Computing
DOI 10.1007/s00607-012-0206-z

Oblivious access control policies for cloud based data sharing systems

Zeeshan Pervez · Asad Masood Khattak · Sungyoung Lee · Young-Koo Lee · Eui-Nam Huh

Received: 27 October 2011 / Accepted: 26 July 2012

Abstract Conventional procedures to ensure authorized data access by using access control policies are not suitable for cloud storage systems as these procedures can reveal valid access parameters to a cloud service provider. In this paper, we have proposed oblivious access control policy evaluation (O-ACE); a data sharing system, which obliviously evaluates access control policy on a cloud server and provisions access to the outsourced data. O-ACE reveals no useful information about the access control policy neither to the cloud service provider nor to the unauthorized users. Through the security analysis of O-ACE it has been observed that computational complexity to compromise privacy of the outsourced data is same as reverting

Z. Pervez · A. M. Khattak · S. Lee (✉)
Ubiquitous Computing Lab, Department of Computer Engineering,
Kyung Hee University, Global Campus, 1 Seocheon-dong, Giheung-gu, Yongin-si,
Gyeonggi-do 446-701, South Korea
e-mail: sylee@oslab.khu.ac.kr

Z. Pervez
e-mail: zeeshan@oslab.khu.ac.kr

A. M. Khattak
e-mail: asad.masood@oslab.khu.ac.kr

Y.-K. Lee
Data and Knowledge Engineering Lab, Department of Computer Engineering,
Kyung Hee University, Global Campus, 1 Seocheon-dong, Giheung-gu,
Yongin-si, Gyeonggi-do 446-701, South Korea
e-mail: yklee@khu.ac.kr

E.-N. Huh
Internet Computing and Network Security Lab, Department of Computer Engineering,
Kyung Hee University, Global Campus, 1 Seocheon-dong, Giheung-gu,
Yongin-si, Gyeonggi-do 446-701, South Korea
e-mail: johnhuh@khu.ac.kr

Published online: 21 August 2012

asymmetric encryption without valid key pair. We have realized O-ACE for Google Cloud. Our evaluation results show the fact that O-ACE CPU utilization cost is 0.01–0.30 dollar per 1,000 requests.

Keywords Cloud storage · Data privacy · Access control policy · Data sharing

Mathematics Subject Classification (2000) 68M01 · 68M14 · 68P25

1 Introduction

Data synchronization and backup services (e.g., Dropbox [2], Amazon Cloud Drive [1], SkyDrive [7]) are becoming integral part of our daily computing life. Through these services we now can synchronize and share our data, on varied devices and with multiple users. Their adoption has certainly changed the way we manage our data at individual as well as at the enterprise level. Another notion, which is rapidly becoming a norm is online collaboration services (e.g., Google Docs [4], Microsoft Office Live [6], Zoho [8]). These services offer productivity suite (generally includes: Word processor, Spreadsheet, Presentation program), allowing multiple users to collaborate on a single document, devoid of concerns like conflict resolution of multiple updates, and real-time collaborative changes.

All of these data management services are powered by cloud computing; an epitome of on-demand and scalable computing paradigm [9]. Vendors of these prevalent services either tap into public or private cloud to cater for the ever increasing demand of storage and computational requirements. Adoption of cloud computing enables these vendors to reduce management cost of underlying infrastructure. Consequently, these services offer the abstraction of unlimited storage and CPU-time on subscription basis [11,12]. With cloud computing provisioning the core infrastructure to these services, growth and availability of digital contents outsourced to them is not an issue for the vendors and consequently for their subscribers as well.

As lucrative as it sounds, adoption of these services brings forth data privacy issues as we now count on cloud service providers to host possibly sensitive data [27,34]. Common approach to ensure data confidentially is to first encrypt the data and then disseminate decryption key to the legitimate users [18,19,24,25,39–41]. However, only encryption is not enough to ensure data confidentiality, as if decryption key is compromised there would be no second line of defense to avoid potential loss of data privacy [37]. As more and more applications and services are provisioned through cloud, risk of privacy breach is becoming higher [28,33]. Conventional security methodologies either fail to provide adequate measures or lack realism in the domain of cloud computing [10].

Access control polices in conjunction with cryptographic capabilities ensure legitimate and authorized access to the resources (i.e., outsourced data). In general access control policy regulates access to the outsourced data according to the rules specified by the data owner, and access parameters assign to a user delineates its access capabilities [36]. However, usual practice of ensuring authorized access loses its efficacy in a cloud storage system, as it would reveal access control policy and valid access parameters to the cloud service provider during policy evaluation [17,26]. This is because,

generally access control policy is a combination of logical expressions. These logical expressions are evaluated on the basis of access parameters, resulting in a final verdict that either grants or restrains access to the outsourced data [15,30].

The key challenge in enforcing access control policy in an untrusted domain is obliviousness. By this we mean that privacy of access control policy and access parameters should be assured. Oblivious access control policies have been employed by many system [14,17,22,29]. These oblivious policies are exploited where both entities (i.e., client and server) are concerned about the privacy of information involved in access control policy evaluation[1]. Typically server does not want to reveal access control policy as it could be a commercial secret, and would jeopardize the business strategy. Whereas, for client access parameters could reveal private information and would affect its privacy. Both of the entities own their private information (server: access control policy; client: access parameters), which they do not want to give away. However, in the case of a cloud storage system, cloud service provider does not own any of the private information. Its core responsibility is to provision outsourced data which is owned by the data owner. Whereas, data owner wants to ensure that only legitimate and authorized user can access the outsourced data.

In order to achieve data privacy in cloud storage systems it is important to design a system that can assure end to end privacy, involving outsourced data, access control policy, and access parameters. To address the privacy issues of a cloud storage system from end to end privacy aspect, we propose an oblivious access control policy evaluation (O-ACE), which assists data owner to outsource confidential data along with access control policy to a cloud server. Outsourced data and access control policy is concealed in such away that policy evaluation automatically grants or prohibits access to the desired contents. Policy evaluation is executed on the cloud server for the access parameters provided by the user. As access control policy is evaluated obliviously, cloud service provider cannot learn any information about the valid access parameters; consequently is unable to derive the valid decryption key which can decrypt the outsourced data.

To ensure privacy of the access control policy and access parameters we extended the notion of private matching [16]—*a private set intersection protocol for two parties*, to an untrusted domain, we called it delegated private matching (DPM). It assists data owner to delegate the task of private matching (policy evaluation) to a cloud service provider. Policy evaluation is executed by the cloud service provider in such a way that it processes the encrypted policy, whilst assists the legitimate and authorized user to learn set of values which are required to derive the valid decryption key.

Viability of O-ACE is demonstrated by realizing a cloud based data sharing system which enables the data owner to achieve fine-grained access control over the outsourced data. We utilize Google Cloud to provision outsourced data. For O-ACE's execution we employed Google App Engine [3]. Security analysis of O-ACE showed that computational complexity to compromise privacy of the outsourced data is equivalent to asymmetric encryption algorithm. Even if attackers team up with each other or with a cloud service provider computational complexity remains the same.

[1] We shall refer access control policy evaluation as policy evaluation.

Cloud service providers bill their subscribers according to the CPU consumption rate. Performance analysis of O-ACE on Google App Engine highlighted the fact that it only costs 0.01–0.30 cents per 1000 requests on a single compute instance having processing capability of 1.20 Ghz. Besides this O-ACE exerts reasonable computational load on data owner and authorized users. Devices having limited computational capabilities (i.e., smartphones and tablets) can be used to outsource and access the data on cloud storage.

O-ACE can be viewed a security service, that leverages the data owner to incorporate oblivious policy evaluation capabilities with the existing cloud based data sharing, and collaborative services. In general with O-ACE we are able to make the following contributions to the area of cloud storage.

- O-ACE is a new access control mechanism for untrusted domain (cloud server), which provides end to end privacy.
- O-ACE ensure end to end privacy by using standard cryptographic primitives. For data concealment it is not bound to any specific cryptographic algorithm. Data owner can opt for any algorithm suitable to its security requirements and computational capabilities of the target users.
- To demonstrate the functional importance of O-ACE, we have implemented it for Google Cloud. In our implementation scenario Google App Engine provides the functionality of oblivious policy evaluator, whereas, confidential documents are hosted by Google Docs in encrypted form.

The rest of the paper is organized as follows: Sect. 2 review the related work. Section 3 present the prerequisites for oblivious policy evaluation. Section 4 elaborate the delegated private matching. Section 5 outline the models, design goals and assumptions. Section 6 present a cloud based data sharing based on delegated private matching. Section 7 discuss the implementation detail of our proposed cloud based data sharing system. Section 8 evaluate our proposed delegated private matching for a data sharing system. Section 9 present the security analysis. Section 10 discusses the efficacy of proposed delegated private matching and its evaluation results. Finally, Sect. 11 conclude the paper.

2 Related work

In this section we discuss state of the art in the area of cloud based storage systems. Particularly systems which are more focused on data sharing and governance in cloud storage are considered.

Weichao Wang et al. [40] have proposed a cloud based data sharing system for massively large data. In order to achieve fine-grained access control, data is divided into multiple data blocks ($D_1, D_2, \ldots, D_n$), each encrypted with a distinctive block encryption key. These keys are managed by the data owner in a binary tree with the possibility to derive valid block encryption key from the parent node (non-leaf node). Data owner himself issues the security token along with a secret value to every legitimate user. Security token is utilized by the storage provider to ensure that request is initiated by a legitimate user. Whereas, secret value is utilized by the user to derive the block decryption key. Access privileges of a user are evaluated by the data owner

himself, after which appropriate secret value is revealed. This leads to the assumption that data owner remains always online to evaluate access control policies, thus greatly effecting the utility of cloud based data sharing. Besides this, proposed system does not consider scalability; with the increase is number of users data owner will become a bottle neck to access the cloud storage.

FADE [39] is a secure overlay cloud storage system based on policy-based assured file deletion. FADE is designed to share outsourced data in an untrusted domain and to assuredly delete it once the need of sharing is over. To ensure data confidentially and authorized data access, data encryption key is used to conceal the outsourced data, and control keys are used to encrypt the data encryption key. Concealed outsourced data and data encryption key are outsourced to a cloud storage; whereas, control keys are managed by a key manager. Policies are maintained and evaluated by the key manager. Whenever a policy is revoked appropriate control key is deleted thus restraining access to the outsourced data. FADE delegates the task of policy evaluation to the key manager, if key manager is compromised outsourced data can be decrypted without much effort, as there is no second line of defense deployed at the cloud storage.

TrustStore [41] is an Amazon S3 based storage service, enabling storage subscribers to outsource their confidential data to storage service provider (SSP), with data confidentiality and integrity considerations. It utilizes a Key management service provider (KMSP) to generate and distribute decryption keys, besides this KMSP also takes over the responsibility of user authentication. TrustStore segregates data into two components; data-fragments: dividing data into equal sized fragments encrypted with distinctive encryption keys, and meta-data-object: containing information about encryption keys and data fragments. Data-fragments are outsourced to the storage provider and meta-data-object is stored at KMSP. TrustStore is based on an assumption that SSP and KMSP do have any knowledge about each other, thus privacy of data is ensured. It does not enforce access control policy at cloud sever and thus fails to achieve fine-grained access control over the outsourced data.

Cryptographic cloud storage [25] is another cloud based data sharing system designed to outsource enterprise data storage. It consists of three core components i.e., data processor (DP), data verifier (DV), and credential generator (CG). DP encrypts the outsourced data, DV verifiers the data integrity at cloud storage, and CG generates decryption key for the users with whom data owner wants to share the outsourced data. Decryption keys are generated according to access control policy and user access privileges. Cryptographic cloud storage achieves fine-grained access control by encrypting the decryption keys with attributes based encryption (ABE) [21]. Although ABE achieves fine-grained access control; however, the entire system fails to utilize cloud storage efficiently. Instead of exploiting the cloud to enforce access control policies, data owner himself generates and disseminates ABE secret key to the authorized users. This key exchange protocol would eventually leads to under utilization of cloud storage as for every legitimate user data owner needs to generate the secret key according to user's access privileges.

Remote untrusted storage system closely resembles to cloud storage system specifically in terms of providing storage services. There are numerous systems which tends to ensure data privacy in remote storage systems. SiRiUS [19] is a secure file system designed to layer over an insecure network. It encrypts File Encryption Key (FEK)

with respective user's public key, which is then stored on remote storage along with the encrypted outsourced data. Similarly, Plutus [24] is a cryptographic file storage system that enables secure file sharing on untrusted servers. It organizes files of similar access patterns into file-groups. Outsourced files are encrypted with file-block keys and these keys are further encrypted with file-lockbox keys. Data owner handovers the file-lockbox keys to the legitimate users. Likewise, CRUST [18] is a cryptographic remote storage system, which assumes the existence of a trusted agent (*i.e., trusted third party*), responsible for key management. Trusted agent also takes on the responsibility to handover the decryption keys to the legitimate and authorized users.

All of these systems discussed above rely on key management protocols and services to restrain illicit data access. They do not consider access control policies to achieve fine-grained access control. By enforcing access control policies within the untrusted domain of storage service provider, key management protocols and specialized trusted agents/services can be avoided and consequently utilization of storage services can be maximized. More importantly access control polices assist the data owner to achieve fine-grained access control over the outsourced data.

3 Preliminaries

Before elaborating the O-ACE for cloud based data sharing, we introduce some of the preliminaries used in its development.

3.1 Homomorphic encryption

A cryptographic scheme is said to be homomorphic if its encryption function $\mathcal{E}_H$ holds the property i.e., $\mathcal{E}_H(x) * \mathcal{E}_H(y) = \mathcal{E}_H(x + y)$. A homomorphic encryption is said to be *semantically secure* if $\mathcal{E}_H$ reveals no information about x and y, hence it is computationally infeasible to distinguish between the case $x \neq y$ and $x = y$ [32].

Public key encryption scheme proposed by Pascal Paillier [31] is additively homomorphic, and consists of subsequent fundamental algorithms.

3.1.1 Key generation

Let p and q be two large primes and $n = p.q$. $\phi(n)$ denotes the Euler's totient function. Carmicheal's function is represented by $\lambda(n)$. For n, the product of two primes, $\phi(n) = (p-1)(q-1)$ and $\lambda(n) = lcm(p-1, q-1)$. These two functions exhibits the following properties over the multiplicative group $\mathbb{Z}^*_{n^2}$, i.e.,

$$|\mathbb{Z}^*_{n^2}| = \phi(n^2) = n.\phi(n) \tag{1}$$

and for any $\omega \in \mathbb{Z}^*_{n^2}$

$$\omega^{\phi(n)} = 1 \ (mod\ n) \tag{2}$$
$$\omega^{n\phi(n)} = 1 \ (mod\ n^2) \tag{3}$$

Public key $\mathcal{PK}$ is defined as (n, g), where g is an element of $\mathbb{Z}^*_{n^2}$, and secret key $\mathcal{SK}$ as $\lambda(n)$.

3.1.2 Encryption

To encrypt a message $m \in \mathbb{Z}_n$, randomly choose $y \in_R \mathbb{Z}^*_{n^2}$, and define an encryption function $\mathcal{E}_H$, such that

$$\mathcal{E}_H : \mathbb{Z}_n \times \mathbb{Z}^*_n \mapsto \mathbb{Z}^*_{n^2} \tag{4}$$

$$\mathcal{E}_H(m, y) = g^m y^n (mod\ n^2) \tag{5}$$

3.1.3 Decryption

To decrypt the cipher text, L is defined as $(u-1)/n,\ \forall u \in \{u | u = 1(mod\ n)\}$. Cipher text c can be decrypted by using secret key $\mathcal{SK} = \lambda(n)$.

$$\mathcal{D}_H(c, \lambda(n)) = \frac{L(c^{\lambda(n)}\ (mod\ n^2))}{L(g^{\lambda(n)}\ (mod\ n^2))} \tag{6}$$

3.1.4 Homomorphic operation

Arithmetic addition between the cipher texts, $c_1 = \mathcal{E}_H(m_1, y_1)$ and $c_2 = \mathcal{E}_H(m_2, y_2)$, is obliviously computed as:

$$\begin{array}{r} \mathcal{E}_H(m_1, y_1) = g_1^m {y_1}^n (mod\ n^2) \\ \mathcal{E}_H(m_2, y_2) = g_2^m {y_2}^n (mod\ n^2) \\ \hline \mathcal{E}_H(m_1, y_1) * \mathcal{E}_H(m_2, y_2) = g^{m_1+m_2}(y_1 * y_2)^n (mod\ n^2) \\ = \mathcal{E}_H(m_1 + m_2) \end{array} \tag{7}$$

3.2 Private matching

Private matching (PM) [16] is a value matching protocol. It assists two interactive entities to compute set intersection over their private set of values, without revealing any element of their private set to each other. It uses homomorphic encryption to identify the commonalities amongst the private sets, whilst ensuring privacy of each set.

Suppose, there is a client $\mathcal{C}$ and a server $\mathcal{S}$. $\mathcal{C}$ has its own private set of values $\mathcal{X}$: $\{x_1, x_2 \ldots x_n\}$, so does $\mathcal{S}$, $\mathcal{Y} : \{y_1, y_2 \ldots y_n\}$. $\mathcal{C}$ wants to compute set intersection with $\mathcal{S}$ over the private set of values (i.e., $\mathcal{X}$, $\mathcal{Y}$). However, $\mathcal{C}$ does not want to seep out any information about $\mathcal{X}$, with an exception of set cardinality. To identify the commonalities between $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{C}$ computes a polynomial (see Eq. 8), whose roots are members of $\mathcal{X}$.

$$P(x \in \mathcal{X}) = (x - x_1)(x - x_2) \ldots (x - x_n) = \sum_{i=0}^{n} \alpha_i x^i \tag{8}$$

$\mathcal{C}$ then sends the homomorphically encrypted coefficients ($\hat{\alpha}_{0\ldots n}$) of $P(x)$ to $\mathcal{S}$. By using $\hat{\alpha}$, $\mathcal{S}$ evaluates $P(y)$ for every element of its private set. It then computes oblivious value by multiplying evaluated $P(y)$ with a random number r and adding it to y, i.e., $\mathcal{E}_H(r.P(y) + y)$, where $\mathcal{E}_H$ is a homomorphic encryption algorithm. These oblivious values are then send to $\mathcal{C}$ for decryption. At $\mathcal{C}$, the decryption of an oblivious value results in y, if $P(y)$ computed by $\mathcal{S}$ is evaluated at z, such that $\langle z \subseteq \bigcap \mid (z \in \mathcal{X}) \wedge (z \in \mathcal{Y}) \rangle$. Otherwise, $\mathcal{C}$ ends up generating a random value. At the end of this protocol, $\mathcal{C}$ learns only the intersection set; whereas, $\mathcal{S}$ ascertains nothing more than the cardinality of $\mathcal{X}$.

4 Oblivious private matching in an untrusted domain

Private matching is an interactive protocol between two entities. Both of the involved entities have to ensure their presence in order to compute set intersection over their private set of values [16]. In this section we extend the notion of private matching to delegated private matching (DPM), by relaxing the availability requirement on one of the involved entity. It assists passive entity to delegate the private matching task to a third party without compromising privacy of its private set.

Similar to PM, in DPM we have a client $\mathcal{C}$ and a server $\mathcal{S}$, with their own private set of values (see Sect. 3.2). However, $\mathcal{S}$ does not want to compute set interaction, it has delegated this task to a third party called validator ($\mathcal{V}$). Although the set intersection has been delegated to $\mathcal{V}$, still $\mathcal{S}$ does not want $\mathcal{V}$ to learn any information about the private set ($\mathcal{Y}$). In the subsequent illustration of DPM, we assume that $\mathcal{S}$ knows the public key of $\mathcal{C}$.

$\mathcal{S}$ selects a random mask ($\tilde{r}$) of an arbitrary length. It then encodes each element of its private set, using a publicly know encoding function i.e., $encode(y, \tilde{r}) \rightarrow \hat{y}$, where $y \in \mathcal{Y}$. Once $\hat{\mathcal{Y}}$ (encoded private set) is computed, $\tilde{r}$ is encrypted with $\mathcal{C}$'s public key. Finally, encrypted random number and encoded private set are send to $\mathcal{V}$. After that availability of $\mathcal{S}$ is not required, $\mathcal{V}$ can obliviously evaluate the matching process without compromising privacy.

To perform set interaction $\mathcal{C}$ obtains the encrypted random number from $\mathcal{V}$. It then decrypts it by using its private key. After that it encodes the private set of values by using the same encoding function as did by $\mathcal{S}$, i.e., $encode(x, \tilde{r}) \rightarrow \hat{x}$, where $x \in \mathcal{X}$. Once $\hat{\mathcal{X}}$ (encoded private set) is computed. $\mathcal{C}$ then computes a polynomial $P(\hat{x})$, whose roots are members of $\hat{\mathcal{X}}$ (see Eq. 8). It then initializes a homomorphic encryption key pair (secret and public key) and encrypts the coefficients $\alpha_{0\ldots n}$ of $P(\hat{x})$ by using homomorphic encryption secret key. After that encrypted coefficients ($\hat{\alpha}_{0\ldots n}$) along with homomorphic encryption public key are sent to $\mathcal{V}$.

On receiving $\hat{\alpha}_{0\ldots n}$, $\mathcal{V}$ evaluates $P(\hat{y})$ for every element of $\hat{\mathcal{Y}}$, by using $\hat{\alpha}$. It then computes oblivious value by multiplying $P(\hat{y})$ with a random number r and adding it to $\hat{y}$, i.e., $\mathcal{E}_H(r.P(\hat{y}) + \hat{y})$. These oblivious values are then send to $\mathcal{C}$.

Finally, to identify the commonalities between $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{C}$ decrypts the oblivious values. Decryption of an oblivious value reveals $\hat{y}$, if $P(\hat{y})$ computed by $\mathcal{V}$ is evaluated at z such that $\langle z \subseteq \bigcap \mid (z \in \hat{\mathcal{X}}) \wedge (z \in \hat{\mathcal{Y}}) \rangle$. Otherwise $\mathcal{C}$ ends up decrypting a random value.

During the entire execution of DPM, $\mathcal{V}$ learns nothing more than the cardinality of $\mathcal{S}$'s private set. This can be easily mitigated by adding dummy values. With the amalgam of a public key cryptography and homomorphic encryption $\mathcal{V}$ learns no useful information, yet being able to obliviously evaluate the private matching.

5 Models, design goals and assumptions

Before discussing O-ACE in detail, we briefly describe its conceptual and security models. We also specify its design goals, illustrating its functional importance for data sharing in an untrusted cloud storage with privacy considerations.

5.1 Conceptual model

To enforce access control policy in an untrusted cloud storage system: credential issuing authority, data provider, data consumer, and cloud storage service provider are considered as the involved entities. For brevity, we refer them as authority, owner, user, and cloud server respectively. Authority is a trusted entity which takes on the responsibility of issuing identity attributes to the users and their identity assertions to the owner. Owner utilizes the storage facility provided by the cloud server, by outsourcing the encrypted data it wants to share (*e.g., pictures, text documents, multimedia files*), along with the oblivious access control policy. Cloud server obliviously evaluates the access control policy and disseminates the secret values to each user - seeking access to the outsourced data. Secret values disseminated by the cloud server are utilized by the user to derive a valid decryption key to access the outsourced data.

5.2 Security model

Threats faced by the owner when outsourcing confidential data to a cloud server can be primarily divided into two categories, internal and external threats. Internal threats: cloud server himself is interested in the outsourced data for some business needs (*i.e., related ad serving, selling confidential data for some wicked motives*). External threats: fraudulent user seeks access to the outsourced data otherwise not allowed.

We believe that internal threats can be reasonably mitigated by the use of appropriate encryption algorithms. However, these algorithms fail to obstruct the external threats. To eliminate the external threats owner must restrain illicit data access by defining access control policy and enforcing it in the untrusted domain. Since, cloud server is not a trustable entity access control policy should have a property of obliviousness. In other words, cloud server should not be able to differentiate between an authorized and fraudulent user, yet being able to assist authorized user in deriving valid decryption key.

5.3 System design goals

The pivotal design goal of O-ACE is to develop a cloud storage system that enables the owner to share confidential data with authorized users, without compromising privacy of the outsourced data. As owner cannot remain always online to enforce

access control policy, cloud server must obliviously evaluate the user access privileges, and disseminate the appropriate secret values used to derive the valid decryption key. Since, cloud server is not a trustable entity, owner must be able to define access control policy which does reveal any information about the attributes required to decrypt the outsourced data. Besides this, these policies must not lose their efficacy in restricting unauthorized users to successfully decrypt the outsourced data.

5.4 Assumption and notations

For the intent of simplicity in the descriptive details of O-ACE, we assume the following assumptions.

- Cloud server is honest but curious. It performs the delegated task honestly but is also interested in the contents of the outsourced data—similar to [35].
- There exists a trustable authority, which issues identity attributes to the individual user.[2]
- Outsourced data is shared with the users to whom authority can issue identity attributes.
- There is no concern of privacy infringement in revealing identity attributes of a user to the owner.
  These attributes are merely account for uniquely delineating the user within an organization.[3]

Table 1 illustrates the notations that we use to explain the core concept of O-ACE.

We intentionally avoid differentiating between the cryptographic key(s) of the individual users. Throughout this text cryptographic key(s) for a particular user j can be referred by affixing j as a subscript, without changing the actual usage semantics i.e., δ_u can be binded to a user j as δ_{u_j}, same applies to the rest of keys.

6 Proposed system

In this section we introduce O-ACE. We first present the main idea, then we propose a cloud based data sharing system which utilizes O-ACE to achieve fine-grained access control over the outsourced data.

6.1 Main idea

Suppose Bob is an under cover agent working on some special assignment. He has collected substantial evidence against a drug lord in downtown area, which he wants to share only with the concerned authorities. Since, Bob cannot presents the evidence (data) in person to the authorities, he decides to use cloud storage facility provided by

[2] Authority can issue a root level certificate to an organization, which uses it to sign X.509 v3 certificate (*attribute certificate*) of its employees [23].

[3] In Sect. 9 we discuss how to prevent adversary from gaining knowledge about the user attributes and then exploiting them to access the outsourced data.

Table 1 Notations used in the descriptive detail of O-ACE

Notations	Description
$\mathcal{F}$	Data to be shared with legitimate users
$att_{0...n}$	Attributes issued to a user by the authority
$\tilde{r}$	Randomly selected mask i.e., $\tilde{r} \in_R \mathbb{Z}_q$
H_1	One way hash function which encodes att_i to an integer of arbitrary length i.e., $H_1(att_i)$: $\{0,1\}^* \rightarrow \mathbb{Z}$
H_2	One way hash function which encodes att_i to integer of module q, where q is a prime i.e., $H_2(att_i)$: $\{0,1\}^* \rightarrow \mathbb{Z}_q^*$
l_i	Legitimacy value: att_i encoded by H_1
$\hat{l_i}$	Mandatory value: att_i encoded by H_2
ψ	Pseudo random function which outputs the symmetric encryption key of an arbitrary length
ω	Legitimacy key: encrypts the legitimacy values
$\mathcal{E}_H, \mathcal{D}_H$	Homomorphic encryption and decryption algorithms
σ_{pk}, σ_{sk}	Public and secret key for homomorphic encryption
$\mathcal{E}_S, \mathcal{D}_S$	Symmetric encryption and decryption algorithms
κ	Master key (encryption or decryption key) for an arbitrary symmetric encryption
$\mathcal{E}_A, \mathcal{D}_A$	Asymmetric encryption and decryption algorithms
k_{pub}, k_{pri}	Public and private key pair for asymmetric encryption

the Eve. However, due to the sensitivity of the data, Bob does not trust Eve and wants to upload (outsource) the encrypted data. As Bob is working under cover for a quite long time, he does not know the lead detective who in charge of drugs related criminal cases in downtown area.

Bob contacts the Home Office (authority) which asserts the identity information of a lead detective responsible for crime related to drugs in downtown area. Since, authority has the information about each employee, it asserts the identity attributes of Alice, along with her public key. Bob encodes the asserted attributes into legitimacy values by using a publicly known encoding function and into mandatory values by employing a private encoding function. Mandatory values are used to derive the master key that encrypts the evidence (data) with arbitrary symmetric encryption algorithm. Whereas, legitimacy values initialize a pseudo random function which generates the encryption key of an arbitrary symmetric encryption algorithm, that encrypts the respective mandatory values. Legitimacy values are then concealed with symmetric encryption algorithm. Finally, encrypted evidence is outsourced to the cloud server along with concealed legitimacy and mandatory values.

Out of bounds, Alice receives information about the shared contents. She encodes her identity attributes into legitimacy values and engages in an oblivious policy evaluation protocol with Eve, as a result she learns the mandatory values. During the oblivious policy evaluation, Eve neither learns the legitimacy values nor the mandatory values. Whereas, Alice only learns the concealed mandatory values if she holds the required set of attributes used by Bob to encode the legitimacy and mandatory values. Once, Alice has the concealed mandatory values, she uses the legitimacy values to decipher them and consequently derives the master key. Ultimately, she decrypts

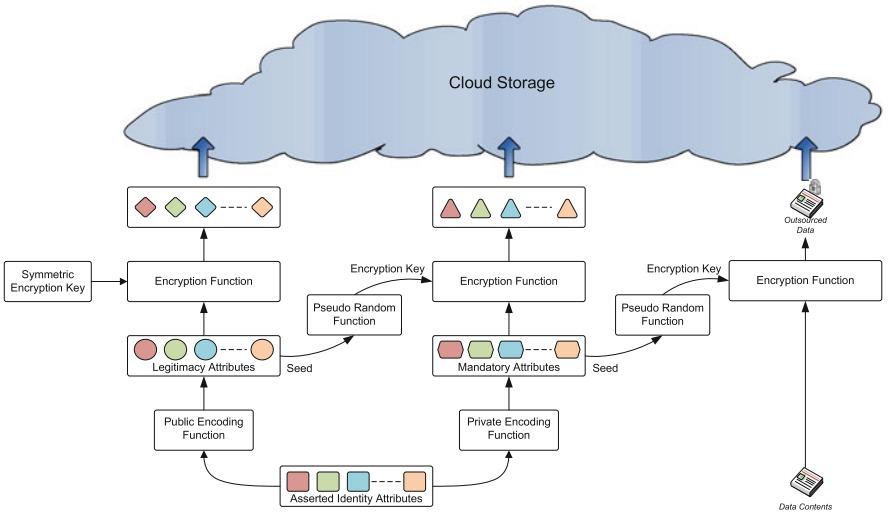

Fig. 1 Oblivious access control policy evaluation (O-ACE) for cloud based data sharing

the outsourced evidence by using the master key. Figure 1 illustrates the conceptual model of our proposed cloud based data sharing system with oblivious access control policy evaluation.

6.2 Enforcing oblivious access control policy in cloud storage

As cloud storage facility is not owned or managed by the owner, there is a need to ensure, not only privacy of the outsourced data but also the policy that govern its access. In order to enforce access control policy, O-ACE utilizes delegated private matching (DPM) along with identity attributes, in such a way that possession of certain identity attributes validates the legitimacy and authenticity of the user. However, as cloud server is not a trustable entity, owner cannot rely on it to validate user identity attributes. Instead, we have coupled the derivation of master key with the identity attributes, such that user possessing the valid set of attributes can learn the information which can generate the valid master key.

To ensure privacy of the outsourced data along with the access control policies, O-ACE processes the data in three fundamental steps, initialization, data outsourcing, and file access. These steps ensures that outsourced data can only be accessed (decrypted) by an authorized user, and during the whole process cloud server is unable to learn any useful information that can lead to privacy breach. Figure 2 illustrates the proposed system in terms of exchange of data and availability of the entities.

6.2.1 Initialization

In order to share confidential data, owner contacts the authority to obtain identity information of the target user. It specifies the identity selection criteria e.g., Department:

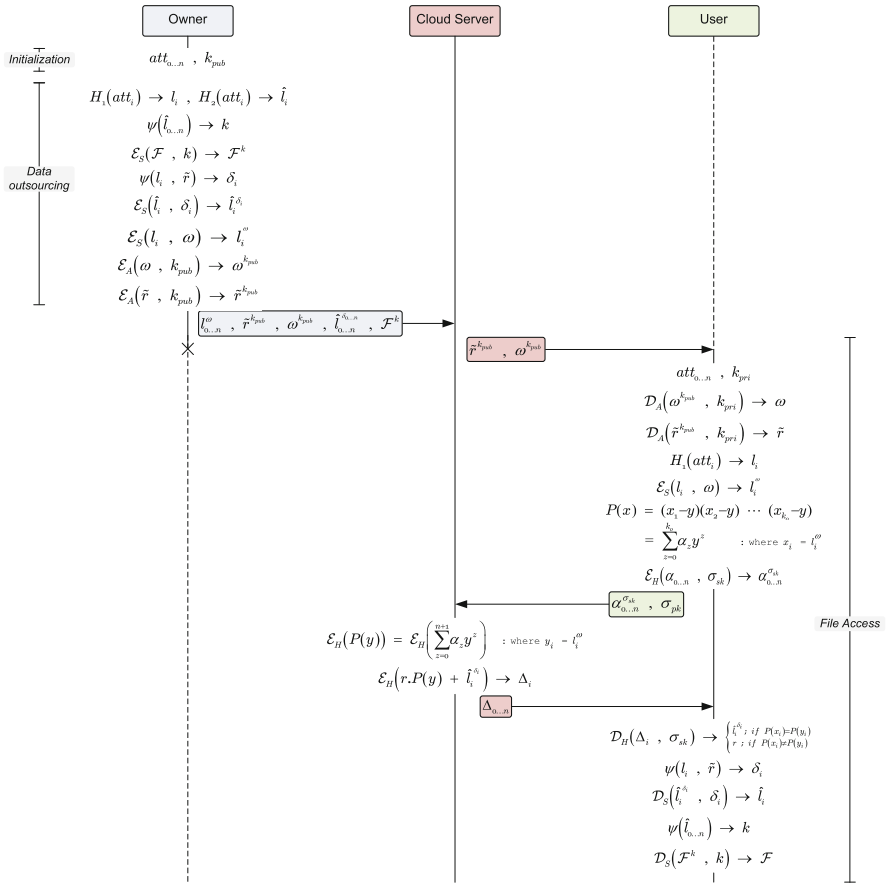

Fig. 2 Exchange of private values between the owner, cloud server and user during O-ACE

Drug Control Division, Designation: Lead Detective, In-charge of: Downtown Area. In response authority asserts the identity attributes ($att_{0...n}$) of an employee (user) that fulfills the identity selection criteria. Besides this, authority also provides the public key (k_{pub}) of the selected user.

6.2.2 Data outsourcing

Once owner has the asserted identity information ($att_{0...n}$) along with the user public key (k_{pub}), it processes them in such a way that only legitimate user manages to gain access to the outsourced data. Data outsourcing in further divided into four cohesive steps, policy modeling, data concealment, policy concealment, and delegation.

– *Policy modeling*: Asserted identity attributes ($att_{0...n}$), that uniquely defines the target user are used to model the access control policy. By using the encoding functions owner exploits these attributes in such a way that their possession ensures the legitimacy and authenticity of the user. For each asserted identity attribute

(att_i), owner computes legitimacy and mandatory values by using the encoding functions (i.e., $l_i : H_1(att_i)$ and $\hat{l}_i : H_2(att_i)$), where H_1 and H_2 are public and private encoding functions respectively. The rational of applying two separate encoding functions is elaborated in the subsequent steps.

- *Data concealment*: Mandatory values ($\hat{l}_{0\ldots n}$) generated in the previous step are used to seed the pseudo random function (ψ) which initializes the encryption key (master key: k) of an arbitrary symmetric encryption algorithm. The derived master key encrypts the data ($\mathcal{F}^k$), which is then outsourced to the cloud server. For the sake of simplicity we consider that these mandatory values are used in a cascading manner, resulting in the derivation of a single master key (i.e., $\psi(\hat{l}_0) \xrightarrow{k_0} \psi(\hat{l}_1, k_0) \xrightarrow{k_1} \cdots \xrightarrow{k_{n-1}} \psi(\hat{l}_n, k_{n-1}) \rightarrow k$) that conceals the data. However, each mandatory value can be used to derive a unique master key (i.e., $\psi(\hat{l}_i) \rightarrow k_i$) that encrypts the respective data block.
- *Policy concealment*: Mandatory values ($\hat{l}_{0\ldots n}$) ensures that master key can only be derived by the user possessing these values. In order to conceal mandatory values, a random mask ($\tilde{r}$) is generated. Then, for each mandatory value $\hat{l}_i$, a pseudo random function (ψ) is initialized with the corresponding legitimacy value l_i and $\tilde{r}$; consequently, a symmetric encryption key (mandatory key: δ_i) is generated (i.e., $\psi(l_i, \tilde{r}) \rightarrow \delta_i$). Each individual $\hat{l}_i$ is encrypted with the respective δ_i. As the policy evaluation is carried on the cloud sever by using the legitimacy values, there is a need to conceal them as well. For that owner generates a random symmetric encryption key (legitimacy key: ω), and encrypts each legitimacy values with it (i.e., $l^{\omega}_{0\ldots n}$).
  Once, the legitimacy and mandatory values are encrypted, random mask and legitimacy key are concealed by using the public key (k_{pub}) of the target user, obtained during the initialization phase.
- *Delegation*: Up till now, the owner has modeled the access control policy by encrypting the data with mandatory values. Also, mandatory values are concealed in such a way that, only authorized user can learn them. Now owner delegates the policy evaluation process to the cloud server by outsourcing the encrypted legitimacy and mandatory values ($l^{\omega}_{0\ldots n}$ and $\hat{l}^{\delta_{0\ldots n}}_{0\ldots n}$), along with the concealed random mask and legitimacy key ($\tilde{r}^{k_{pub}}$ and $\omega^{k_{pub}}$).

6.2.3 File access

In order to gain access to the outsourced data authorized user needs mandatory values. For that user engages in a delegated private matching protocol (DPM) with the cloud server, at the end of which user learns the mandatory values, if it possess the required set of legitimacy values. To evaluate the access control police and assist authorized user in deriving the valid master key, file access is divided into three cohesive step. In the first step user ensures the privacy of its identity attributes. In the second step access control policy is obliviously evaluated at the cloud server, and in the last step, user accesses (decrypts) the outsourced data if its identity attributes adhere to the access control policy.

- *Attribute preparation*: To access the outsourced data, user first obtains the concealed random mask ($\tilde{r}^{k_{pub}}$) and legitimacy key ($\omega^{k_{pub}}$) from the cloud server. Then by using k_{pri} it deciphers them to get $\tilde{r}$ and ω. After that, for each of its identity attribute att_i, legitimacy value (l_i) is computed as $H_1(att_i)$. Legitimacy values ($l_{0...n}$) are then concealed by using ω. A polynomial $\mathcal{P}(y)$ is computed having roots $l^{\omega}_{0...n}$. User then initializes a homomorphic encryption and encrypts the co-efficients ($\alpha_{0...n}$) of $\mathcal{P}(y)$ with the homomorphic secret key (σ_{sk}). After that, encrypted co-efficients ($\alpha^{\sigma_{sk}}_{0...n}$) along with homomorphic public key (σ_{pk}) are sent to the cloud server.
- *Policy evaluation*: On receiving encrypted co-efficients ($\alpha^{\sigma_{sk}}_{0...n}$), cloud server homomorphically evaluates $\mathcal{P}(y)$ with encrypted co-efficients ($\alpha^{\sigma_{sk}}_{0...n}$), for the concealed legitimacy values provided by the owner ($l^{\omega}_{0...n}$). Once $\mathcal{P}(y)$ is evaluated, cloud server computes oblivious value (Δ_i) as $\mathcal{E}_H(r.\mathcal{P}(y) + \hat{l}_i^{\delta_i})$, for each of the concealed mandatory value ($\hat{l}_i^{\delta_i}$). Then the resultant oblivious values ($\Delta_{0...n}$) are sent to the user.
- *Mandatory value recovery*: On receiving oblivious values ($\Delta_{0...n}$) user utilizes its homomorphic secret key (σ_{sk}) to decrypt ($r.\mathcal{P}(y) + \hat{l}_i^{\delta_i}$). As $\mathcal{P}(y)$ was evaluated having roots l^{ω}_i, the decryption function reveals the concealed mandatory value ($\hat{l}_i^{\delta_i}$). However, if the concealed legitimacy values generated by the user does not match with the values provided by the owner to the cloud server, decryption of Δ_i would reveal a random number, thus restraining the user to generate a valid master key.

  Once, user has the concealed mandatory values ($\hat{l}^{\delta_{0...n}}_{0...n}$), it utilizes the legitimacy value (l_i) along with the random mask ($\tilde{r}$) to initialize pseudo random function (ψ) which generates the mandatory key (i.e., $\psi(l_i, \tilde{r}) \rightarrow \delta_i$). It then deciphers the concealed mandatory values by using the mandatory keys. Once, all of the mandatory values are obtained master key is derived as illustrated in Sect. 6.2.2 (data concealment). Eventually outsourced data is accessed by using the master key.

7 Implementation

To demonstrate the practicality of O-ACE we have implemented delegated private matching process as a Java Web service and deployed it on Google App Engine. The functionality of an authority is emblemized as a web service which issues identity certificates and assertions. Confidential data (documents and images) are outsourced to the Google Cloud i.e., Google Docs. The owner and client components are realized as a standard Java program as well as an Android based mobile application. Owner component assists the owner to process the identity assertions and outsource encrypted data along with access control policy to Google Cloud. Client component processes the user's identity certificate and access the outsourced data after learning mandatory values.

Our implementation of O-ACE utilizes the standard cryptographic primitives provided by the jdk 1.6. O-ACE is realized as

- X.509 v3 certificates [23] are issued as identity certificates, created by using Bounty Castle API [5].

- Security Assertion Markup Language (SAML) [13] is utilized to assert identity assertions.
- Public encoding function uses SHA-512 as a hash function. The output of of SHA-512 is encoded as a BigInteger of arbitrary length.
- Private encoding function uses HMAC (SHA-512) with key length of 512 bits. The output of SHA-512 is encoded as a BigInteger of arbitrary length.
- Pascal Paillier cryptosystem is utilized as a homomorphic cryptosystem to ensure privacy of the data involved in delegated private matching.

Outsourced data is encrypted with AES by using 256 bit key. AES encryption key is initialized with the mandatory values. However, as our proposed system is not confined to any specific encryption algorithm, AES can be replaced with any suitable encryption algorithm, according to the security needs and computational limitations of the owner and user. Besides this our implementation of delegated private matching is not bound to any cloud provider. We have selected Google App Eninge for its native support of Java; however, our implementation can be deployed to any cloud infrastructure or platform (i.e., Amazon EC2, Microsoft Azure) that provide Java runtime and can persist private set of values (i.e., $l^{\omega}_{0...n}$ and $\hat{l}^{\delta_{0...n}}_{0...n}$).

8 Evaluation

To demonstrate the viability of our design we have conducted tests to observe its efficacy when deployed in real environment. These tests are carried out to identify the computational requirements and storage cost of O-ACE in cloud infrastructure. The evaluation process also highlights the fact that for the data owner and client O-ACE exerts reasonable computational load and can be deployed on devices having limited computational and storage capabilities i.e., smartphones and tablets.

The evaluation process is divided into two phases. First phase examines the access control policy evaluation processing time in Google AppEngine by using a single compute node having processing power of 1.20 GHz. Second phase evaluates the computation time of the data owner and client components on a PC having 2.60 GHz Dual Core Processor with 4.0 GB main memory. In addition, owner and client components are also evaluated on an Android device having 800MHz processing power.

We average each of our measurement results over 25 different trials. All of the time measurements are in milliseconds. Our evaluation results do not consider any subsequent network transmission time i.e., time required to obtain identity assertions and certificate, and network latency of Google AppEngine. These parameters are network dependent and are beyond the scope of our evaluation process.

8.1 Phase 1: performance analysis of access control policy evaluation on Google AppEngine

For the computational analysis of access control policy evaluation on Google AppEngine we consider the execution time of a billable CPU and estimated CPU usage cost for 1000 request (cpm_usd). Table 2 shows the test result of access control policy evaluation, comprises of different number of attributes i.e., 2, 4, 6, 8, *and* 10.

Table 2 Computational time and cost of access control policy evaluation on Google App Engine

No. of attributes	CPU usage time (ms)	CPU usage cost ($) for 1,000 requests
2	716	0.0198844
4	2, 139	0.0594174
6	4, 289	0.1191628
8	7, 420	0.2061114
10	1, 1340	0.3150144

Table 3 Computational time required to model the access control policy

No. of attributes	Policy modeling (ms)	
	PC	Mobile device
2	11	17
4	12	28
6	14	35
8	15	52
10	17	92

The computational time in Table 2, is directly proportional to the number of attributes in the access control policy. To evaluate access control policy Google AppEngine manipulates the polynomial co-efficients provided by the data owner, over the values sent by the client (i.e., delegated private matching). To model access control policy with n, attributes $n + 1$ co-efficients are required. Thus, access control policies involving higher number of attributes are modeled with polynomial of higher degree as compared to access control policies having fewer attributes. We modeled access control policies with arbitrary sized integer values (starting from 120 bit), thus the data points that need to be satisfied during access control policy evaluation demands more computational time. However, the size of the integer values (encoded identity assertions) can be changed according to the computational capabilities of the data owner and client.

8.2 Phase 2: performance analysis of data owner and client components

To evaluate the computational complexity of O-ACE for data owner, we consider the time required to model the access control policy. For client we consider the time required to process the identity attributes for delegated private matching and to obtain the mandatory values. Tables 3 and 4 present the test results for the data owner and client components respectively, on a personal computer and mobile device.

Owner component is only responsible for policy modeling by encoding the identity assertions into mandatory and legitimacy values and then further concealing the mandatory values with legitimacy values. Policy modeling comprises of two hashing function (public and private encoding) and a symmetric encryption function. Computational complexity of a client component is more than the owner component, as it need

Table 4 Computational time required to process the identity attributes and mandatory values recovery

No. of attributes	Attribute processing (ms)		Mandatory value recovery (ms)	
	PC	Mobile device	PC	Mobile device
2	94	168	262	255
4	134	242	846	842
6	151	248	1,744	1,779
8	183	310	2,999	3,069
10	218	420	4,536	4,652

to ensure privacy of the identity attributes and also prepare these attributes for delegated private matching. Client component first process the identity attributes and then extract the mandatory values from the response send by the cloud server. Attribute processing comprises of a hashing function for encoding the identity attributes, a function to define a polynomial for the encoded attributes and a homomorphic encryption function which ensures privacy of the defined polynomial (co-efficients). For the extraction of mandatory values client component first need to homomorphically decrypt the co-efficients send by the cloud sever. Then the decrypted co-efficients are further decrypted by using a symmetric encryption algorithm to reveal the mandatory values.

9 Security analysis

In this section we exam the computational complexity of a malicious user to gain access to the outsourced data. For that, a malicious user would need the mandatory values ($\hat{l}_{0...n}$) with which valid master key can be derived. The proposed system uses the standard cryptographic primitives to ensure privacy of the outsourced data; thus, inheriting their computational complexities. As discussed in Sects. 6 and 7, O-ACE uses one way hash function to model the access control policy. Symmetric encryption function is used to ensure that only authorized users can decrypt the response send by the cloud server. Private matching is used to obliviously evaluate the access control policy against the identity attributes provided by the user. Homomorphic encryption is used to ensure privacy of access control policy and identity attributes. Reader may refer to [16,20], and [31], for the security analysis of aforementioned cryptographic primitives.

As discussed O-ACE inherits the computational complexities of the underlying cryptographic primitives. However, cloud server, client and even authority can act maliciously by teaming up with each other to compromise privacy of the outsourced data. In the subsequent security analysis we examine O-ACE against the malicious behavior of the involved entities. First, we discuss the efficacy of O-ACE when cloud server does not perform its task honestly, falsifying our assumption that cloud server is honest but curious (see Sect. 5.4). Second, we discuss, up to what extent O-ACE provides protection when multiple malicious users can learn the partial set of mandatory values ($\hat{l}_{0...n}$) and combine them to derive a master key. Third, we review the efficacy

of O-ACE when authority seeps out information about the asserted identity attributes to the unauthorized users.

9.1 Malicious cloud server

There are two possibilities through which cloud server can assist unauthorized users to learn the mandatory values ($\hat{l}_{0...n}$). Firstly, bypassing the access control policy evaluation and simply handing over the concealed mandatory values ($\hat{l}^{\delta_{0...n}}_{0...n}$) to the user(s). Secondly, by incorrectly evaluating the access control policy i.e., instead of the adding a random value (r) to compute oblivious values (i.e., $\Delta_{0...n}$) it adds a zero value. In both of these cases unauthorized user learns the mandatory values ($\hat{l}^{\delta_{0...n}}_{0...n}$) that are encrypted with the masked legitimacy values i.e., $\psi(l_{0...n}, \tilde{r}) \rightarrow \delta_{0...n}$. In order to decipher the mandatory values unauthorized user would have to gain access to the encoded attributes ($H_1(att_{0...n}) \rightarrow l_{0...n}$), and random mask ($\tilde{r}$) which is encrypted with the authorized user's public key i.e., $\tilde{r}^{k_{pub}}$.

As access control policy is obliviously evaluated, cloud server cannot identify the user that can access the outsourced data successfully. Thus malicious user would have to try all possible encoded attributes in order to learn the correct legitimacy values ($l_{0...n}$). This could be a trivial task if possible identity attributes are limited as legitimacy value can be learned by simply applying the publicly known encoding function. However, to mitigate this threat we have masked the legitimacy value before it can be used to conceal the mandatory values i.e., $\psi(l_{0...n}, \tilde{r}) \rightarrow \delta_{0...n}$. This random mask ($\tilde{r}$) is encrypted with the authorized user's public key. In order to decipher the mandatory values unauthorized user would have to decipher $\tilde{r}^{k_{pub}}$ without private key, thus making its computational complexity equal to asymmetric encryption algorithm.

9.2 Malicious clients

We now consider the scenario in which cloud server is performing its task honestly; however, unauthorized users team up to gain access to the outsourced data otherwise not allowed. In order to successfully gain access to the outsourced data, unauthorized users must team up in such a way that collectively they can learn the mandatory values ($\hat{l}_{0...n}$) which can be used to derive the master key. As access control policy is obliviously evaluated on the cloud server, unauthorized users cannot identify the identity attributes ($att_{0...n}$) which conform to the access control policy. Individual unauthorized user would have to learn subset of the mandatory values ($\hat{l}_{0...\acute{n}} \subset \hat{l}_{0...n}$) and then combine them altogether to learn the entire set i.e., $\hat{l}_{0...\acute{n}_1} \cup \hat{l}_{0...\acute{n}_2} \cup \ldots \hat{l}_{0...\acute{n}_n} = \hat{l}_{0...n}$. It is a nontrivial task to identity the team members that can completely learn the mandatory values ($\hat{l}_{0...n}$).

Similar to the scenario discussed in the previous section, if there are limited number of the identity attributes then unauthorized users can effortlessly combine their partial set of mandatory values. Although mandatory values can be obtained but still unauthorized users need the random mask ($\tilde{r}$) which is used to mask the legitimacy values (i.e., $\psi(l_{0...n}, \tilde{r}) \rightarrow \delta_{0...n}$). As cloud server is working honestly unauthorized users

cannot obtain the random mask encrypted with the user's public key with whom data owner wants to share the outsourced data. Even if random mask can be learned still unauthorized users would have to decipher $\tilde{r}^{k_{pub}}$ without private key (k_{pri}), making its computational complexity equal to asymmetric encryption algorithm.

9.3 Malicious identity provider

O-ACE is highly dependent on the identity assertions and attributes provided by an authority. In the descriptive detail of O-ACE we assumed that authority is a trustable entity (see Sect. 5.4). For the security analysis we now consider that the authority is working maliciously and seeps out information about the identity assertions and attributes to unauthorized users and cloud server. They can then use it to learn the mandatory values ($\tilde{l}_{0\ldots n}$). However; similar to the scenarios discussed previously access to the mandatory values ($\tilde{l}_{0\ldots n}$) do not compromise privacy of the outsourced data. The attacker would need the random mask ($\tilde{r}$), which masks the legitimacy values. As $\tilde{r}$ is concealed with the authorized user's public key the attacker would have to revert back the asymmetric encryption without private key (k_{pri}).

10 Discussion

Access control policy evaluates access privileges of an entity and grants or denies access to the required resources (i.e., data, services, and business logic). As it contains information about the valid access parameters it must be evaluated by a trusted entity (i.e., policy evaluator). An untrusted or malicious entity can exploit it to gain unauthorized access to the resources. To achieve data privacy in an untrusted domain (i.e., cloud storage), existing systems either rely on a trusted third party or data owner to manage and distribute appropriate decryption keys to the authorized users. These methodologies tend to decrease utility of cloud storage by restricting data owner to stay always online and exerting computational load on the access control policy evaluator.

In order to achieve data privacy without engaging trusted third party or data owner itself we proposed oblivious access control policy evaluation in cloud storage called O-ACE. Through O-ACE cloud service provider can grants or denies access to the outsourced resources without learning any useful information about the access control policy. We extended the notion of private matching to an untrusted domain called DPM. Privacy of the identity attributes and obliviousness of the access control policy is achieved by DPM. Through DPM authorized user can learn the values that derive the valid decryption key; whereas, unauthorized learns the random value which does not reveal any information about the access control policy.

For the security analysis of O-ACE, we analyzed the risk of privacy breach when entities (i.e., cloud server provider, user, and identity provider) behave maliciously. Even if malicious entities team up with each other confidentiality of the outsourced data is still preserved. To demonstrate the security viability of O-ACE we analyzed the situation in which all of the entities behaved maliciously. Even in the worst case the computational complexity for the attackers was equivalent to reverting asymmetric encryption without valid key pair.

So far attribute based encryption (ABE) [21] has leveraged the data sharing systems to achieve access control policy evaluation on the client side. However, ABE is 100–1,000 times slower then RSA [38]. Current implementation of ABE uses Bison and YACC parsing packages to extract access control policy from the cipher text, thus it is difficult to realize it for the mobile devices. Systems based on ABE has to generate secret key for the legitimate users, exerting computational load on the data owner. In O-ACE data owner just need to disseminate random seed to the legitimate users. Apart from that, to revoke a user in system using ABE, data owner needs to update the access control policy so that revoked user cannot conform to it. Whereas, in O-ACE data owner only needs to update the random seed ($\tilde{r}$) and access of the revoked user can be restrained as it cannot learn the mandatory values without the random seed ($\tilde{r}$).

To highlight the practicality of O-ACE we realized a cloud based data sharing system which assist data owner to achieve fine-grained access control over the outsourced data. However, the applicability of O-ACE is not restricted to data sharing systems only. It can be used to provision services whose response can only be decrypted by the authorized users. Similarly it can also be used to provide access to compute and storage resources (i.e., password protected virtual machines and databases). With O-ACE there is no need to engage multiple administrative entities (e.g., Key Manager, Policy Manger) to restrain illicit data access. Cloud service provider obliviously performs the task of Key Manager by storing and distributing the mandatory values, and also takes over the responsibility of Policy Manager by obliviously matching the identity attributes with valid access parameters.

With O-ACE we have achieved fine-grained access control over the outsourced data. However, so far O-ACE does not support logical conditions for access control policy evaluation like ABE does. Support for the logical conditions can be added by encoding the range values, and distributing the appropriate mandatory values during policy evaluations. Although it will increase the number of legitimacy and mandatory values cloud service provider has to maintain; however, it will not effect security of O-ACE. The current implementation of O-ACE assumes that identity attributes assigned to a user do not contain any private information and can be asserted to the data owner. To avoid disclosing any private information, identity provider can assert the masked identity assertions to the data owner, and disseminates the mask to the respective user.

Through the experiments we have shown that O-ACE can be realized for the devices having limited computational capabilities. Also it exerts reasonable computational load on the cloud service provider, casting around 0.01–0.30 dollars per 1,000 requests. Additionally, it uses standard cryptographic primitives which are natively provided by the most programming languages. Thus, it can be effortlessly realized according to the security requirements and computational capabilities of the involved entities.

11 Conclusion

In this paper, we proposed access control policy evaluation procedure O-ACE for cloud based data sharing systems. With O-ACE cloud server obliviously evaluates the access control policy without learning any useful information that can be used to

compromise privacy of the outsourced data. Oblivious access control policy evaluation is achieved by extending the private matching to delegated private matching, which assist authorized user to learn the secret values that are then used to derive the valid decryption key. Whereas, unauthorized users learn nothing more than the random numbers; even if they team up with the cloud server provider or identity provider. Through the computational analysis we showed that O-ACE's CPU usage cost is only 0.01–0.30 dollars for every 1,000 requests. The security analysis has shown that for each entity the complexity to compromise privacy of the outsourced data is equivalent to decrypt cipher text encrypted with asymmetric encryption without appropriate key pair. O-ACE is not restricted to data sharing applications only, it can also be exploited to provision compute resources and even can be used to grant access to application services.

Acknowledgments This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MEST) (No. 2011-0030823). Also, this research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0020515), and was supported by a grant from the Kyung Hee University in 2011 (KHU-20111372).

References

1. Amazon cloud drive—anything digital, securely stored, available anywhere. https://www.amazon.com/clouddrive/
2. Dropbox—simplify your life. https://www.dropbox.com/
3. Google app engine—run your web applications on google's infrastructure. http://code.google.com/appengine/
4. Google docs—create and share uour work online. http://www.google.com/google-d-s/b1.html
5. The legion of the bouncy castle. http://www.bouncycastle.org/
6. Microsoft office live—access, edit, and share documents from anywhere. http://www.officelive.com
7. Windows live skydrive—online document storage and file sharing. http://www.windowslive.co.uk/skydrive
8. Zoho—suite of online web applications. https://www.zoho.com/
9. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53:50–58. doi:10.1145/1721654.1721672
10. Brunette G, Mogull R, et al (2009) Security guidance for critical areas of focus in cloud computing. http://www.cloudsecurityalliance.org/csaguide.pdf
11. Buyya R, Yeo CS, Venugopal S (2008) Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In: Department of Computer Science and Software Engineering (CSSE), The University of Melbourne, Australia, pp 10–1016
12. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility. Elsevier Science Publishers B. V., Amsterdam, pp 599–616. doi:10.1016/j.future.2008.12.001
13. Cantor S, Kemp J, Philpott R, Maler E (2005) Assertions and protocols for the oasis security assertion markup language (saml) v2.0. http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf
14. Coull SE, Green M, Hohenberger S (2011) Access controls for oblivious and anonymous systems. ACM Trans Inf Syst Secur 14:10:1–10:28. doi:10.1145/1952982.1952992
15. Ellison C, Frantz B, Lampson B, Rivest R, Thomas B, Ylonen T (1999) Spki certificate theory
16. Freedman M, Nissim K, Pinkas B (2004) Efficient private matching and set intersection. Springer, New York, pp 1–19
17. Frikken K, Atallah M, Li J (2006) Attribute-based access control with hidden policies and hidden credentials. IEEE Trans Comput 55(10):1259–1270

18. Geron E, Wool A (2007) Crust: cryptographic remote untrusted storage without public keys. In: Fourth international IEEE security in storage workshop, 2007. SISW '07, pp 3–14. doi:10.1109/SISW.2007.9
19. Goh EJ, Shacham H, Modadugu N, Boneh D (2003) Sirius: securing remote untrusted storage. In: Proceedings of network and distributed systems security (NDSS) symposium 2003, pp 131–145. doi:10.1.1.104.6458
20. Goldreich O, Israel R, Dana T (1995) Foundations of cryptography
21. Goyal V, Pandey O, Sahai A, Waters B (2006) Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on computer and communications security, CCS '06, ACM, New York, pp 89–98. doi:10.1145/1180405.1180418
22. Holt JE, Bradshaw RW, Seamons KE, Orman H (2003) Hidden credentials. In: Proceedings of the 2003 ACM workshop on privacy in the electronic society, WPES '03. ACM, New York, pp 1–8. doi:10.1145/1005140.1005142
23. Housley R, Polk W, Ford W, Solo D (2002) Internet x.509 public key infrastructure. http://www.ietf.org/rfc/rfc3280.txt
24. Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K (2003) Plutus: scalable secure file sharing on untrusted storage. In: Proceedings of the 2nd USENIX conference on file and storage technologies. USENIX Association, Berkeley, pp 29–42. http://dl.acm.org/citation.cfm?id=1090694.1090698
25. Kamara S, Lauter K (2010) Cryptographic cloud storage. In: Proceedings of the 14th international conference on financial cryptograpy and data security, FC'10. Springer, Berlin, pp 136–149. http://dl.acm.org/citation.cfm?id=1894863.1894876
26. Kamara S, Papamanthou C, Roeder T (2011) Cs2: a searchable cryptographic cloud storage system. TechReport MSR-TR-2011-58, Microsoft Research
27. Kaufman LM (2009) Data security in the world of cloud computing. IEEE Secur Privacy 7:61–64. doi:10.1109/MSP.2009.87
28. Kaufman LM (2009) Data security in the world of cloud computing. IEEE Secur Privacy 7:61–64. doi:10.1109/MSP.2009.87
29. Li J, Li N (2006) Oacerts: oblivious attribute certificates. IEEE Trans Dependable Secur Comput 3:340–352. doi:10.1109/TDSC.2006.54
30. Li N, Mitchell JC, Winsborough WH (2002) Design of a role-based trust-management framework. In: Proceedings of the 2002 IEEE symposium on security and privacy. IEEE Computer Society, Washington, p 114. http://dl.acm.org/citation.cfm?id=829514.830539
31. Paillier P (1999) Public key cryptosystems based on composite degree residuosity classes. In: Proceedings of the 17th international conference on theory and application of cryptographic techniques, EUROCRYPT'99. Springer, Berlin, pp 223–238. http://dl.acm.org/citation.cfm?id=1756123.1756146
32. Paillier P (2000) Trapdooring discrete logarithms on elliptic curves over rings. In: Proceedings of the 6th international conference on the theory and application of cryptology and information security: advances in cryptology, ASIACRYPT'00. Springer, London, pp 573–584. http://dl.acm.org/citation.cfm?id=647096.716885
33. Pearson S (2009) Taking account of privacy when designing cloud computing services. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, CLOUD '09. IEEE Computer Society, Washington, DC, pp 44–52. doi:10.1109/CLOUD.2009.5071532
34. Ristenpart T, Tromer E, Shacham H, Savage S (2009) Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on computer and communications security, CCS '09. ACM, New York, pp 199–212. doi:10.1145/1653662.1653687
35. Vimercati SDCd, Foresti S, Jajodia S, Paraboschi S, Samarati P (2007) Over-encryption: management of access control evolution on outsourced data. In: VLDB, pp 123–134 (2007)
36. Samarati P, Vimercati SDCd (2001) Access control: policies, models, and mechanisms. In: Revised versions of lectures given during the IFIP WG 1.7. International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures, FOSAD'00. Springer, London, pp 137–196 (2001). http://dl.acm.org/citation.cfm?id=646206.683112
37. Singh A, Liu L (2008) Sharoes: a data sharing platform for outsourced enterprise storage environments. In: IEEE 24th international conference on data engineering, 2008, ICDE 2008, pp 993–1002. doi:10.1109/ICDE.2008.4497508
38. Sun J, Zhu X, Fang Y (2010) A privacy-preserving scheme for online social networks with efficient revocation. In: 2010 Proceedings IEEE, INFOCOM, pp 1–9 (2010). doi:10.1109/INFCOM.2010.5462080

39. Tang Y, Lee PPC, Lui JCS, Perlman R (2010) Fade: secure overlay cloud storage with file assured deletion. In: SecureComm, pp 380–397
40. Wang W, Li Z, Owens R, Bhargava B (2009) Secure and efficient access to outsourced data. In: Proceedings of the 2009 ACM workshop on cloud computing security, CCSW'09. ACM, New York, pp 55–66. doi:10.1145/1655008.1655016
41. Yao J, Chen S, Nepal S, Levy D, Zic J (2010) Truststore: making amazon s3 trustworthy with services composition. In: 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing (CCGrid), pp 600–605 (2010). doi:10.1109/CCGRID.2010.17