A Privacy Manager for Cloud Computing

Siani Pearson, Yun Shen, and Miranda Mowbray

HP Labs, Long Down Avenue, Stoke Gifford, Bristol BS34 8QZ, UK
{siani.pearson,yun.shen,miranda.mowbray}@hp.com

Abstract. We describe a privacy manager for cloud computing, which reduces the risk to the cloud computing user of their private data being stolen or misused, and also assists the cloud computing provider to conform to privacy law. We describe different possible architectures for privacy management in cloud computing; give an algebraic description of obfuscation, one of the features of the privacy manager; and describe how the privacy manager might be used to protect private metadata of online photos.

Keywords: Cloud computing, privacy.

1 Introduction

In this paper we describe a privacy manager for cloud computing, which reduces the risk to the cloud computing user of their private data being stolen or misused, and also assists the cloud computing provider to conform to privacy law.

Cloud computing, in which services are carried out on behalf of customers on hardware that the customers do not own or manage, is an increasingly fashionable business model. The input data for cloud services is uploaded by the user to the cloud, which means that they typically result in users' data being present in unencrypted form on a machine that the user does not own or control. This poses some inherent privacy challenges.

There is a risk of data theft from machines in the cloud, by rogue employees of cloud service providers or by data thieves breaking into service providers' machines, or even by other customers of the same service if there is inadequate separation of different customers' data in a machine that they share in the cloud. Governments in the countries where the data is processed or stored may have legal rights to view the data under some circumstances [1,2]. There is also a risk that the data may be put to unauthorized uses. It is part of the standard business model of cloud computing that the service provider may gain revenue from authorized secondary uses of the user's data, most commonly the targeting of advertisements. However, some secondary data uses would be very unwelcome to the data owner (such as, for example, the resale of detailed sales data to their competitors). At present there are no technological barriers to such secondary uses.

There are, however, some legal constraints on the treatment of users' private data by cloud computing providers. Privacy laws vary according to jurisdiction, but EU countries generally only allow personally-identifiable information to be processed if the data subject is aware of the processing and its purpose, and place special

M.G. Jaatun, G. Zhao, and C. Rong (Eds.): CloudCom 2009, LNCS 5931, pp. 90–106, 2009.

restrictions on the processing of sensitive data (for example, health or financial data), the explicit consent of the data owner being part of a sufficient justification for such processing [3]. They generally adhere to the concept of *data minimization*, that is, they require that personally identifiable information is not collected or processed unless that information is necessary to meet the stated purposes. In Europe, data subjects can refuse to allow their personally identifiable data to be used for marketing purposes [4]. Moreover, there may be requirements on the security and geographical location of the machines on which personally identifiable data is stored. A UK business processing data about individual customers with some cloud computing services could find itself in breach of UK data processing law, if these services do not give assurances that the machines they use are adequately secure [5]. European law limiting cross-border data transfers also might prohibit the use of the cloud computing services to process this data if they stored data in countries with weak privacy protection laws [6].

The structure of the paper is as follows. In section 2 we present our solution, which is in the form of a privacy manager for cloud computing. In section 3 we discuss different architectures for privacy management in cloud computing, giving an overview of how the privacy manager may be used. We also describe how Trusted Computing mechanisms [7] can optionally be used to enhance privacy management. In Section 4 we describe obfuscation in mathematical terms. We give an algebraic formulation of the task of obfuscation, and algebraic specifications of different obfuscation methods suitable for particular examples. In Section 5 we discuss an application scenario, the management of online photos, and present the user interface for the privacy manager for this application. In Section 6 we review previous approaches to privacy management for data repositories. The paper concludes with a general analysis and discussion of next steps.

2 Our Solution: Privacy Manager

Our contribution to addressing these problems is a Privacy Manager, which helps the user manage the privacy of their data in the cloud. As a first line of defence, the privacy manager uses a feature called *obfuscation*, where this is possible. The idea is that instead of being present unencrypted in the cloud, the user's private data is sent to the cloud in an encrypted form, and the processing is done on the encrypted data. The output of the processing is de-obfuscated by the privacy manager to reveal the correct result. (We call it obfuscation rather than encryption because some of the information present in the original data is in general still present in the obfuscated data.) The obfuscation method uses a key which is chosen by the user and known by the privacy manager, but which is not communicated to the service provider. Thus the service provider is not able to de-obfuscate the user's data, and the un-obfuscated data is never present on the service provider's machines. This reduces (or even eliminates) the risks of theft of this data from the cloud and unauthorized uses of this data. Moreover, the obfuscated data is not personally identifiable information, and so the service provider is not subject to the legal restrictions that apply to the processing of the un-obfuscated data. Where obfuscation is practical, the principle of data minimization gives a legal impetus to use it.

However, it is not practical for all cloud applications to work with obfuscated data. For applications for which users have to upload some private data to the cloud, the privacy manager contains two additional features, called *preferences* and *personae*, which help the users to communicate to service providers their wishes for the use of this personal data. These two features do not guarantee that a user's wishes will be observed if the service provider is not trustworthy. However they assist trustworthy service providers to respect privacy laws that require the user's consent.

The preferences feature allows users to set their preferences about the handling of personal data that is stored in an unobfuscated form in the cloud. A similar approach has been taken within P3P [8] and PRIME [9]. It communicates these preferences to a corresponding policy enforcement mechanism within the cloud service. The preferences can be associated with data sent to the cloud, and preferably cryptographically bound to it (by encrypting both the policy and data under a key shared by the sender and receiver). For stickiness of the privacy policy to the data, public key enveloping techniques can be used. Alternatively, it is possible to use policy-based encryption of credential blobs (a form of Identity-Based Encryption (IBE) technology) [10]: the policies could be used directly as IBE encryption keys to encrypt the transferred material [11]. Part of the preference specification could involve the purpose for which the personal data might be used within the cloud, and this could be checked within the cloud before access control were granted, using mechanisms specified via [12].

The persona feature allows the user to choose between multiple personae when interacting with cloud services. In some contexts a user might want to be anonymous, and in others he might wish for partial or full disclosure of his identity. The user's choice of persona provides a simple interface to a possibly complex set of data use preferences communicated to the service provider via the preference feature, and may also determine which data items are to be obfuscated.

This paper extends the basic idea of a client-based Privacy Manager introduced in [13]. We describe several different possible architectures for usage of a Privacy Manager in cloud computing, not just one in which the privacy manager is within the user's client; we show how trusted computing can be used to enhance this approach; we give a general mathematical description of the obfuscation mechanisms used within the Privacy Manager; and we describe an application to a particular scenario – photo management in the cloud – and a demonstration of the Privacy Manager that allows investigation of how the Privacy Manager can operate in practice to help users protect their private information in the cloud.

3 Architectural Options

In this section we describe different possible architectures for privacy management within cloud computing, and demonstrate how trusted computing can be used to strengthen this approach. The most appropriate architecture to be used depends upon the cloud infrastructure deployed for a particular environment, and the trust relationships between the parties involved.

3.1 Privacy Manager in the Client

The overall architecture of our solution is illustrated in Figure 1. Privacy Manager software on the client helps users to protect their privacy when accessing cloud services. A central feature of the Privacy Manager is that it can provide an obfuscation and de-obfuscation service, to reduce the amount of sensitive information held within the cloud. For further detail, see Section 4. In addition, the Privacy Manager allows the user to express privacy preferences about the treatment of their personal information, including the degree and type of obfuscation used. Personae – in the form of icons that correspond to sets of privacy preferences – can be used to simplify this process and make it more intuitive to the user. So for example, there could be an icon with a mask over a face that corresponds to maximal privacy settings, and other icons that relate to a lower level of protection of certain types of personal data in a given context. The user's personae will be defined by the cloud service interaction context. Personae may be defined by the user, although a range of default options would be available.

Trusted computing solutions, like those being developed by the Trusted Computing Group (TCG) [14], can address the lower-level protection of data, and this can be exploited in our solution. The TCG is an organization set up to design and develop specifications for computing platforms that create a foundation of trust for software processes, based on a small amount of extra hardware called a Trusted Platform Module (TPM) [14]. This tamper-resistant hardware component within a machine acts as a root of trust. In the longer term, as specified by TCG, trusted computing will provide cryptographic functionality, hardware-based protected storage of secrets, platform attestation and mechanisms for secure boot and integrity checking [7]. Allied protected computing environments under development by certain manufacturers and open source operating systems such as Linux can support TCG facilities further. For details about how trusted computing might be used to enhance privacy, see [15].

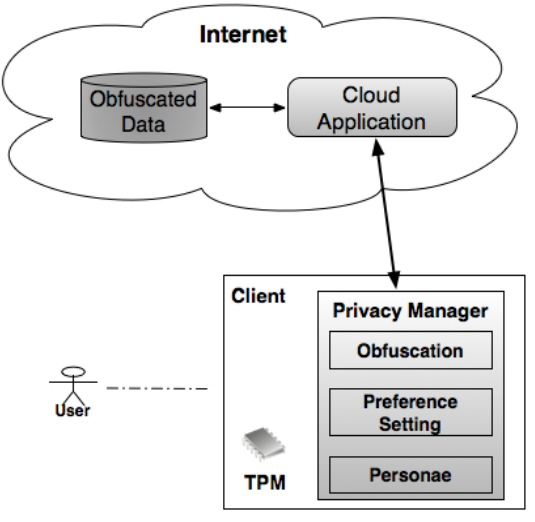

Fig. 1. Client-Based Privacy Manager

As an enhancement to our solution, a TPM on the client machine can be used to protect the obfuscation keys and provide further benefits. The privacy manager software and the methods linking sensitive data to pseudonyms can be protected by a TPM (see Figure 1). The TPM can provide encryption services and also allow integrity checking of the Privacy Manager software. In general, the main benefits that trusted computing could provide for client-based privacy management are hardware-based cryptographic functionality, confidentiality and integrity. In terms of confidentiality, it decreases the risk of unsecured access to secret material, by means of tamper-resistant hardware-based protection of keys. Moreover, protected data on the platform is not usable by other platforms. Trusted computing could yield greater trust in integrity of the privacy management software, integrity of the involved platforms, and platform identities.

3.2 Privacy Manager in a Hybrid Cloud

As an alternative, as illustrated in Figure 2, the Privacy Manager may be deployed in a local network, or a private cloud, to protect information relating to multiple parties. This would be suitable in environments, such as enterprise environments, where local protection of information is controlled in an adequate manner and its principal use would be to control personal information passing to a public cloud. The Privacy Manager can itself be virtualized within the internal cloud. Note that the TPM could also be virtualized, within the private cloud.

Advantages to this approach include that the benefits of the cloud can be reaped within the private cloud, including the most efficient provision of the Privacy Manager functionality. It can provide enterprise control over dissemination of

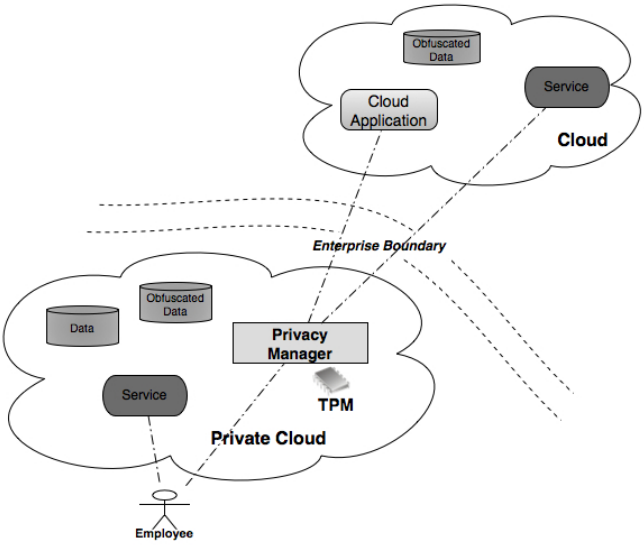

Fig. 2. Enterprise-focused Privacy Manager

sensitive information, and local compliance. A significant issue however is scalability, in the sense that the Privacy Manager might slow down traffic, provide a bottleneck and may not be able to adequately manage information exposed between composed services.

There are various different options with respect to this type of architecture. For example, the proxy capability could be combined, even in a distributed way, with other functionalities, including identity management. Another example is that trusted virtual machines [16] could be used within the privacy cloud to support strong enforcement of integrity and security policy controls over a virtual entity (a guest operating system or virtual appliance running on a virtualized platform). It would be possible to define within the Privacy Manager different personae corresponding to different groups of cloud services, using different virtualized environments on each end user device. In this way, virtualization is used to push control from the cloud back to the client platform. As with the previous architecture, there could be mutual attestation of the platforms, including integrity checking.

3.3 Privacy Infomediary within the Cloud

Figure 3 shows how the Privacy Manager may be deployed as (part of) a privacy infomediary [17], mediating data transfer between different trust domains. The Privacy Manager would act on behalf of the user and decide the degree of data transfer allowed, based upon transferred user policies and the service context, and preferably also an assessment of the trustworthiness of the service provision environment. Notification and feedback by the Privacy Manager to the user would also be preferable here, in order to increase transparency and accountability.

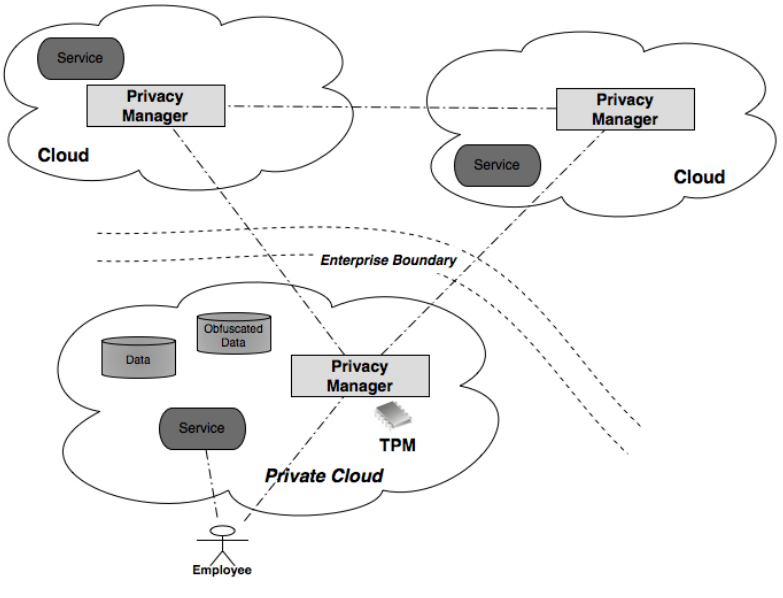

Fig. 3. Privacy Manager within the Cloud

The infomediary could be a consumer organization or other entity that is trusted by the users. It might alternatively be an entity that already exists within the cloud in order to provide an alternative function, such as an identity provider or auditor, and the functionality could be an extension of that. For example, the open source project Otemba [18] implements key management and user management, separating cryptographic keys from the cloud infrastructure. A key management role might be extended to a general infomediary role.

The infomediary may also play a role in checking that the user preferences are satisfied before providing a decryption key for decrypting any data that needs to be decrypted in order for the cloud service to be provided (for example, it could be a Trust Authority in order to provide IBE decryption keys [10,11]). Again, trusted infrastructure [7] could be useful in ensuring that the infrastructural building blocks of the cloud are secure, trustworthy and compliant with security best practice.

The following section provides more detail about the obfuscation mechanism used by the Privacy Manager, in these different cases.

4 Obfuscation

The aim of obfuscation is to solve the following general problem. A user has private data $\underline{x}$. He wishes to carry out some communication protocol with a service provider, which will enable him to learn the result of some function f on $\underline{x}$, without revealing $\underline{x}$ to the service provider. (The function f may itself depend on some data known to the service provider but not to the user, and on some data supplied by the user which is not private).

If the user and service provider are both willing to use whatever protocol will solve the problem, and have sufficient computing power and storage to do so, Yao's protocol for secure two-party computation [19] solves this problem for any f which can be expressed as a circuit. In fact, Yao's protocol can be used to ensure that the service provider learns no information at all about $\underline{x}$. So in this case, any polynomial-time application could be calculated in a completely obfuscated fashion. In fact, Yao's protocol can be used to ensure that the service provider learns no information at all about $\underline{x}$. Yao's protocol requires several rounds of interactions between the user and service provider, which depend on the choice of f. Gentry [20] has recently removed this requirement for interaction by constructing a remarkable encryption scheme which allows the service provider to calculate the encrypted value of $f(\underline{x})$ given the encrypted value of $\underline{x}$, for *any* f which can be expressed as a circuit, while also ensuring that the service provider learns no information about x. Gentry's encryption scheme improves on prior work on homomorphic encryption, eg. [21].

However, there are two problems with applying these solutions in cloud computing. The first is efficiency. Gentry's full scheme is impractical due to its rather high computational complexity. Although there has been a body of work improving the efficiency of Yao's protocol and related secure computation techniques such as privacy-preserving data mining [21,22], when the input data $\underline{x}$ is large these methods can still require a large amount of storage or computation on the part of the user. One of the attractions of cloud computing is that it can enable users to process or store large amounts of data at times of peak demand without having large amounts of computing

resources in-house. The other problem is that cloud computing providers may not be willing to rewrite their applications. If this is the case, the user has to calculate $f(\underline{x})$ using only the functions provided by the service, which in this section are denoted $f_1,\ldots,f_n$. The set of functions f for which it is possible to do this without revealing $\underline{x}$ to the service provider depends on $f_1,\ldots, f_n$ and on the extent of the user's computing resources. For some specialized cloud computing services, only one function is provided, which will typically be a MapReduce-style function [23] if the input data is a large data set; some more generic services offer full SQL SELECT functionality [24].

The requirement that we make of obfuscation in this paper is only that it is difficult for the service provider to determine $\underline{x}$ given the obfuscated data. It may be that the service provider can easily obtain some information about $\underline{x}$, but not enough to determine $\underline{x}$. As a different example of obfuscation methods that allow some but not all information about the input data to be learned from the obfuscated data, Narayanan and Shmatikov [25] describe an obfuscation method which allows individual records to be retrieved from an obfuscated database by anyone who can specify them precisely, while making "mass harvesting" queries matching a large number of records computationally infeasible. As remarked in [25], there is a tension between the strict security definitions and loose notions of efficiency used by the cryptography community, and the strict efficiency requirements but loose security requirements of the database community. Like the database community we prioritize efficiency over strength of the security definition, as it is essential for us that the privacy manager be practical and scalable to implement.

4.1 The Algebra of Obfuscation

The general algebraic description of obfuscation is as follows. Suppose you wish to use an application to calculate the result of a function f on an input $\underline{x}$, without revealing $\underline{x}$ to the application. The application can calculate functions $f_1,\ldots, f_n$. (Typically, but not necessarily, one of these will be equal to the function f). You can use the application to compute function f in an obfuscated fashion if for some positive integer m there are encryption functions $o_1,\ldots, o_m$ such that it is difficult to determine $\underline{x}$ from the tuple $o_1(\mathrm{k},\underline{x}),\ldots., o_m(k,\underline{x})$ without knowing the key k, and a decryption function d such that for all inputs $\underline{x}$ and keys k,

$$d(k,\ f_1(o_1(k,\underline{x})),\ldots, f_m(o_m(k,\underline{x}))) = f(\underline{x}) \tag{1}$$

To perform the obfuscated calculation, first encrypt $\underline{x}$ with each of the encryption functions to form the tuple $(\mathrm{o}_1(k,\underline{x}),\ldots, o_m(k,\underline{x}))$, using a key k known to you but not to the application. Send the values in the tuple to the application, to compute the values $f_1(o_1(k,\underline{x})),\ldots, f_m(o_m(k,\underline{x}))$. Finally, apply the decryption function to obtain $f(\underline{x})$ from the key k and the values output from the application. Since the only information the application receives is the tuple $(o_1(k,\underline{x}),\ldots, o_m(k,\underline{x}))$, it is difficult for the application to determine $\underline{x}$.

We now give some examples of calculating functions an obfuscated fashion. Since we are interested in using cloud applications, in the examples the functions $f_1,\ldots, f_n$ are typically SQL SELECT commands or MapReduce-style functions of a list (possibly a very long list) of values, indexed by a finite index set I.

Example 1: Using Privacy Homomorphisms
Privacy homomorphisms were first introduced by Rivest, Adelman and Dertouzos [26], who give several examples. A privacy homomorphism is a family of functions (e_k,d_k,f,g) where e_k is an encryption function depending on key k, such that for each key k and messages $a_1,\ldots,\ a_r$,

$$d_k(g(e_k(a_1),\ldots,\ e_k(a_r))) = f(a1,\ldots,\ ar) \tag{2}$$

If you wish to calculate function f and know a privacy homormorphism (e_k,d_k,f,g), you can use it to calculate f in an obfuscated fashion via an application that can calculate the function g. Set m=1 and f_1=g, and for each $\underline{x}$ = $(x_i{:}i$ in $I)$, i in I and key k, set $o_i(k,\underline{x}) = e_k(x_i)$. Set d to be the function sending (k,y) to $d_k(y)$ for all y. Then by equation (2), equation (1) holds.

Example 2: TC3 Health
TC3 Health [27] is a cloud-based company which checks health insurance claims on behalf of insurance companies. The functions that they calculate have the property that if a patient identifier (a name, say, or a hospital ID) is replaced everywhere in the input by a pseudonym, the resulting output is the same as that obtained by replacing the patient identifier by the pseudonym everywhere in the original output. They are therefore able to offer a service which checks the claims without requiring patient identifiers to be released to the cloud: the insurance companies replace the patient identifiers with (unique) pseudonyms, send the result as input to TC3 Health, and translate back the pseudonyms in the output. In terms of equation (1) above, m=1, the key k is the insurance company's map from patient identifiers to pseudonyms, o_1 is the application of this map and d is the application of the inverse map.

Example 3: Share Investment Web Site
In this example the function calculated is the current value of a portfolio of shares $\underline{x}$ = $(x_i{:}\ i$ in $I)$ where I is a set of companies and x_i is the number of shares of company i in the portfolio. The application offers the same function, which calculates the sum over i in I of $x_i.v_i$, where v_i is the current value of a share in company i: the value v_i is known to the application, but the share owner does not know the value without help from the application. A straightforward way of performing this calculation in an obfuscated fashion is choose k to be a positive integer and set

$$m=1,\ \ o_1(k,\underline{x})=k.\underline{x},\ \ d(k,\underline{y})=k^{-1}.\underline{y}\ \text{ for all } k,\ \underline{x},\ \underline{y} \tag{3}$$

In fact, some people use just this obfuscation method when they calculate the value of their portfolios online. However, $o_1(\underline{x})$ reveals the set of companies whose shares are contained in the portfolio (and also their relative frequency). This information might be enough to identify an individual investor. Moreover, the key k must be a divisor of the highest common factor of the entries of $o_1(\underline{x})$, and so may be easy to guess given $o_1(\underline{x})$ and some likely bounds on the total value of the portfolio. An obfuscation method that does not give away as much information is to construct several different portfolios, such that $\underline{x}$ is a linear combination of these portfolios, and derive the value of $\underline{x}$ from the values of these portfolios.

For example, choose a key k consisting of a map k_0: I $\rightarrow$ {1,2}, two portfolios $\underline{k}_1$, $\underline{k}_2$, and two integers k_3, k_4 greater than 1. For any portfolio $\underline{x}$ write $\underline{x}^{(k,1)}$, $\underline{x}^{(k,2)}$ for the

portfolios (x_i: i in I, $k_0(x_i)$=1) and (x_i: i in I, $k_0(x_i)$=2) respectively. Set m=3 and define the obfuscation functions by

$$o_1(k,\underline{x}) = \underline{x}^{(k,1)} + \underline{k}_1,\ o_2(k,\underline{x}) = \underline{x}^{(k,2)} + \underline{k}_2,$$
$$o_3(k,\underline{x}) = k_3.\underline{k}_1 + k_4.\underline{k}_2 + (k_3\text{-}\min\{k_3,k_4\}).\underline{x}^{(k,1)} + (k_4\text{-}\min\{k_3,k_4\}).\underline{x}^{(k,2)} \quad (4)$$

If you know the triple ($o_1(k,\underline{x})$, $o_2(k,\underline{x})$, $o_3(k,\underline{x})$) but not k, it is difficult to guess $\underline{x}$, and also difficult to guess the set of companies i such that $x_i > 0$ or the relative values of these x_i. Define the deobfuscation function d by

$$d(k,v_1,v_2,v_3) = (\min\{k_3, k_4\})^{-1}.\ (k_3.v_1 + k_4.v_2 - v_3) \text{ for all } v_1,v_2,v_3 \quad (5)$$

It is straightforward to check that equation (1) holds for these obfuscation functions and deobfuscation function, when both f and f_1 are the function that returns the value of the portfolio. So this allows the value of portfolio $\underline{x}$ to be calculated in an obfuscated fashion.

Example 4: Simple Obfuscation of SQL Queries

Suppose that $\underline{x}$ = (x_i: i in I) describes the content of a SQL database: each x_i is a database row, containing entries $x_i(1)$, …, $x_i(c)$ in columns 1 to c. For j=1,2,..c let nj be the name of column j, and let Vj be the set of values that are permitted in column j. The application can perform SQL queries, and we would like to compute an SQL query of the form

```
SELECT <ex 1> WHERE <ex 2> GROUP BY <ex 3> ORDER BY
<ex 4> LIMIT n
```

in an obfuscated fashion, where

- <ex 1> is a nonempty list of terms of the form `MAX(ni)`, `SUM(ni)`, `COUNT(*)` or `ni` for some $1 \le i \le c$,
- <ex 2> is obtained by combining terms of the form `ni = nj`, `ni = v`, or `ni > v` (for some $1 \le i, j \le c$ and v in V_i) using the logical operators `AND`, `OR` and `NOT`,
- <ex 3> is a possibly nonempty list of column names,
- <ex 4> is an element of <ex 1> or its negative, or is empty
- n is either a positive integer or infinity; if it is infinity then the "`LIMIT n`" clause is omitted from the query.

If any of <ex 2>, <ex 3>, <ex 4> are empty then the relevant subclause is omitted from the query. Examples of such queries include, for instance,

```
SELECT MAX(n3) WHERE ((n7 != v7 OR n3 <= v3) AND n1 = n2 )
SELECT n1, SUM(n5), COUNT(*) GROUP BY n1 ORDER BY -
SUM(n5) LIMIT 10
```

The key k used for the obfuscation of such a query consists of c+1 functions

$$\pi: \{1,2,\ldots,c\} \rightarrow \{1,2,\ldots,c\},\ k_i: V_i \rightarrow V_{\pi(i)} \quad (1 \le i \le c) \quad (6)$$

chosen such that k_i is sum-preserving if `SUM(ni)` is in <ex 1>, and is order-preserving if either `MAX(ni)` is in <ex 1>, `ni > v` appears in <ex 2>, or `ni` appears in <ex 4>. The obfuscation function o_1 is given by

$$o_1\text{: } (k, \underline{x}) \rightarrow (y_i\text{: } 1 \leq i \leq c) \text{ such that for all } i,\ y_{\pi(i)} = k_i(x_i) \tag{7}$$

To calculate the query in an obfuscated form, an obfuscated query is performed on $o_1(k, \underline{x})$, where the obfuscated query f_1 (which depends on the original query and on k) is obtained by substituting each column name ni in the original query by nj where $j = \pi(i)$, and substituting each value v in the original query occurring in a substring of form `ni = v` or `ni > v` by the value k_i(v). The answer (ans_1,…, ans_a) is then decrypted using function d, (which again depends on the original query and k), where

- $d(ans_1,\ldots, ans_a) = (d'(\text{ans}_1),\ldots, d'(\text{ans}_a))$,
- $d'(ans_j) = k_i^{-1}(ans_j)$ if the j^{th} element of <ex 1> in the original query is `ni`, `MAX(ni)` or `SUM(ni)`,
- $d'(ans_j)$ = is ans_j if the j^{th} element of <ex 1> in the original query is `COUNT(*)`.

It is straightforward to check that the result of this calculation is the same as the result of performing the original query on $\underline{x}$, so equation (1) holds.

Example 5: More Complicated SQL Query Obfuscation

In the previous example, the value of an entry in the obfuscated input database depended only on the value of one entry in the original database. It is possible to calculate selected SQL queries in an obfuscated fashion in such a way that the entries in some columns of the obfuscated database depend on multiple entries in the original database. As an example, consider the two queries

```
SELECT SUM(n2) WHERE n1 = v
SELECT n1 GROUP BY n1
```

and suppose that the values in column 1 of the original database are particularly sensitive, so for extra protection we want to avoid having any entry in the obfuscated database depend solely on an element in column 1 of the original database. Pick a key k consisting of c+3 one-to-one functions

$$\begin{gathered}\pi\text{: } \{1,..c\}\rightarrow\{1,\ldots c\},\ q\text{: } \mathrm{V}_{\pi(1)}\rightarrow \{0,1\},\ k_0\text{: } V_1 \rightarrow \mathrm{V}_{\pi(1)},\\ k_{\mathrm{i}}\text{: } V_i \rightarrow V_{\pi(\mathrm{i})},\ 1 \leq i \leq c\end{gathered} \tag{8}$$

where k_2 is chosen to be sum-preserving, and q can be expressed in the SQL query language. Set m=1 and define the obfuscation function o_1 by

$$\begin{gathered}o_1\text{: } (k, \underline{x}) \rightarrow (y_i\text{: } 1\leq i\leq c) \text{ such that } y_{\pi(i)} = k_i(x_i) \text{ for all } i>1,\\ y_{\pi(1)} = k_j(x_1) \text{ where } j = q(y_{\pi(2)})\end{gathered} \tag{9}$$

The same obfuscation function can be used for both queries. Write m1, m2 for the names of columns $\pi(1)$, $\pi(2)$, and w0, w1 for the values $k_0(v)$, $k_1(v)$. In the case of the first query, set f_1 to be the function calculating the query

```
SELECT SUM(m2) WHERE ((m1=w0 AND q(m2)=0) OR (m1=w1 AND
q(m2)=1))
```

and set d to be the function sending (k,y) to $k_2^{-1}(y)$. In the case of the second query, set f_1 to be the function calculating the query

```
SELECT m1, q(m2) GROUP BY m1, q(m2)
```

and set d to be the function which, given key k and a list of pairs $(y1,y2)$ with $y2$ in $\{0,1\}$, calculates $k_{y2}^{-1}(y1)$ for each pair in the list and returns a list of the unique results obtained. As for the other examples, it is straightforward to check that for both of the queries equation (1) holds (with m=1), so that the query is calculated in an obfuscated fashion.

In this section we have described various different obfuscation mechanisms that can be used by the Privacy Manager. Different functions, with varying degrees of obfuscation, can be specified within preferences available to the user via the Privacy Manager. This process can be made more intuitive via the use of personae, as described in Section 2. The following section provides an illustration of this approach which is fully implemented.

5 Online Photo Scenario

This section looks at a particular scenario, and describes how the privacy manager could operate in this scenario. In particular, we discuss the user interface for the privacy manager for this application.

5.1 Scenario: Cloud Photo Application

Vincent, a professional photographer and freelance writer for a geographic magazine, loves taking photographs, travelling, and writing articles. He is also a social man and likes to share his photos with family members and members of the photographic forums that he subscribes to.

Vincent recently bought a new professional digital camera with a built-in Global Positioning System (GPS) module, which provides a NMEA data stream from which the camera can extract the positional information (longitude and latitude) and record in the image metadata the location that the picture was taken. This feature helps him track and organize pictures geographically.

Vincent uses a commercial digital imaging web site to share his pictures online and order various products displaying his photos, such as postcards, T-shirts, and calendars. He likes the web site's simple and straightforward user interface. However, he soon realizes that the positional information contained in the pictures shot by his new camera may reveal the location of his house and his travel patterns, as such GPS information can be easily and accurately visualized in Google Earth.

With an increasing number of people using GPS-enabled cameras, the company owning the web site rolls out a new privacy manager assisting people to obfuscate certain metadata attributes which may reveal their private information – for example location information. By using this privacy manager, only the owner of the pictures can have access to the obfuscated attributes. The quality of the pictures is not affected. To demonstrate the scalability of our proposed obfuscation methods, we implemented two functions k_i for use in simple SQL query obfuscation (see example 4

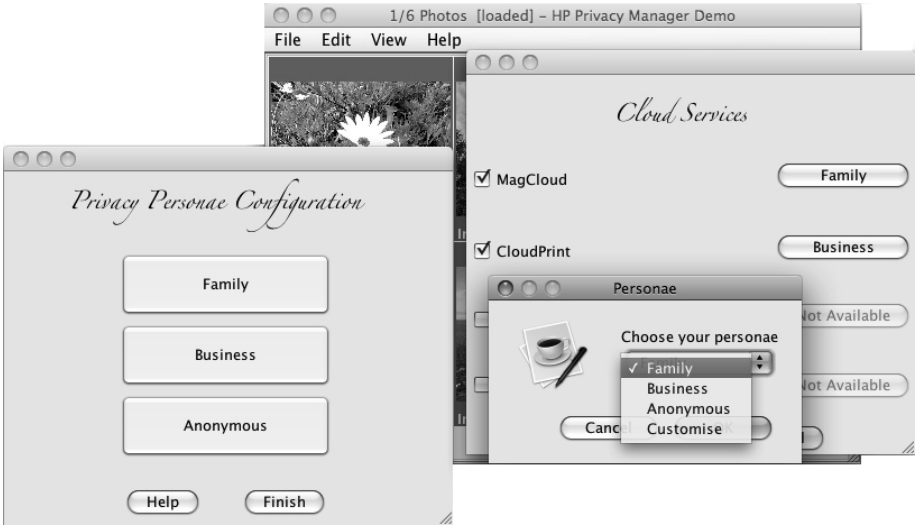

Fig. 4. Privacy Manager User Interface

of Section 4): *add*, which adds a secret number to an unencrypted numerical value, and *Caesar*, which does Caesar's Alphabet encryption using a secret alphabet order and shift value. Our implementation was not optimized for speed, but carried out 100,000 *add* calculations in 0.6s, and 100,000 *Caesar* calculations in 1.03s.

5.2 Privacy Manager User Interface

We have built a demonstrator of the use of the privacy manager within this scenario. The user interface for the privacy manager is shown in Figure 4. The end user selects the pictures that will be shared through certain cloud services. Specific personae, e.g. family, business, or anonymous, can be applied to obfuscate certain attributes associated with the pictures. The user can also customize the personae (i.e. choosing which attributes are to be obfuscated, and by which obfuscation methods) by changing the default settings via the privacy personae configuration window. By using the Privacy Manager, only the owner has control over the attributes, and the underlying obfuscation methods (as stated in Section 4) are transparent to the end users. Nevertheless, this method will not affect photo quality and still allows the photos to be further encrypted.

6 Previous Approaches to Privacy Management for Data Repositories

Since in this paper we are interested in managing the privacy of data which is sent to a database in the cloud, in this section we place this work in a wider context by reviewing previous general approaches to privacy management for data repositories, for

which various techniques have been developed to ensure that stored data is accessed in a privacy compliant way.

Some mechanisms and solutions have been built to encrypt confidential data when it is stored in data repositories, for example solutions using Translucent Databases [28]. Most of these solutions focus on confidentiality and access control aspects, and have little flexibility in providing policy-driven mechanisms encompassing aspects beyond authentication and authorization. [29,30] describe access control policy-based encryption mechanisms for XML documents. [29] describes mechanisms for fine-grained encryption of parts of XML documents, in which decryption keys can either be granted to data receivers or collected from LDAP servers, based on data receivers' credentials. [30] focuses on related cryptographic mechanisms.

Hippocratic Databases [31] include mechanisms for preserving the privacy of the data they manage. Their proposed architecture is based on the concept of associating privacy metadata (i.e. privacy policies) to data stored in data repositories, along with mechanisms to enforce privacy. The drawback of this approach is that it might require substantial changes to current data repository architectures, and therefore might take a long time and require substantial investment (by all the involved parties) to succeed. In addition, this approach does not take into account that the management of privacy spans across the database boundaries: such management has to be carried out within a broader context within cloud computing.

Although now withdrawn from production, IBM Tivoli Privacy Manager [32] provided mechanisms for defining fine-grained privacy policies and associating them with data. The privacy policies contain authorization constraints along with constraints on contextual information and intent. This approach addressed the privacy management problem purely from an access control perspective within a single enterprise. It did not include additional aspects relevant for privacy management within cloud computing such as trust management and dealing with ongoing privacy obligations dictated by legislation and enterprises' guidelines.

An alternative approach is based on an adaptive privacy management system where data are retrieved from standard data repositories, and parts of these data are encrypted and associated with privacy policies [33]. This aims to make use of current data repository technologies and reduce to the minimum the impact on them, in terms of required changes: interactions with data repositories can still happen but in a way that confidential data is protected and contextually released, in a fine-grained way, based on the fulfilment of associated privacy policies.

7 Analysis and Next Steps

The current state of this work is that we have a working implementation of the obfuscation feature of the privacy manager, both for the online photo application described in this paper and for another scenario that we have implemented [13] which is based on SalesForce.com's sales force automation suite [34]. The techniques described in examples 5 and 6 of Section 4 are implemented within our code, and it would be relatively simple to extend this to implement the other examples. Our next steps are to

extend this implementation to a greater range of cloud scenarios, including more complex ones.

Ideally, the privacy manager might be extended to allow consideration of trust assessment of third parties (risk management [35], reputation management [36], etc.), policy enforcement on service side (cf. sticky policies, involvement of Trust Authorities [11]), feedback and notification, subject data access requests, etc. We plan to consider these aspects in the EnCoRe project [37].

Our solution is not suitable for all cloud applications. Theoretically, as discussed in Section 4, *any* application which calculates a function of the input that can be expressed as a circuit could be calculated in a fully obfuscated fashion, if the service provider were willing to implement the application using Yao's protocol [19] or Gentry's encryption scheme [20]: however, the implementation of these for a large data set $\underline{x}$ may be impractical when resources are limited. For users with access to limited computing resources there is a tradeoff between the extent to which data is obfuscated and the set of applications that can effectively be used, even when the service provider gives full cooperation. Nevertheless, if the service provider cooperates then the other features of our solutions can still be used.

The picture is different if the service provider does not provide full cooperation. Some cloud service providers that base their business models on the sale of user data to advertisers (or other third parties) may not be willing to allow the user to use their applications in a way that preserves his privacy. Other providers may be willing to respect users' privacy wishes, but not to implement the service-side code that is necessary for some of the privacy manager's features. Yet other service providers may claim to cooperate, but not be trustworthy. In these cases, the features of our solution other than obfuscation will not be effective, since they require the honest cooperation of the service provider.

There is still a possibility that in these cases a user may be able to use obfuscation to protect the privacy of his data. However, the ability to use obfuscation without any cooperation from the service provider depends not only on the user having access to sufficient computing resources to carry out the obfuscation and de-obfuscation, but also on the application having been implemented in such a way that it will work with obfuscation. For example, a service that is customized with a map showing the area around a US user's zip code might theoretically be implemented in a way that would allow a user to obtain the correct customized result without revealing his zip code to the service provider. But a common method of implementing this type of service is to pass the input zip code directly to a map server, and mash up the map with the result from the rest of the service. With such an implementation it is difficult for the user to obtain the correct result without revealing the correct zip code to the application. As a more general example, for some applications it may be difficult to discover the set of input values that are treated as valid by the application. Without some knowledge of the set of valid inputs, it is not possible to design an obfuscation function such that the obfuscated input data is still valid input.

Despite this, we believe that many existing cloud services could be used in an obfuscated fashion without any cooperation from the service provider.

8 Conclusion and Acknowledgements

In conclusion, we have described a Privacy Manager and shown that this is a practical approach. We have also explored how the architecture would vary for different scenarios.

An earlier draft of this paper benefitted from helpful feedback from John Erickson and Guillaume Belrose.

References

1. Regulation of Investigatory Powers Act, Part II, s 28, UK (2000)
2. Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT ACT) Act, Title V, s 505 (2001)
3. Organization for Economic Co-operation and Development (OECD): Guidelines Governing the Protection of Privacy and Transborder Flow of Personal Data. OECD, Geneva (1980)
4. EU Data Protection Directive (95/46/EC) (1995)
5. Salmon, J.: Clouded in uncertainty – the legal pitfalls of cloud computing. Computing magazine, September 24 (2008), http://www.computing.co.uk/computing/features/2226701/clouded-uncertainty-4229153
6. Mowbray, M.: The Fog over the Grimpen Mire: Cloud Computing and the Law. Scripted Journal of Law, Technology and Society 6(1) (April 2009)
7. Pearson, S. (ed.): Trusted Computing Platforms. Prentice-Hall, Englewood Cliffs (2002)
8. World Wide Web Consortium (W3C): Platform for Privacy Preferences (P3P) Project, http://www.w3.org/P3P
9. PRIME, Privacy and Identity Management for Europe, https://www.prime-project.eu/
10. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Goos, G., Hartmanis, J., van Leeuwen, J. (eds.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
11. Casassa Mont, M., Pearson, S., Bramhall, P.: Towards Accountable Management of Identity and Privacy: Sticky Policies and Enforceable Tracing Services. In: IEEE Workshop on Data and Expert Systems Applications, pp. 377–382. IEEE Computer Society Press, Washington (2003)
12. Casassa Mont, M., Thyne, R.: A systemic approach to automate privacy policy enforcement in enterprises. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 118–134. Springer, Heidelberg (2006)
13. Mowbray, M., Pearson, S.: A client-based privacy manager for cloud computing. In: COMSWARE 2009. ACM, New York (2009)
14. Trusted Computing Group: Trusted Platform Module (TPM) Specifications (2009), https://www.trustedcomputinggroup.org/specs/TPM/
15. Pearson, S.: Trusted Computing: Strengths, Weaknesses and Further Opportunities for Enhancing Privacy. In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) iTrust 2005. LNCS, vol. 3477, pp. 305–320. Springer, Heidelberg (2005)
16. Dalton, C., Plaquin, D., Weidner, W., Kuhlmann, D., Balacheff, B., Brown, R.: Trusted virtual platforms: a key enabler for converged client devices. In: ACM SIGOPS Operating Systems Review, vol. 43(1), pp. 36–43. ACM, New York (2009)

17. Gritzalis, D., Moulinos, K., Kostis, K.: A privacy-enhancing e-business model based on infomediaries. In: Gorodetski, V.I., Skormin, V.A., Popyack, L.J. (eds.) MMM-ACNS 2001. LNCS, vol. 2052, pp. 72–83. Springer, Heidelberg (2001)
18. Otemba project: The Reasons for Otemba's Existence, http://sourceforge.net/apps/trac/otemba/wiki/Reasons%20for%20existence
19. Yao, A.C.: How to Generate and Exchange Secrets. In: 27th Symposium of Foundations of Computer Science (FoCS), pp. 162–167. IEEE Press, New York (1986)
20. Gentry, C.: Fully Homomorphic Encryption Using Ideal Lattices. In: 41st ACM Symposium on Theory of Computing, Bethesda, Maryland, USA, May 31-June 2 (2009), pp. 169–178 (2009)
21. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
22. Lindell, Y., Pinkas, B.: Privacy Preserving Data Mining. J. Cryptology 15(3), 151–222 (2002)
23. Liu, K.: Privacy Preserving Data Mining Bibliography, http://www.cs.umbc.edu/~kunliu1/research/privacy_review.html
24. Dean, J., Ghemawat, S.: Map Reduce: Simplified data processing on large clusters. Communications of the ACM 51(1) (2008)
25. Date, C.J.: A guide to the SQL standard. Addison-Wesley Longman Publishing Co., Boston (1986)
26. Narayanan, A., Shmatikov, V.: Obfuscated Databases and Group Privacy. In: Proceedings of the 12th ACM conference on Computer and Communications Security, pp. 102–111
27. Rivest, R., Adelman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In: DeMillo, R.A., et al. (eds.) Foundations of Secure Computation, pp. 168–179. Academic Press, New York (1978)
28. Amazon Web Services LLC: Case Studies: TC3 Health, http://aws.amazon.com/solutions/case-studies/tc3-health/
29. Wayner, P.: Translucent Databases, Flyzone Press (2002)
30. Bertino, E., Ferrari, E.: Secure and Selective Dissemination of XML Documents. In: Proc. TISSEC, pp. 290–331. ACM, New York (2002)
31. Miklau, G., Suciu, D.: Controlling Access to Published Data Using Cryptography. In: VLDB, VLDB Endowment (2003)
32. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: Proc. VLDB, VLDB Endowment, pp. 143-154 (2002)
33. IBM: IBM Tivoli Privacy Manager for e-Business (2009), http://www-01.ibm.com/software/tivoli/products/privacy-mgr-e-bus/
34. Casassa Mont, M., Pearson, S.: An Adaptive Privacy Management System for Data Repositories. In: Katsikas, S.K., López, J., Pernul, G. (eds.) TrustBus 2005. LMXS, vol. 3592, pp. 236–245. Springer, Heidelberg (2005)
35. Salesforce.com, Inc.: Sales Force Automation, http://www.salesforce.com/products/sales-force-automation/
36. Haimes, Y.Y.: Risk Modeling, Assessment, and Management. Systems, Man, and Cybernetics, Part C: Applications and Reviews 29(2), 315 (1999)
37. Despotovic, Z., Aberer, K.: P2P reputation management: Probabilistic estimation vs. social networks. Management in Peer-to-Peer Systems, Computer Networks 50(4), 485–500 (2006)
38. EnCoRe: EnCoRe: Ensuring Consent and Revocation, http://www.encore-project.info