however, if this information is not associated with other Joan Smith's data, the cloud provider is not able to profile her and derive her personal identifiable information.

Owing to the earlier requirements, we aim at ensuring privacy of clients when they interact with cloud providers, only; that is, our goal is to guarantee that the client trusts the cloud provider and not vice versa. In addition, no interactions among potential DaaS clients in the cloud exist; thus, trust relationships are established between a single client and the provider.

4 Progressive Privacy Architecture in DaaS

The privacy preserving model proposed in this paper is called *Progressive privacy*. The model allows clients of cloud services to configure, not just quantitatively (as in [14]), but rather qualitatively the privacy that can be achieved, i.e. what type of information must be preserved and to what extent. In doing so, the progressive privacy model permits to incrementally meet all the privacy requirements described in the previous section. This is done through the definition of a set of privacy policies that can be progressively activated on demand according to client desired privacy profiles. The policies are implemented using state-of-the-art encryption schemes increasingly used even in the cloud computing context.

The next subsections introduce these schemes, the architecture of the progressive privacy model for DaaS cloud services, and the privacy policies that can be incrementally enabled in order to meet the above requirements.

4.1 Encryption Schemes

Three main encryption schemes have been used in the progressive privacy solution; namely, Homomorphic Encryption, Oblivious Transfer and Query Obfuscation.

Homomorphic Encryption. It is an encryption scheme where a specific algebraic operation is performed on the plaintext and another (possibly different) algebraic operation is performed on the ciphertext. Consider the following scenario. A user owns a message x (s)he wishes to keep private while copying it at a service provider space. Traditional encryption schemes are sufficient to meet the privacy requirement. However, with traditional schemes if the user wishes to perform some function f on x a decryption of x server-side is required, thus revealing x to the service provider and violating user's privacy.

With homomorphic encryption it is possible to copy x in an encrypted form at the service provider space; the user performs any f on it using a public-key only and obtains a result which is the encrypted value of *f(x)* [10]. In this way, the user can let the service provider work on her/his data that are kept private, as the service provider has no means to read x.

Constructing fully homomorphic encryption with polynomial overheads remained an open challenge for a long time [24]. Only recently Gentry proposed an implementation of a fully homomorphic encryption scheme based on lattice [12] and subsequent improvements were proposed [26],[25]. Despite their strong security guarantees which allow them to resolve potentially any privacy problem, these protocols (including their lightweight versions) still employ computationally expensive public-key operations. They are thus characterized by very high overheads, especially in case of large input data and key encryption sizes. This is the main reason for which some research works [13] [24] consider these protocols unfeasible in cloud services, since such overheads may cancel the benefits of outsourcing data and/or software computations. However, in contrast to these negative results, other recent researches [18] are more optimistic on the future of homomorphic encryption in clouds. In particular, [18] indicates that for both small key encryption sizes and specific types of applications homomorphic encryption can be used. In particular, [18] argues that for applications such as searching images or videos (which can be used in DaaS systems) it may be acceptable to lose accuracy in the exchange for better performance and protection of privacy. These latter observations are also in accordance with the results obtained by the experimental evaluation discussed in Section 5.4.

Oblivious Transfer. Once the data are protected using homomorphic encryption schemes, it may be required that also the function f representing a computation to be performed on the encrypted content is to be maintained private. Several approaches exist in the literature that make hard or even impossible to track computations on data. The most interesting cryptographic scheme in this context is the oblivious transfer [21] scheme. In oblivious transfer one party, the sender, transmits some information to another party, the chooser, in a manner that protects both of them: the chooser is assured that the sender does not learn which part of the information it is received, while the sender is assured that the chooser does not receive more information than that it is entitled to.

Note that in the context of privacy preserving in DaaS, the latter guarantee can be relaxed since all the information is owned by the chooser (the cloud client) or openly offered to the chooser (i.e. applications of the Cloud system). In DaaS, only the first oblivious transfer guarantee is relevant; that is, the provider does not learn what piece of data is requested and what function is requested to be executed on it.

Query Obfuscation. It is another privacy preserving scheme [15] that can be successfully used in case it is required to guarantee privacy of data computations. IT consists in introducing noisy or fake queries at random or mixing queries from different users so that cloud providers are not able to know which actions are executing on the data.

4.2 The Architecture

In DaaS cloud services, interactions between clients and cloud providers are in the form of $action(data)$ and $reply(data)$, as illustrated in Figure 1. These interactions are managed using two types of communication channels established between a client and a cloud provider: a graphics channel over which graphical information, for graphical elaboration and rendering, is sent to the desktop and back to the clients, and a command channel over which sending commands to be executed on personal client data and receiving the results of the execution on the data (see Figure 1). For instance, a type of action could be launching an application, uploading/downloading a file, whereas the data could be the url of a web site or the file saved by a text editor application.

To preserve privacy of data and actions, a client-side privacy system has been designed. The system consists of a proxy named *Hedge Proxy* (see Figure 1), installed in client end devices (client terminals are currently equipped with sufficient computational resources to be able to host small software programs such as proxies). Through the proxy, clients can specify a desired privacy profile. For instance, a client may require that only the privacy of her/his personal data is preserved, and/or only the privacy of specific typologies of actions is to be guaranteed. For doing so, the proxy is structured as follows.

It embodies a table named *Privacy Table.* The table expresses the user desired privacy profile the Hedge Proxy has to enforce. Specifically, clients are allowed to "fine tune" their privacy requirements at the level of single actions and/or data (see Figure 2). This tuning is maintained through the privacy table that contains

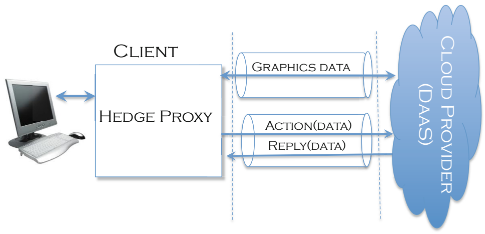

Fig. 1. Progressive Privacy architecture in DaaS

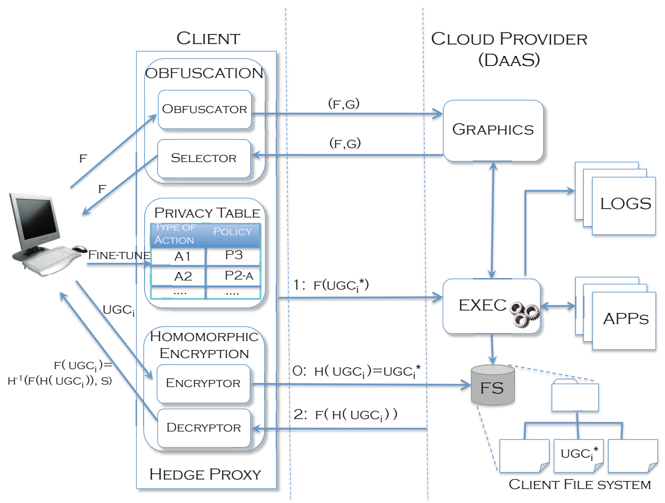

Fig. 2. Hedge Proxy and privacy-preserved interactions

all the possible associations between the types of actions the user perform with DaaS and privacy policies (see Section 4.3) that are to be applied on those actions and data.

The associations <type of actions/policy > included in the privacy table are respected by the proxy through the use of two modules, that are part of it; namely the *Homomorphic Encryption* and *Obfuscation* modules. The modules are mainly responsible for activating the privacy policies as specified in the privacy table.

Homomorphic Encryption module it is responsible for guaranteeing privacy of data (i.e., UGC, UCD, URD), only. It implements the homomorphic encryption scheme previously described in order to encrypt the data to be sent to the cloud provider and decrypt them when they are sent back to the client terminal.

Data are stored encrypted at the cloud provider and a secret s is provided to the client. In particular, Figure 2 depicts the interactions between a client i and a cloud provider in case a UGC is to be privacy preserved and transferred from the client to the server.

At time 0, the encryptor sub module applies the homomorphic encryption function h to the UGC, thus obtaining an encrypted UGC_i, indicated as UGC_i^* in Figure 2. UGC_i^* is then stored in the home in the client file system.

At time 1, client i requests a desktop action f to be performed on the encrypted UGC. At time 2, DaaS replies with the execution of such a function on the encrypted data. In this latter case, the decryptor sub module is activated and, thanks to the homomorphic property, it outputs the result of the application of the f action on UGC_i. This is done by inverting the function h by means of the secret s i.e. the private key of the client (in an asymmetric crypto system such as the one proposed in Section 5.2); that is,

$$f(UGC_i) = h^{-1}(f(h(UGC_i)), s)$$

Each client sends the public key to the provider, only. Private keys are stored in the proxy and are generated, managed and updated locally at client side.

Obfuscation module it is responsible for guaranteeing privacy of actions on the data, only. Specifically, this module implements a hybrid privacy preserving scheme that combines both oblivious transfer and query obfuscation schemes. When an action is requested by the user, the obfuscator sub module appends fake action requests. The selector submodule is then responsible for filtering out useless output from cloud provider responses.

Referring to upper level of Figure 2, an f function is requested by the client to be performed. The obfuscation module works as follows. Two actions f and g (the latter being a fake) are issued by the obfuscator sub module to the DaaS cloud provider (note that this is just an example; the number of fake actions can be arbitrarily high according to the desired level of protection). Both functions are executed by the server. However, the thin client is able to show only the output of action f thanks to the execution of the selector sub module that filters the fake actions.

The obfuscation module has been designed so that it is possible to choose the percentage of the actions having one or more appended fakes and the number of such fakes. These values determine different levels of obfuscation (i.e., obfuscation density) that can be achieved, and thus stronger levels of privacy protection (see Section 5.4 for an assessment of these values).

4.3 Privacy Policies

In the progressive privacy model, the following four principal privacy policies have been defined:

- *P0*: P0 does not guarantee any privacy of data and actions, when it is selected by cloud clients as privacy policy. The Hedge proxy of Figure 2 is not activated when the client uses her/his desktop.
- *P1*: P1 guarantees privacy of UGC and UCD, only, when it is selected by cloud clients in the interactions with cloud providers. In this case, the Hedge proxy at the client side is activated: the Homomorphic Encryption module is enabled in order to enforce this policy. Owing to the "fine tune" feature offered by the proposed progressive privacy solution, P1 policy can be (i) requested on demand for a specific data (e.g., a client wishes to mask her/his desktop background photo, only); and (ii) selected for all possible data.
- *P2*: P2 guarantees privacy of the actions (or computations) performed on data, only, when it is selected by cloud clients. In this case, the Hedge proxy at the client side enables the Obfuscation module responsible for enforcing

this policy. Since the cloud provider can not be sure about the actions the clients perform when they use their desktops (whether they are a fake or not), this policy guarantees that collected ULD are meaningless. Once again, owing to the "fine tune" feature of the progressive privacy solution, the P2 policy can be (i) requested on demand on a specific action (e.g., a client wishes that only "open file(creditAccount)" is privacy protected); (ii) selected to be applied to all the same typologies of actions (e.g., a client wishes that all "visualize photo" actions are to be privacy protected, only); (iii) applied to all the possible actions users can perform with DaaS.
- *P3*: P3 is the strongest privacy policy; when selected by clients, it ensures that both P1 and P2 are applied to data and actions on the data. The Hedge proxy enables the Homomorphic Encryption as well as the Obfuscation module in the interactions between the client and cloud provider, as indicated by the privacy table of the Hedge Proxy.

A privacy policy applied to data can be specified by the user at any action without compromising the level of data protection, i.e., a client can decide to store at the cloud provider site an encrypted file whereas (s)he can decide not to encrypt other files. This on-demand behavior can be also employed by a client to apply different policies when issuing each action. However, in this latter case, owing to the characteristics of the obfuscation technique, the level of protection offered by the policy might be reduced.

Referring to the data typologies of Section 3.1, Table 1 summarizes the different main privacy profiles that can be guaranteed using the different privacy policies. Note that if P3 Policy is applied by a user to any action, there is no need to worry about URD. In this condition, the cloud provider is not able to associate any meaningful types of actions or data with a user. In all other cases, URD can be associated to all or part of the other data, thus inferring more information about the user.

On the other hand, URD must be plain text by definition, as, when requested, they are needed for administrative purposes by cloud provider; hence having them hidden (encrypted or obfuscated) to cloud provider would make it pointless to ask for them.

Finally, it is worth noticing that, when the strongest privacy profile is selected by a client, there is no possibility to trace clients' actions from the cloud provider. Only clients own the keys for reconstructing all the information and when HE is enabled, the information is encrypted while it is treated by the server. This

Table 1. Progressive privacy profiles

	P0	P1	P2	P3
UGC	X	√	X	√
UCD	X	√	X	√
ULD	X	X	√	√

also implies that malicious security attacks carried out against the system and in particular man-in-the middle attacks cannot compromise the privacy of the client information. To the best of our knowledge, the only way of violating client privacy is stealing his/her key at client-side or using brute force attacks on encrypted messages. In countries or situations where legislation would impose some external regulators to acquire information on specific activities, it is always possible to employ a Trusted Third Party in the architecture that maintains those important elements of the implemented encryption schemes necessary for tracing the required information.

5 Progressive Privacy Implementation

The progressive privacy approach discussed in the previous section has been implemented in the context of a DaaS service, developed by the authors and named Virtual Distro Dispatcher (VDD). This section summarizes the main characteristics of VDD and presents the implementation and experimental evaluation of Progressive Privacy in VDD.

5.1 Virtual Distro Dispatcher

VDD is a distributed system whose aim is to project virtual, fully operational and multiple operating system instances on terminals in a network. Client terminals can be random PCs or energy saving thin clients (such as mini-ITX) managed by a powerful, multiprocessor (and possibly clustered) central system. Desktops are instantiated on a server and then provided to thin clients on demand across a network [4],[9],[16]. VDD is a Desktop-as-a-Service (DaaS) solution: desktops can be transformed into a cost-effective, scalable and comfortable subscription service.

VDD gives users the possibility to use their own favorite operating systems, at the same time, on each single thin client. Thin clients are interfaces to proper and isolated machines, that can be made to measure for whatever need and in whatever number (within server limits, of course). This is completely transparent to users, who, even from an obsolete machine, can select a particular machine with certain characteristics and then do everything they would do on such a machine as if it was physical and with its defined performance. Contrary to other systems such as LTSP (Linux Terminal Server Project), VDD offers not only the host operating system to thin clients (e.g., GNU/Linux), but projects virtualized guest systems, i.e. fully operational and independent machines, equipped with diversified desktop systems.

5.2 Implementation

The Hedge proxy has been implemented in C and tested at client end-devices. In particular, the Paillier cryptosystem [1] has been used for the implementation of the Homomorphic Encryption module of the Hedge proxy. This system offers an additive homomorphism: the product of two cipher texts are decrypted to the sum of their corresponding plain texts. A GPL-licensed C library (i.e., libpaillier-0.8), which implements Paillier key generation, encryption, decryption, has been chosen to introduce the homomorphic properties. The GNU Multiple Precision Arithmetic Library (GMP) has been used for the underlying number theoretic operations.

5.3 Testbed and Data Stream

A number of tests have been conducted in order to show the overheads introduced in VDD when augmenting the level of privacy; that is, when the different privacy policies defined in Section 4.3 are applied to data and actions performed on data. A testbed has been appropriately adapted and prepared from the development environment of VDD; it consists of 2 servers and 10 thin clients. The hardware and software characteristics of the testbed are illustrated in Tables 2 and 3.

DaaS data stream corresponds to data exchanged between clients and cloud provider for the client to visualize what the server elaborate. The data stream is considered in terms of bandwidth consumption and frame dimension. Bandwidth consumption for visualization have been taken from the estimation of the developers of VNC, i.e., a remote control software which allows users to interact with desktop applications across any network [27]. In particular, the tests have been carried out considering the following three bandwidth values: 33Kbps, sufficient for accessing desktops with very simple imagery, with reduced colour resolution and frame-rate; 128Kbps, sufficient for accessing desktops with simple imagery with reduced colour resolution; 1Mbps, sufficient for accessing desktops with simple imagery in full colour, or complex imagery with reduced colour or frame-rate.

Table 2. Server hardware configuration

CPU	Intel core i7 920 @ 2.67GHz 8MB SK1366 RET
RAM	8GB (4 DIMM KINGSTON 2GB PC1333 DDR III CL9)
Disks	3 Western Digital hard disks, RAPTOR 74GB 10000rpm 16MB SATA
OS arch	X86-64
OS	Gentoo Linux kernel 2.6.35-gentoo-r11

Table 3. Client hardware configuration

CPU	Intel(R) Core(TM)2 Quad CPU Q9400 @ 2.66GHz
RAM	4GB
Disks	Seagate Barracuda 7200.11 RAID 1
OS arch	X86-64
OS	Gentoo Linux kernel 2.6.35-gentoo-r11

As for frame dimension, three dimensions have been considered; namely, 10 KB, 100 KB and 1 MB.

5.4 Evaluation

With policy P0, i.e. no privacy, the time to transfer the earlier mentioned amounts of data (without compression) is: 2424 ms (10KB @ 33Kbps), 6250 ms (100KB @ 128Kbps), 8000ms (1MB @ 1Mbps).

Table 4 shows the impact on the latency in the stream when privacy policy P1 is selected by clients. Specifically, it illustrates the increase of the perceived response time at client side when both the key encryption size (64bit, 128bit, and 256bit) used by the homomorphic encryption module and the amount of data (10KB, 100KB, 1MB) transmitted from the client to the cloud provider augment.

Table 4. Client-side decryption latency (key size vs frame size)

	10KB	100KB	1MB
64 bit	2ms	30ms	327ms
128 bit	9ms	116ms	1104ms
256 bit	53ms	543ms	5510ms

Figure 3 shows the ratio between the values in Table 4 and time to transfer 10KB, 100KB, 1MB frame, varying the key size. From the results it emerges that improvements in the achievable level of privacy (i.e., the increase of the key encryption size) and/or in the quality of the visualization of the data (i.e., frame size), affect the overall VDD performance. This is principally due to on-the-fly decryption performed at the client side.

On the other hand, server side performances may also suffer for two reasons: (1) the server has to perform a number of operations on encrypted data in case P1 privacy policy is selected to be enforced; (2) potentially high fake action requests can be issued to the cloud provider in case P2 privacy policy is selected; this results in a waste of bandwidth and high local processing.

Figure 4 depicts the impact on the performances at the server side when executing operations on encrypted data. In order to evaluate this impact, the ratio between the execution of an adequately long series of multiplications of two big integer operands and the execution of the same operation with encrypted operands has been assessed and reported in Figure 4 when varying the level of encryption ensured by the homomorphic encryption module. This estimation can be used as the worsening factor for general execution of actions on encrypted data on that system (Figure 4). It appears clear that the impact of executing operations on encrypted data is quite severe.

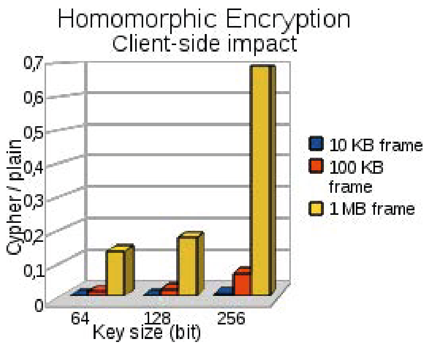

Fig. 3. Client-side impact

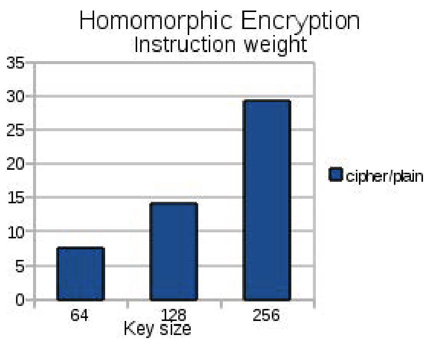

Fig. 4. Server-side impact

In case P2 privacy policy is selected by the client when using VDD, it is straightforward to evaluate the decay factor of the obfuscation by doing the following assumption: desktop operations, to be duplicated or triplicated with fakes, have equal impact on average. By varying the previously mentioned percentage of actions with one or more appended fakes and the number of such fakes, three obfuscation levels named Bronze, Silver and Gold, have been determined.

Hence, if x is the number of actions executed during a VDD session, k is the number of actions executed when the event of obfuscation occurs and E is the percentage of obfuscation events on the total x, a decay factor can be defined as follows:

$$Decayfactor = \left[\frac{x}{x + (k-1) \cdot E \cdot x}\right] = \frac{1}{1 + (k-1) \cdot E}$$

Three different decay factors for each level of obfuscation have been identified and reported in Table 5. When P2 is enabled, system load augments and system performance decays. The decay factor expresses such a decay over 1 (i.e. no decay). Hence, depending on how much obfuscation is done, performance decreases and decay factor tends to 0.

Finally, in case P3 privacy policy is selected by the client when using VDD, the impact client-side is the one depicted in Figure 3 (obfuscation does not take up client resources), whereas the impact server-side is the application in series of the two effects described above (Figure 4 and Table 5).

Table 5. Server-side performance decay due to obfuscation

Obfuscation level	Event occurrence	Fakes per event	Decay factor
Bronze	30% (E=.3)	k=2	0.77
Silver	30% (E=.3)	k=3	0.63
Gold	40% (E=.4)	k=3	0.56

In summary, progressive privacy implementation in VDD presents quite costly benefits, when data dimension and/or protection increase, especially for executing operations on encrypted data. This is mainly due to the fact that, at present, the implemented cryptographic protocols are quite demanding in terms of performances. However, the proposed solution shows that the system can reach fair level of privacy without significantly deteriorating user performances, thus allowing such protocols to be practically used even in the cloud computing context.

6 Concluding Remarks

The paper described the design, implementation and preliminary experimental evaluation of a so-called progressive privacy cloud solution. Progressive privacy is a privacy preserving model in which users of DaaS services can select different privacy profiles on demand to encrypt both data, actions and graphic streams. The profiles are enforced, transparently to the clients, by a lightweight software module, namely Hedge Proxy, installed at client end-devices. The software module progressively augments the level of privacy protection on both personal client data and actions on those data, according to desired user privacy requirements. A preliminary experimental evaluation has been carried out using a DaaS service implemented at University of Rome and named VDD. From the obtained results it emerged that there exists a clear trade-off between the privacy level that can be guaranteed and the degradation of the performances experienced by both the clients and server. The combination of contents/actions to be protected and the level of protection in terms of key size and obfuscation density are crucial parameters that must be properly defined in order to *practically* deploy a privacy preserving mechanism in the cloud. In fact, despite that tradeoff, experiments suggest that a user can achieve a fair level of privacy with a reasonable overhead in terms of operation latency, when "light" stream of data are exchanged between clients and cloud providers (e.g., graphic streams with low resolution frames encrypted with keys up to 128 bits).

The level of privacy protection that can be achieved so far, with respect to a brute force attack, is relatively low. Nevertheless, it is worth remarking that the proposed architecture is agnostic about future achievements (e.g., different encryption schemes) in cryptography. Thus, it could provide better results in terms of overall privacy protection as long as new software, hardware and protocols devoted to improve the encryption/decryption operations will be released. Considering the increased investments of large cloud providers (e.g., HP, IBM) in this direction a positive evolution of this process is likely to happen.

Currently, a massive series of experiments are being carried out showing which is the latency experienced by a user that is running common applications on DaaS, such as mailer programs, browsers etc, when increasing the number of virtual desktops delivered by the system. These results will be detailed in a future work; however the communication and computation costs would suggest to split users in separate classes where each user in a certain class can exploit the advantages of progressive privacy using different key sizes and obfuscation levels according to their status.

Acknowledgments. The authors would like to thank Flavio Bertini, Salvatore Cristofaro and Binario Etico, in particular Ruggero Russo, Fabrizio Nasti and Stefano Buscaglia, for their precious work on VDD.

References

1. Adida, B., Wikström, D.: How to Shuffle in Public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
2. Ateniese, G., Baldoni, R., Bonomi, S., Di Luna, G.: Oblivious Assignment with m Slots. Technical report, MIDLAB 2/12 - University of Rome La Sapienza (2012), `http://www.dis.uniroma1.it/mid-lab/publications.php`
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)
4. Bertini, F., Lamanna, D.D., Baldoni, R.: Virtual Distro Dispatcher: A Costless Distributed Virtual Environment from Trashware. In: Stojmenovic, I., Thulasiram, R.K., Yang, L.T., Jia, W., Guo, M., de Mello, R.F. (eds.) ISPA 2007. LNCS, vol. 4742, pp. 223–234. Springer, Heidelberg (2007)
5. Bertino, E., Paci, F., Ferrini, R., Shang, N.: Privacy-preserving digital identity management for cloud computing. IEEE Data Engineering Bull. 32(1), 21–27 (2009)
6. Camp, J.L.: Designing for trust. In: Proc. of the International Conference on Trust, Reputation, and Security: Theories and Practice (AAMAS 2002), pp. 15–29. ACM Press (2003)
7. Cavoukian, A.: Privacy in the Clouds: Privacy and Digital Identity-Implications for the Internet. Information and Privacy Commissioner of Ontario (2008)
8. Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., Molina, J.: Controlling data in the cloud: outsourcing computation without outsourcing control. In: Proc. of the ACM Workshop on Cloud Computing Security (CCSW 2009), pp. 85–90. ACM Press (2009)
9. Cristofaro, S., Bertini, F., Lamanna, D., Baldoni, R.: Virtual Distro Dispatcher: A Light-weight Desktop-as-a-Service Solution. In: Aversky, D.R., Diaz, M., Bode, A., Ciciani, B., Dekel, E. (eds.) Cloudcomp 2009. LNICST, vol. 34, pp. 247–260. Springer, Heidelberg (2010)
10. Fontaine, C., Galand, F.: A survey of homomorphic encryption for nonspecialists. EURASIP Journal on Information Security 15(1), 1–15 (2007)
11. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: A virtual machine-based platform for trusted computing. In: Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), pp. 193–206. ACM Press (2003)
12. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proc. of the 41st Annual ACM Symposium on Theory of Computing, pp. 169–178 (2009)
13. Huber, M.: Towards Secure Services in an Untrusted Environment. In: Proc. of the 15th International Workshop on Component-Oriented Programming, pp. 47–54 (2010)
14. Itani, W., Kayssi, A., Chehab, A.: Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures. In: Proc. of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing, pp. 711–716. IEEE Press (2009)

15. Jones, R., Kumar, R., Pang, B., Tomkins, A.: Vanity fair: privacy in querylog bundles. In: Proc. of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), pp. 853–862. ACM Press (2008)
16. Lamanna, D., Bertini, F., Cristofaro, S., Etico, B.: Vdd project (June 2007), `http://www.vdd-project.org/`
17. Lodi, G., Querzoni, L., Baldoni, R., Marchetti, M., Colajanni, M., Bortnikov, V., Chockler, G., Dekel, E., Laventman, G., Roytman, A.: Defending Financial Infrastructures Through Early Warning Systems: The Intelligence Cloud Approach. In: Proc. of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies (April 2009)
18. Liu, J., Lu, Y.H., Koh, C.K.: Performance Analysis of Arithmetic Operations in Homomorphic Encryption. ECE Technical Reports, Electrical and Computer Engineering. Purdue Libraries (2010)
19. Lu, Y., Wang, W., Bhargava, B., Xu, D.: Trust-based privacy preservation for peer-to-peer data sharing. IEEE Transactions on Systems, Man and Cybernetics 36(3), 498–502 (2006)
20. Mowbray, M., Pearson, S.: A client-based privacy manager for cloud computing. In: Proc. of the 4th International ICST Conference on COMmunication System softWAre and middlewaRE (COMSWARE 2009), pp. 1–8. ACM Press (2009)
21. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Proc. of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), pp. 448–457. SIAM Press (2001)
22. Oliveira, S.R.M., Zaïane, O.R.: Achieving Privacy Preservation when Sharing Data for Clustering. In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178, pp. 67–82. Springer, Heidelberg (2004)
23. Pearson, S., Shen, Y., Mowbray, M.: A Privacy Manager for Cloud Computing. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 90–106. Springer, Heidelberg (2009)
24. Sadeghi, A.-R., Schneider, T., Winandy, M.: Token-Based Cloud Computing - Secure Outsourcing of Data and Arbitrary Computations with Lower Latency. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) TRUST 2010. LNCS, vol. 6101, pp. 417–429. Springer, Heidelberg (2010)
25. Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
26. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
27. `http://www.realvnc.com/`
28. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009)