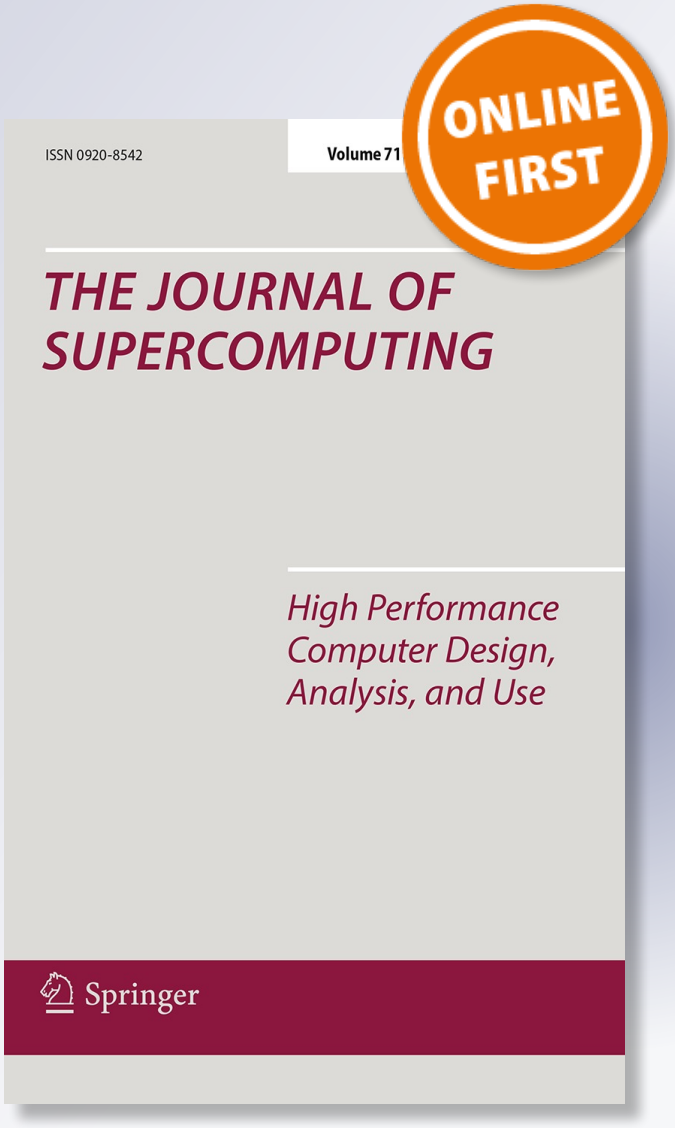
Oblivious user management for cloud-based data synchronization

Mahmood Ahmad, Zeeshan Pervez, Taechoong Cheong & Sungyoung Lee

The Journal of Supercomputing
An International Journal of High-Performance Computer Design, Analysis, and Use

ISSN 0920-8542

J Supercomput
DOI 10.1007/s11227-014-1369-5



Springer

J Supercomput
DOI 10.1007/s11227-014-1369-5

Oblivious user management for cloud-based data synchronization

Mahmood Ahmad · Zeeshan Pervez · Taechoong Cheong · Sungyoung Lee

Abstract One of the main issues with data sharing in cloud environment is to manage user access and its auto revocation in a controlled and flexible way. The issue becomes more complex when privacy on user access has to be ensured as well to hide additional leakage of information. For automatic revocation over cloud data, access can be bounded within certain anticipated time limit so that the access expires beyond effective time period. This time-oriented approach is more rigid and not a one-size-fits-all solution. In certain circumstances, exact time anticipation is not an easy choice. Instead, the alternate solution could be task oriented to restrict user beyond certain number of permissible attempts to access the data. We have proposed oblivious user management (OUM) in which a user can have access on cloud data for certain number of attempts without imposing any time restriction. For user authorization and her subsequent revocation, owner will perform one time setup activity and that is same for all users. The model also alleviates the burden of managing different access parameters at user end with each request as she will always use the same parameter for all valid attempts. Our approach also conceals the privacy of user attempts throughout the communication. Hiding this information helps to avoid distinguishing importance of

M. Ahmad · T. Cheong (✉) · 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: tcchung@khu.ac.kr

M. Ahmad
e-mail: rayemahmood@oslab.khu.ac.kr

S. Lee
e-mail: sylee@oslab.khu.ac.kr

Z. Pervez
School of Computing, University of the West of Scotland, Paisley PA1 2BE, UK
e-mail: zeeshan.pervez@uws.ac.uk

Published online: 11 January 2015

particular user that has more authorization over others. Evaluation results have proved that OUM hides $(N-1)$ number of permissible attempts until Nth request arrives at Cloud Storage. The Performance analysis conducted on Google App Engine revealed that the cost of operations performed in OUM is within the range of 0.097–0.278 $ per 1,000 requests.

Keywords Cloud storage · Private matching · Oblivious access · Privacy

1 Introduction

Cloud computing has a profound impact towards society over the past few years. As a significant technology trend, its evolutionary role has reshaped IT services manifold [1,2]. Features like scalability, elasticity and fault tolerance have made cloud computing most suitable for a robust business environment. Usage of cloud computing is quite economical due to its pay-per-use model where a subscriber does not need to manage the computing resources [3,4]. These features have made it possible for cloud proponents to afford computers and storage resources that, just a few years ago, would have been available only to enterprises [5,6].

The need for massive storage space and its intelligent utilization are two main reasons in building towards the epitome of cloud computing which otherwise not possible on commodity hardware [7,8]. It is believed that the amount of data is doubling after every eighteen (18) months and world creates five exabytes of information every 2 days [9,10]. This trend will make data growth 44 times between 2009 and 2020 [11]. Besides social media, data generated from machines and sensors massively contribute to this data explosion [12]. The cloud storage accommodates this intensity, volume and variety of data easily which is then used to extract functional intelligence and hidden knowledge from it. Specialized and highly skilled professionals transform tons of data into crux of useful information, e.g., [12,13]. Companies that once use to rely on their in house data streams have now started utilizing this functional knowledge as a catalyst for effective business proliferation. Realization of this fact has changed the business landscape where organizations are now totally data driven [14].

Authorization on cloud data has various access models depending upon its type [2] and priorities set by its owner and user. A public data [15,16] have an open access, whereas sensitive data, e.g., related to health, individual or organization are protected with encryption [17,18] and are accessible by authorized users only. Temporal data bear its value for certain interval of time and customized data are prepared for a selected range of audience [12,13,19]. If outsourced data are sensitive or valuable, owner concern would be to protect it from unauthorized access by applying techniques of encryption or anonymization [17,18,20]. While utilizing cloud services, it is desired that leakage on data contents must be avoided or kept minimal at least. For this purpose, encrypted outsourcing is strongly encouraged [21,22] which also hides details on data for a curious cloud service provider (CSP). From user perspective who will use this data has altogether different requirements. She might be interested to hide her identity or does not want to show off her authorization limit in advance for a particular resource. For example, Alice, Bob and Malory are solving a criminal case.

Alice is the lead detective and has higher authority and access permission on this case. To avoid any external influence on Alice, this information has to be concealed until the case is solved completely. In this task-oriented scenario and others similar to this, besides protecting the original data, achieving privacy on user access is equally important.

Leakage of unnecessary information is not limited with outsourced data alone, user access models and patterns on access log is another avenue for a curious CSP to know about additional information. Considering another example where a resource being accessed frequently by a large pool of users reveals its importance. A single resource, which is accessed sequentially by multiple users from within an organization, can sketch out the work flow of that organization. A particular resource, which is accessed on cloud, usually ends up in a valid reply or with response of "Not Found" which tells about its presence or absence. While preventing the unsolicited disclosure of sensitive information, additional measures like oblivious access policies [23–25] and oblivious term matching [44] help to avoid indirect leakage of information. It includes Ciphertext-Policy attribute-based encryption (CP-ABE) [26] and Key-policy ABE [27], which are the branches of Attribute-Based Encryption (ABE) [28]. Besides all these access patterns, there is another parameter, which is a potential resource for leaking the unnecessary and unwanted information and that is related with user management.

The user that has been granted with permission to access cloud data usually requires to be revoked for subsequent requests at a later stage. This revocation can be done by the owner who gave her the authorization at first step; however, this activity becomes hectic when number of users and frequency of authorization and revocation are very high. Therefore, the owner of information has choice to delegate authority of user revocation to CSP or semi trusted third party (TTP). A similar realization on user revocation has been addressed in [31] where user access is bound for certain time period. The methodology presented in [31] also meets the user's concerns where she does not want to disclose her access period in advance. In this time-oriented system, a user access remains active on cloud data for predetermined time. This technique might work well where anticipation of time is trivial. For situations where time anticipation is hard to determine in advance, this technique would not be that much effective. We will augment this with a simple example where a user Alice has been granted permission for certain period of time, say, until March 2014 for a particular data resource on a cloud. Alice would like to use these data with additional information which she is expecting from another resource and that is delivered to her in April 2014. In this situation, Alice access rights on cloud data have already been expired for which she needs to be given with fresh account or extension in the existing one. Reliance on external factor which is time independent can make entire situation task oriented rather time oriented. The permission that has been granted to Alice has to be availed before her time expires, no matter it is useful for her or not, which is too strict. The solution for this scenario could be to allow herself for certain number of permissible attempts which should be independent from time and that too without disclosing them in advance. She can access the data when it is most suitable for her and her authorization will become ineffective after the permissible attempts have been availed.

Through OUM, following contributions have been made in the area of oblivious access policies while interacting with cloud data:

- for user revocation, owner does not have to be online 24/7,
- task-oriented provisioning of cloud data makes it convenient for users to access data in their own time,
- users concern for hiding their permissible attempts from their peers as a business secret is possible with OUM,
- oblivious evaluation of user request does not help cloud service provider (CSP) to infer total number of permissible attempts at any stage before all have been availed. For this purpose, services of trusted third party have been utilized.

The rest of the paper is structured as follows. Section 2 is on related work. Technical preliminaries used in this paper are presented in Sect. 3. System Model, design goal and assumptions are given in Sect. 4. Construction of OUM including main idea is in Sect. 5. Detail on implementation is given in Sect. 6. Section 7 is on result evaluation. Salient features, discussion on user request patterns and system limitations are given in Sect. 8. Finally, we conclude our paper and provide future direction in Sect. 9.

2 Related work

We assume that user management starts with her authorization and ends with revocation over cloud data. To incapacitate any single out of N users on encrypted data, there exist few approaches. The first one, which is a naive solution, is to re-encrypt the complete data and re-distribute the new decryption keys to $(N-1)$ users. This approach is quite computationally intensive when frequency of users entering and leaving the system is very high. In addition, data owner has to be online all the time to execute this operation, which is difficult to maintain 24/7. Relaxing responsibilities for data owner can be achieved using Proxy Re-Encryption (PRE) [18,29] either through TTP or through CSP. PRE converts a cipher text that can be decrypted by Alice into another cipher text that can be decrypted by Bob. This whole operation hides the actual data being transformed from one key to another. Work done by [30] is considered a pioneer to combine Key-Policy ABE (KP-ABE) and PRE to delegate most of the computation tasks involved in user revocation to the CSP. Still, activating PRE by CSP awaits for an event to trigger, which is again the responsibility of data owner. To handle this issue, activation of PRE has been coupled with time in such a way that user authorization expires on predetermined time [31]. This approach also conceals user effective time period from CSP to know in advance. Allowing CSP to know about user effective time period on cloud data can reveal user importance who has permission for longer time than other users.

The outsourced data in a public cloud have its own importance and value. Employing encryption on this trove of information is mostly a favorable choice by the data owner. On the other hand, hiding access patterns is more desirable and appreciated by users who will use these data. Recent research, which is cited in upcoming discussion, is related to concealing user access and possible leakage of information on her behalf. Instead of giving exact IDs to authorized users, certain descriptive attributes are more suitable for identification. If Alice is working as a manager in a company then instead

of using Alice as her ID, the attribute of 'manager' is preferred. The same idea has been chosen in (CP-ABE) [32,33] by employing the Ciphertext-policy attribute-based encryption (CP-ABE) Using this technique, identities of users can be concealed using feature of attributes instead of exact IDs.

User revocation is a complex process and very few techniques have been used to handle this issue in cloud model. In [31], the idea of time-based proxy re-encryption has been employed. In this work, usage of CP-ABE is extended with HABE [34,35] using the concept of time to trigger automatic proxy-re encryption. In this approach, the granularity of time has been sliced into three layers, namely year, month and day. This solution is prepared for situation where time anticipation can be determined in advance before a user is granted access over cloud data. Revoking user access rights using this methodology is appealing for situations where time anticipation is a trivial. In task-oriented situations, this solution will be least effective.

The authors in [46] employed the functional encryption as a new way with larger possibilities for sharing and using encrypted data. In comparison to previously presented schemes of ABE, the proposed work is claimed to be fully secure with attribute hiding properties with both ABE and predicate encryption (PE). Chase [47] presented the problem of ABE with multiple authorities. To help the audit process and to discover the identities of misbehaving users leaking their secret keys to others is presented in [48]. While introducing the concept of access control list (ACL), the work presented in [49] is based on symmetric key and public key cryptographic system. In this model, the data ownership first specifies an ACL for data and then encrypts the data with symmetric key which is then encrypted with the public key of users in the ACL.

Besides revoking a user from subsequent request on cloud data, it is also very important to determine the mechanism with which CSP will evaluate and come to know that further access has to be ceased. The simple solution is to let CSP know about effective time of all users in advance. If longer access duration implies user importance, then it is open to CSP with this naive solution.

The work in [36], which requires users to expose their tickets to CSP, may also expose effective time of each ticket. This approach might also be interesting to know for a curious CSP when two competitors are assigned on a same resource where authorization of one user is higher than the other. For users who consider this information as their business secret would avoid to go with this solution or any other similar to it. In this paper, we have considered two issues, which are subset of above discussion: first, automatic user revocation and that too independent from time; second, hiding user authorization attempts throughout her communication with cloud. For CSP, these attempts would remain unknown to predict and hard to distinguish between any number of users. In OUM's design, we have utilized the cryptographic primitives of homomorphic encryption [39] and private matching [37] to meet the desired results.

3 Technical preliminaries

Before we elaborate the design and working methodology of OUM, we introduce few preliminaries used in its development.

3.1 Homomorphic encryption

A cryptographic mechanism is said to be homomorphic if its encryption function $\mathcal{E}_H$ holds the property, $\mathcal{E}_H(x) \times \mathcal{E}_H(y) = \mathcal{E}_H(x + y)$. A homomorphic encryption is said to be *semantically secure* if $\mathcal{E}_H$ reveals no information about x and y, and making it computationally infeasible to ascertain and distinguish the case where $x \neq y$ and $x = y$ [38].

Paillier [39] proposed a public key encryption scheme which is additively homomorphic, and consists of subsequent fundamental algorithms.

3.1.1 Key generation

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

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

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

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

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

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

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

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

3.1.3 Decryption

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

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

3.1.4 Homomorphic operation

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

$$\mathcal{E}_H(m_1, y_1) = g^{m_1} {y_1}^n (\text{mod } n^2)$$
$$\mathcal{E}_H(m_2, y_2) = g^{m_2} {y_2}^n (\text{mod } n^2)$$

$$\overline{\mathcal{E}_H(m_1, y_1).\mathcal{E}_g(m_2, y_2) = g^{m_1+m_2}(y_1 \cdot y_2)^n (\text{mod } n^2)}$$
$$= \mathcal{E}_H(m_1 + m_2) \tag{7}$$

3.2 Private matching

Private matching (PM) [37] is a value matching protocol. It assists two interactive parties to find set intersection over their private set of values, without revealing any element of their private set to each other.

Suppose, there is a client $\mathcal{C}$ and a server $\mathcal{S}$. $\mathcal{C}$ has its own private set of values $\mathcal{X} : \{x_1, x_2 \ldots x_n\}$, and $\mathcal{S}$ has values $\mathcal{Y} : \{y_1, y_2 \ldots y_n\}$. $\mathcal{C}$ wants to compute set intersection with $\mathcal{S}$ over the private set of values (i.e., $\mathcal{X}$, $\mathcal{Y}$). To identify the commonalities between $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{C}$ computes a polynomial (see Eq. 8), whose roots are members of $\mathcal{X}$.

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

$\mathcal{C}$ then sends the homomorphicly encrypted coefficients ($\hat{\alpha}_{0\ldots n}$) of $P(x)$ to $\mathcal{S}$. 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., $\mathcal{E}_H(r \cdot P(y) + y)$, 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 $\left\langle z \subseteq \bigcap \mid (z \in \mathcal{X}) \wedge (z \in \mathcal{Y}) \right\rangle$. 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 $\mathcal{X}$.

Although we have used the same idea in OUM but instead of creating polynomial with higher degree, we used only U_a and ψ, where ψ is offset value of user such that

$$P(x \in \mathcal{X}) = (x - \psi)(x - (\psi + U_a) \tag{9}$$

Here, ψ is unique for each user.

4 Models, design goals, and assumptions

4.1 System model

CSP, TTP, data user and data owner are considered to be the involved entities to realize oblivious evaluation of user request in cloud storage. For brevity, we have also referred them as cloud storage, third party, user and owner, respectively. Due

to large volume and variety of data, it is shared with users through synchronization service. Upon receiving request, cloud storage sync the updated version of data with authorized user. The owner owns the rightfully processed data useful for its users on pay per access model. The pricing model on data synchronization depends upon its volume and number of user attempts. Only authorized users who are willing to pay for this information have access to the data synchronization facility on cloud storage. Third party acts as a mediator and transforms the user request obliviously for cloud storage. Cloud server obliviously evaluates forwarded request and only learns that the request falls within the subscription or not. In case the request falls within the subscription limit, cloud storage synchronizes data with user or denies otherwise.

4.2 Security model

The protection mechanism like encryption is a non-trivial barrier for a malicious or curious user to know about unauthorized data. The indirect knowledge that slips away through legitimate communication is another source for acquiring allied information. In OUM, allied information means 'user permissible attempts' and legitimate communication means 'communication made by authorized users'. Our model of OUM safeguards privacy of user access by hiding the permissible attempts. Neither the third party nor curious cloud storage can learn about exact attempts in advance or during evaluation process of user request. Partially and obliviously execution of user request at both, third party and cloud storage helps to achieve this model. This model further incapacitates cloud storage to calculate the entire life cycle for user access in advance. The indication for cloud storage to stop further requests is popped only when a user forwards her request beyond authorized quota. This is the only situation where CSP will learn about it.

4.3 System design goal

Controlling and restricting user access on cloud storage have been met with various access policies and security models depending upon requirements of user and owner. These access policies facilitate owner and user according to their needs while exploiting cloud resources at maximum. OUM is also claimed as a similar system but with distinctive design features. In OUM, bar of managing user revocation is handled by third party and CSP, whereas privacy of user authorization is also preserved during the legitimate communication. Further, user is free to avail her access rights independent of time restriction. The automatic termination of user access under constant set of operations boosts up system's efficiency. The pivotal design goal of our system is threefold. First, any particular user has same access mechanism and execution time irrespective of how many attempts she is authorized with. Second, the owner does not need to revoke authorized user herself; the system mechanism itself informs on it. Third, the cloud storage and semi trusted third party will learn nothing about the valid attempts until all have been availed.

Table 1 Notations used in the descriptive detail of OUM

Notation	Description
$\mathcal{D}_p$	Periodically updated data that is outsourced for sharing on a cloud
$\mathcal{U}_a$	User attempts for which authorization is granted
∂	User access parameter for unique identification
$\mathcal{E}_H, \mathcal{D}_H$	Homomorphic encryption and decryption algorithms
$\sigma_{\text{pk}}, \sigma_{\text{sk}}$	Public and secret keys for homomorphic encryption algorithm
$\mathcal{E}_A, \mathcal{D}_A$	Asymmetric encryption and decryption algorithms
$k_{\text{pub}}, k_{\text{pri}}$	Public and private key pair for asymmetric encryption algorithm
$\alpha_{0\ldots n}$	List of coefficients of a polynomial P that defined for TTP
$\Delta_{y_{1\ldots n,n+1}}$	Oblivious values: result range of homomorphic evaluation at CSP for a particular user with n number of valid requests. Oblivious value for $(n+1)$th request is denoted with $\Delta_{y_{n+1}}$
ψ	Offset value for user
Ω	Constant operation performed by TTP with user request over $\alpha_{0\ldots n}$
$\Upsilon_{1\ldots n}$	Partialy computed result at TTP
$\uplus$	Constant operation performed by CSP on $\Upsilon_{1\ldots n}$ which ends up in $\Delta_{y_{1\ldots n,n+1}}$
$\mathcal{E}_v$	Echo value, which repeats itself only twice during the entire lifetime of user access period such that. $\mathcal{E}_v = \Delta_{y_1}$ and $\mathcal{E}_v = \Delta_{y_{n+1}}$
$\mathcal{R}_v$	The evaluation results at CSP with each user request other than $\mathcal{E}_v$ are considered as residual values such that $\mathcal{R}_v = \sum_{i=2}^{n} \Delta_{yi}$
Œ	Digital certificate of TTP

4.4 Assumption and notations

Oblivious user management focuses on enabling task-oriented and privacy-aware data access from cloud storage. We intentionally neglected the details of data sharing from security and privacy point of view. Readers may refer to [40] for more details on efficient and secure data sharing in cloud storage. Third party is assumed to process the partial computation with honest operation and rightfully issuing the offset value for each user request. Table 1 illustrates the notations that we use to explain the core concept of OUM.

$\mathcal{D}_p$ represents the periodically updated data outsourced to cloud storage, accessible by authorized subscribers only. $\mathcal{U}_{id}$ represents the ID for authorized user over $\mathcal{D}_p$. $\mathcal{U}_a$ is the number of user attempts for which user is authorized to access $\mathcal{D}_p$. $\mathcal{E}_H$ and $\mathcal{D}_H$ are homomorphic encryption and decryption functions, respectively. For realizing secure communication between TTP and CSP, $\mathcal{E}_A$ and $\mathcal{D}_A$ are asymmetric encryption

functions, which ensure that only CSP can decrypt the information sent by TTP. $\alpha_{0\ldots n}$ are polynomial coefficients that are evaluated partially by third party. $\Delta_{y_{1\ldots n,n+1}}$ represent oblivious values that are calculated by CSP as a result of private matching protocol. ψ represents the offset value for a particular user. Ω is a constant operation performed with $\alpha_{0\ldots n,n+1}$ by TTP. $\uplus$ is a constant operation performed by cloud storage that ends up in $\mathcal{E}_v$ or $\mathcal{R}_v$ which is echo or residual value, respectively.

5 Proposed system

Following discussion is about the main idea, data outsourcing, user registration and initial setup and evaluation of user request at TTP and CSP.

We have used OUM in a system which is task oriented. In this system, multiple users are interested to access cloud data to enhance their respective business models. This data resource is shared through synchronization with valid users, having valid requests. As a business secret, the number of permissible attempts for each user is kept unknown throughout the communication and for flexibility the access model does not impose any time restriction. With each request initiated by the user, it travels through third party and reaches at CSP. Upon receiving user request, CSP performs a constant operation on it and ends up in a finite value. This value indicates either the user request falls within the subscription limit or not. In very first request made by a particular user, this value is noted by CSP against her ID. During user interaction, this value repeats itself only once if request falls beyond the subscription limit. Repetition of this value is adjusted during the setup process by owner and it cannot be predicted until user has availed all attempts. On this repeating value, further access is ceased by CSP. We call this as echo value $\mathcal{E}_v$ and all others in between them as Residual values $\mathcal{R}_v$. This mechanism is designed in such a way that CSP cannot predict any pattern from this key or residual values even for two users having same number of permissible attempts. To accomplish this model, we have also utilized services of a semi trusted third party (TTP).

We first briefly present the main idea for this oblivious data synchronization, then we describe the details of our methodology and setup activity.

5.1 Main idea

DataKon is a large research enterprise that deals with data related to weather, health, crime and social media. Statistical analysis on its data-hive provides interesting and useful patterns for hospitals, law enforcement agencies and few business organizations which are permanent stakeholders with DataKon. Voluminous growth of data under analysis and realization of Big-data [41] have made DataKon to utilize cloud services offered by Eve for both storage and computation. An expert team of analysts and computer programmers fully exploits the offered resources for deep insight to reveal the hidden knowledge and functional intelligence. The ability to analyze and forecast crime, health issues, drug consumption and social interaction has improved well due to DataKon services. Recently, few small/medium organizations have realized that utilizing these data can assist their business model more efficiently to enhance their product line. FutureLife is a famous insurance company in the town, which is inter-

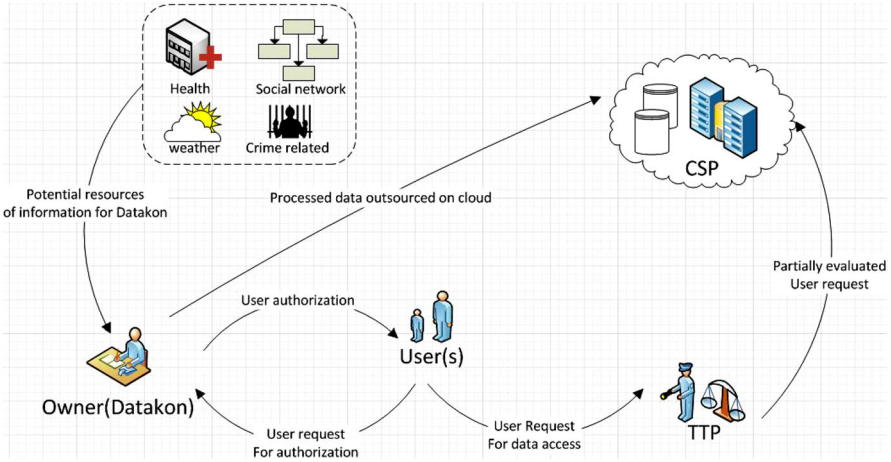


Fig. 1 Abstract model for oblivious user management (OUM) in a cloud storage

ested to access this service to assist its business policies. FutureLife also wants that value of its subscription limit should not be revealed to anyone until all attempts have been availed. Desire of this preference by FutureLife is to hide this fact from peer organizations that how frequent it may revise its business plan (Fig. 1).

FutureLife wants to consult DataKon service for n times and got the authorization for that. During the setup phase of OUM , DataKon shares certain parameters with TTP and CSP for its new customer i.e, FutureLife. OUM ensures that neither the TTP nor the CSP can deduce the valid attempts in advance. The request made by FutureLife routes through the TTP and reaches at CSP. Evaluation results by CSP distinguish between valid and invalid requests. For valid request, updated information is shared with FutureLife and this activity is logged at CSP for the billing purpose. DataKon will pay the CSP for this synchronization service on behalf of FutureLife only for valid logs (valid attempts only). After availing the permissible attempts if FutureLife forwards its request for $(n+1)$th attempt (beyond authorization), the evaluation result will end up in $\mathcal{E}_v$, which is an indication for CSP to cease user access from that point onward. Explanation on this indication through $\mathcal{E}_v$ is given in Sect. 5.1.4. Other indirect supportive factors to OUM are:

- hiding subscription limit is the user priority; therefore, access beyond authorized quota would automatically be discouraged from user side,
- from CSP perspective, only valid logs of valid attempts are subject to payment by DataKon; therefore, entertaining user request beyond subscription limit cannot be claimed.

The access request forwarded by user first goes through the third party where all the computation process take place homomorphically. The only information known at third party is the incremental value of offset which is ψ against ∂ . Further evaluation on these results by CSP tells that either request is valid or not. To make everything functional, entire process has been divided into four steps which are Data Outsourcing, Process

for User Authorization, User Request Evaluation and Provisioning of Authorized Data Through Synchronization.

5.1.1 Data outsourcing

Data outsourcing is a periodic activity by DataKon on Eve's cloud. Reason for these periodic updates is mainly due to the most updated analysis and with furnished user requirements. With each update, previous data are overwritten; hence at any instance of time, these data are considered as the latest version. For any authorized user, these data are represented as $\mathcal{D}_p$. The user request has to route through third party for which digital signature file of third party Œ is shared with the CSP in advance. Using digital signatures helps DataKon and CSP in assuring that user request has followed a legitimate path. Usage of cloud services and volume of data that travel towards authorized users are logged on Eve's cloud. As an evidence on utilization of cloud resources, this log file is used during the payment process between DataKon and Eve.

5.1.2 User registration and initial setup

A user with desired number of attempts $\mathcal{U}_a$ over $\mathcal{D}_p$ forwards her request to Datakon. After the mutual financial agreement in between Datakon and current user to access the $\mathcal{D}_p$ for $\mathcal{U}_a$ number of times, Datakon performs the following steps to complete the user registration process.

For unique identification, an identifier ∂ is given to user. Value of ∂ is also shared with third party and CSP. Next is the generation of random number which we will refer to as offset value ψ for user. This value ψ is shared only with the third party. The following step is the creation of polynomial $P(x)$ using ψ and $(\psi + \mathcal{U}_a)$. List of coefficients $\alpha_{0\ldots n}$ is forwarded to third party. With each user request, third party will perform a constant operation Ω on these coefficients. For a particular user, the operational sign $\pm$ on these coefficients $\alpha_{0\ldots n}$ is shared with CSP. To evaluate this polynomial $P(x)$ homomorhpcially Datakon shares σ_{pk} and σ_{sk} with third party and CSP, respectively. All communications that will take place between TTP and CSP are encrypted asymmetrically and for this purpose public K_{pub} and private keys K_{pri} are shared with TTP and CSP, respectively. With completion of registration process, user is now ready to request for $\mathcal{D}_p$.

5.1.3 Evaluation of user request at TTP

The request arrives at third party with user id (∂) where this access parameter remains the same throughout the entire lifetime of user access period. Against this (∂), an offset value ψ and public key for homomorphic operation σ_{pk} have already been communicated by Datakon in previous step. Third party performs a constant operation Ω by employing the homomorphic encryption over $\alpha_{0\ldots n}$. Results of this calculation $(\Upsilon_{1\ldots n})$ are then multiplied with a random number r and finally ∂ is added into this. These values are then sent to the CSP for further evaluation. The multiplication with random number r helps to hide a unique access pattern at CSP which is discussed in Sect. 8.2. This process is repeated whenever a request arrives at third party. Complete

set of values which are encrypted and then sent to CSP by TTP with each request is given in Eq. 10

$$\mathcal{E}_{\mathcal{A}}(\{\Upsilon_{1\ldots n}, \text{Œ}, \partial\}, K_{\text{pub}}) \tag{10}$$

5.1.4 Evaluation of user request at CSP

During the user registration process, CSP is shared with Œ, ∂, K_{pri} and σ_{sk}. After receiving values from third party, it is decrypted using the asymmetric key K_{pri}.

$$\mathcal{D}_A(\mathcal{E}_A((\Upsilon_{1\ldots n}, \text{Œ}, \partial), K_{\text{pub}})), K_{\text{pri}}) \tag{11}$$

Result of Eq. 11 ends up in $\{\Upsilon_{1\ldots n}, \text{Œ}, \partial\}$. CSP will use function of homomorphic decryption on $\Upsilon_{1\ldots n}$.

$$\mathcal{D}_H(\Upsilon_{1\ldots n}, \sigma_{\text{sk}}) \tag{12}$$

A constant operation of $\uplus$ starts with the output of Eq. 12. Its work flow is shown in Fig. 2. If user request comes for the first time, output value of 12 will be equal to ∂ which would be same as third parameter of Eq. 11. For first request, this value, i.e., ∂ is compared with already communicated values by data owner. If this value matches, user is marked active and provided with the synchronization service for data provisioning. Other than first request, value outcome from Eq. 12 will fall within $\sum_{i=2}^{n} \Delta_{yi}$ which is the residual value $\mathcal{R}_v$. After availing authorized attempts $\mathcal{U}_a$, if user forwards her request to CSP, the output of Eq. 12 will echo back with same value as earlier recorded

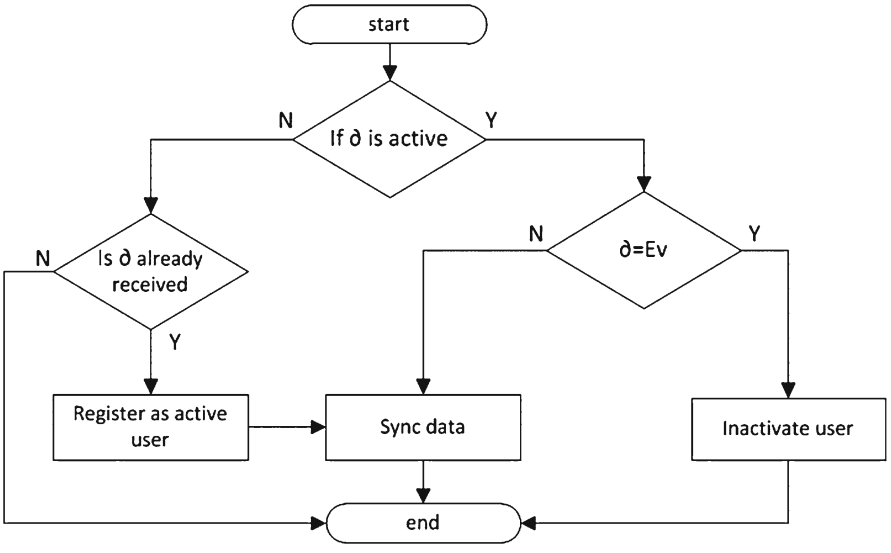


Fig. 2 Flow of user request evaluation at CSPlabelfig

as echo value, $\mathcal{E}_v$. With this observation, CSP will come to know that user has already availed desired number of attempts $\mathcal{U}_a$, and her current and subsequent requests are to be ceased. The purpose of using digital signature Œ of third party is to let CSP know that request has been forwarded through designated trusted party. The log activity which is stored at CSP will assist for transparent pecuniary matters with Datakon. Mandatory values stored in this log for each users are ∂, Œ and $\Delta_{y_{1\ldots n}}$.

6 Implementation

For the viability of OUM and its efficacy, the idea has been implemented and tested on Google cloud [42]. The process of user registration and responsibility of third party is tested on local machine. Further processing on user request via third party is then transported on Google cloud using the Google app engine SDK [42] and deploying a Java Web Service. A user request for specific number of attempts is initialized using local machine and all parameters are shared as discussed in Sect. 5.1.2. User request is then initiated from local machine to third party where it is evaluated and its computational results are then send to Google cloud. The web service running there interacts with incoming parameters and evaluates the results. The application has been tested and verified for complete range of values at CSP which are $\Delta_{y_{1\ldots n,n+1}}$. The graph pattern on these values which is discussed in Sect. 8.2 found unique and unpredictable during the entire life-cycle of user access period. All operations that are performed homomorphically are achieved using the Pascal Paillier cryptosystem. We have used standards Java SE 7.0 for developing the OUM.

7 Evaluation and results

Oblivious user management is tested on local machine, Google App Engine [42], and Android platform. Process of user registration and subsequent evaluation on behalf of TTP are executed on local machine. Initial setup and third party evaluation time are shown in Fig. 3. Role of CSP and relevant execution on user request are done on Google App engine. While evaluating user request on App engine, we have selected the F4 front end instance. This instance has 2,400 MHz processing power and 512 MB

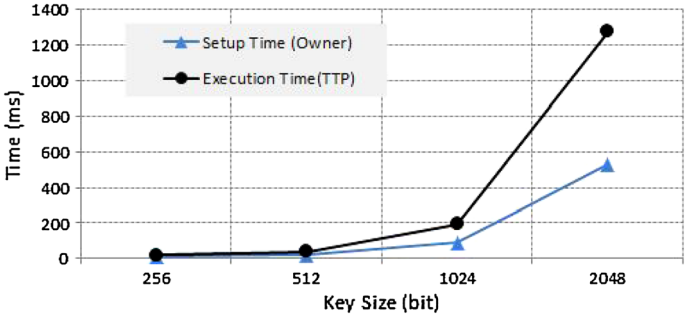


Fig. 3 Time required for initial setup and trusted third party for request evaluation

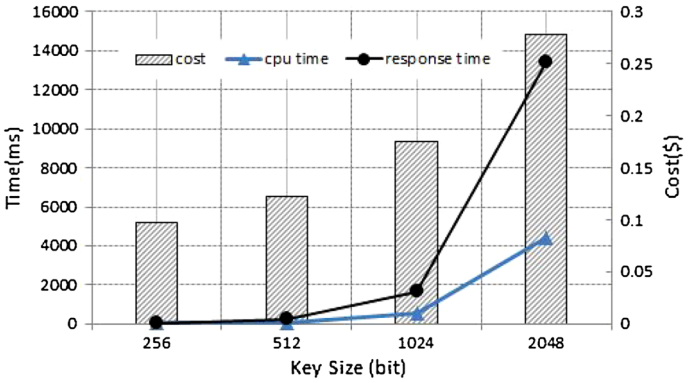


Fig. 4 Time, cpu and cost estimation on cloud while evaluating user request

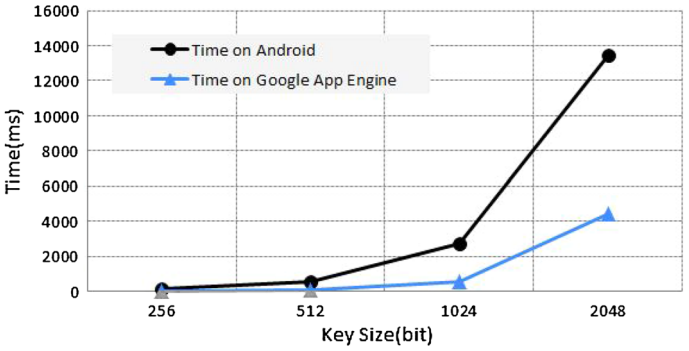


Fig. 5 Time comparison for user request evaluation on cloud and Android

of RAM. Figure 4 shows the cpu time, response time and cost analysis of user request on F4 instance. Initial setup by owner and execution steps by third party are tested on a local machine with Intel(R) Core(TM) i3 processor and 4 GB of RAM. Microsoft(R) Windows7(TM) X 64 bit is the OS installed on it. Other than Google App engine, OUM has been evaluated on Android platform too. Comparison of execution time for request evaluation at App engine and Android is shown in Fig. 5. For this purpose, Nexus emulator with 512 RAM and VM heap of 32 MB is selected using Android SDK [43].

8 Discussion

In this section, we will discuss salient features of OUM, pattern evaluation for Residual values at CSP followed by system limitations.

8.1 Salient features and efficiency

The main concept of system design is twofold. First, it authorizes users to avail cloud resources in a more flexible way without imposing any time restriction. Secondly,

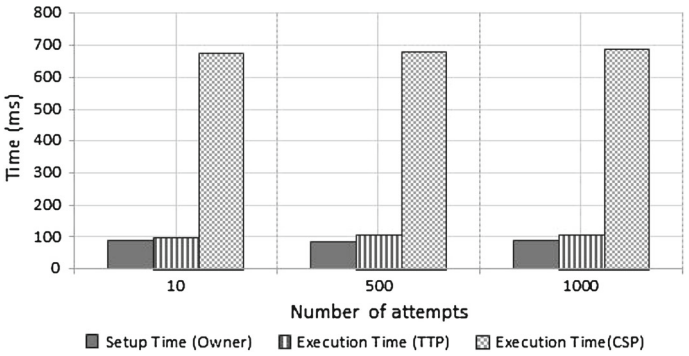


Fig. 6 Execution time for different number of attempts

the efficacy of hiding user attempts has been achieved using the cryptographic primitives, i.e, homomorphic encryption and private matching. Minimum complexity of our design has made it light weight at all frontiers which are setup activity at owner end and partial execution at TTP and CSP. The worth mentioning contribution of our idea is its negligible reliance on number of attempts. The computational activity for three different users having 10, 500 and 1,000 attempts, respectively, consumes almost similar amount of time as shown in Fig. 6. Execution time that is consumed at TTP and CSP for different users reveals no clue about which user has more subscription limit over the others. It is the constant operation that is performed each time at TTP and CSP no matter it ends up in $\mathcal{K}_v$ or $\mathcal{R}_v$.

For multiple attempts, a particular user can be given with various access keys where each key is assumed to be used only once. Using this approach might be efficient where user has few number of attempts. Managing these unique keys for each attempt will be the responsibility of authorized user which is an additional burden, especially when number of keys is very high. To overcome this, we used polynomial with multiple roots; for more detailed discussion refer to Sect. 3.2 on defining polynomial with multiple roots. Instead of creating a polynomial whose degree and complexity grow with number of user attempts $\mathcal{U}_a$, we devised a more efficient approach that is free from this limitation. Using only $\mathcal{U}_a$ and its offset value ψ kept degree of polynomial same for all users. For this purpose, we used ψ and $(\psi + \mathcal{U}_a)$ while defining the polynomial as given in Eq. 9.

In initial version of our work, this $\mathcal{R}_v$ has been found helpful to predict the number of attempts by CSP. This weakness of prediction is removed and discussed in next section.

8.2 Pattern evaluation of residual values

For every request forwarded by a user, CSP performs a constant operation $\uplus$. Outcome of this operation helps CSP to perform various activities like user registration, data provisioning, and freezing user account. After implementing the proposed methodology, these values are tested to ensure presence of any pattern that might be helpful in

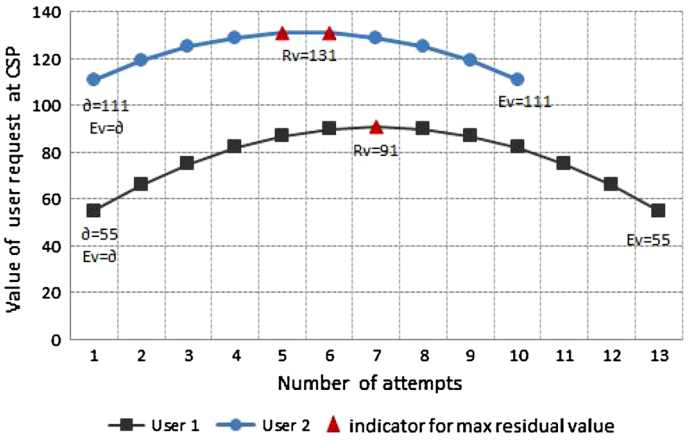


Fig. 7 Pattern of Residual values—without randomization

predicting the user attempts at any stage before it is consumed in totality. To explain it, a user who is authorized for $\mathcal{N}$ number of attempts, the value of $\mathcal{R}_v$ appeared with following observations:

$$\sum_{i=1}^{N/2} Rv_{i+1} > Rv_i \tag{13}$$

This pattern reflects itself back in reverse value when half of the attempts have been achieved which is

$$\sum_{i=(N/2)+1}^{N} Rv_{i+1} < Rv_i \tag{14}$$

with this pattern, CSP will remain unaware until $(\mathcal{N}/2)$ attempts, however; after that, it can predict the remaining attempts easily. At this stage, the role of TTP will become ineffective as remaining attempts are now obvious to CSP. The pattern of residual values given in Eqs. 13 and 14 is shown in Fig. 7. In this figure, two users with even and odd number of attempts have been tested to find out these identifiable patterns. In both cases, the output value at CSP can be seen as normal distribution. After half of this distribution is availed, the remaining values are just in reverse order. To overcome this drawback, OUM design is tweaked by incorporating randomization by adding a random value in Ω. This randomization is applied in Ω such that $r_i \times \Upsilon_{1\ldots n}$, where r is a random number. The additional computational cost is again a constant operation and hence never affected the overall performance. Now all values $\sum_{i=1}^{N} \mathcal{R}_{vi}$ that end up at CSP do not follow any specific pattern or normal distribution. For verification, two users having similar number of attempts have been tested. Now, the residual values calculated by CSP are logged and shown in Fig. 8 which gives no clue to predict remaining number of valid attempts. With this tweak, also the role of TTP remains effective for the entire lifetime of user validity on cloud data.

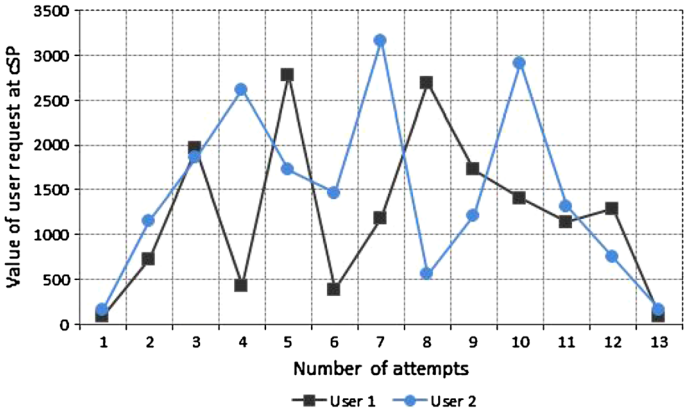


Fig. 8 Pattern of Residual values—with randomization

It has always been a great deal to balance both security and performance [50]. Encryption and oblivious data processing are aimed to achieve desired level of security, however, at the cost of performance. To ensure performance in such scenario, distributing the participating algorithms of oblivious processing by practicing high-performance computing (HPC) can be considered to boost the performance. For applications where response time is a critical factor, HPC can play its significant role. The proposed design of OUM is independent of underlying infrastructure and it can be implemented for cloud and HPC with required modifications. In its present design, OUM relies on the services of a single cloud server and primarily aims to assure high privacy; however, for data intensive application the synchronization services can be engaged with more than one instances of cloud server for optimal performance. For secure and fast data sharing, reader may refer to [45].

8.3 Limitations

The scheme proposed in OUM is best suitable where time anticipation is non-deterministic to access the cloud data. In the current approach, the data which become inaccessible after permissible number of attempts can still be shared by CSP to that user if user still holds the effective decryption keys. Our model works under the assumption that CSP will not collude with malicious user. This breach can be effective as long as data are not encrypted again by the owner. To meet this limitation, the idea of proxy re-encryption can be incorporated into the existing system through third party or CSP.

9 Conclusion and future work

Cloud services that are related with sensitive data usually raise concerns for security and privacy by owner as well as its consumer in terms of storage and subsequent usage.

Instead of accepting the underlying coarse premise of trust, applications of specific and fine-grained solutions are required to harness the formidable challenges of security and privacy. Realization on this has been addressed using our proposed model of OUM to resolve the secure usage of sensitive data, especially from the usage perspective. With OUM, we provide task-oriented data access model for users and at the same time it facilitates owner with mechanism of auto revocation of users and that too with minimal managerial burden. The novelty of time-independent approach adopted in OUM takes an advantage where time-dependent models are hard to implement, thus giving another option to meet the user requirements. With evaluation and experimental results, we highlighted the design efficiency of OUM in terms of time and user privacy. Considering legitimate access limit of an authorized user over sensitive data as her business secret, we have achieved to preserve it throughout system execution. The design and evaluation mechanism of user requests through OUM have been made independent from number of authorized attempts. This key feature makes the process of user registration and request evaluation light weight and oblivious for both data owner and cloud server, respectively. The evaluation results appearing as residual value give no additional information to cloud server to discover the echo value prior the user has availed her authorized limit.

While considering the element of flexibility for user access over encrypted data, the proposed model of OUM focuses on a task-oriented approach. Under existing model of OUM, a user once authorized will remain active until she avails all permissible attempts. Although this access model is flexible enough but this indeterministic access period needs to be bounded with certain time threshold, thus establishing the need of a hybrid solution. This hybrid approach is envisioned to adopt its underlying model on authorized limit as well as certain time bounds. In our future work, we will combine both approaches of task and time in building a more comprehensive system.

Acknowledgments This research was supported by a grant from the Kyung Hee University in 2013[KHU-20130439].

Appendix: Performance evaluation: Data tables

Performance evaluation presented in Sects. 6 and 7 is based on the following data tables. Figure 3 presented the visual representation of Table 2. Similarly, Figs. 4, 5, 6, 7, 8 are represented by Tables 3, 4, 5, 6 and 7 respectively.

Table 2 Time required for initial setup by owner and request evaluation at TTP with variable key length

Key size	Setup time-ms (owner)	Execution time-ms (TTP)
256	9.5	8.3
512	20.06	18.33
1,024	89.66	103.13
2,048	528.7	749.3

Table 3 Cost analysis and execution time at F4 instance of Google cloud

Key size	CPU time (ms)	Response time (ms)	Cost ($)
256	27	42	0.097
512	75	160	0.123
1,024	551	1,120	0.175
2,048	4,438	9,013	0.278

Table 4 Execution time on android and Google app engine

Key size	Android time (ms)	Google app engine time (ms)
256	151	27
512	585	75
1,024	2,703	551
2,048	13,487	4,438

Table 5 Execution time for different numbers of attempts(10,500,1000) where key size is 1024 for all

Attempts	Setup time (ms) by owner	Execution time (ms) by TTP	Execution time (ms) by Google app engine
10	88	99	674
500	87	105	678
1,000	89	106	690

Table 6 Values of users request after evaluation at CSP, i.e., $\Delta_{y_{1\ldots n}}$ (without randomization)

User attempt	User 1 (9 attempts)	User 2 (12 attempts)
1	111	55
2	119	66
3	125	75
4	129	82
5	131	87
6	131	90
7	129	91
8	125	90
9	119	87
10	111	82
11	–	75
12	–	66
13	–	55

Table 7 Values of users request after evaluation at CSP, i.e., $\Delta_{y_{1...n}}$ (with randomization)

User attempt	User 1 (12 attempts)	User 2 (12 attempts)
1	90	166
2	726	1,155
3	1,960	1,860
4	432	2,619
5	2,784	1,728
6	385	1,470
7	1,188	3,168
8	2,695	560
9	1,728	1,216
10	1,404	2,916
11	1,140	1,320
12	1,287	748
13	90	166

References

1. Mell P, Grance T (2011) The nist definition of cloud computing (draft). NIST Spec Publ 800(145):7
2. Motahari-Nezhad HR, Stephenson B, Singhal S (2009) Outsourcing business to cloud computing services: Opportunities and challenges. In: IEEE Internet Computing, Palo Alto, 10
3. 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. Future Gener Comput Syst 25(6):599–616
4. Armbrust M, Fox A, Griffith R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al (2010) A view of cloud computing. Commun ACM 53(4):50–58
5. Giles J (2012) Big data: lessons from the leaders. economist intelligence unit limited
6. Leavitt N (2009) Is cloud computing really ready for prime time? Computer 42(1):15–20
7. Dikaiakos MD, Katsaros D, Mehra P, Pallis G, Vakali A (2009) Cloud computing: Distributed internet computing for it and scientific research. IEEE Internet Comput 13(5):10–13
8. Catteddu D (2010) Cloud Computing: benefits, risks and recommendations for information security. Springer
9. Gammage B, Plummer D, Valdes R, McGee K, Potter K, Tan S, Dave A, Richard H, Jay H, Brian P et al (2011) Gartners top predictions for it organizations and usersand beyond: Its growing transparency. Document ID G00208367:2010
10. Weller M (2010) Big and little oer. In: 2010 Proceedings. Barcelona. http://hdl.handle.net/10609/4851
11. Jacques B, Corb L, Manyika J, Nottebohm O, Chui M (2011) Borja de Muller Barbat, and Remi Said. Search, The impact of internet technologies
12. Dijcks J-P (2013) Oracle:big data for the enterprise. http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=609&p_org_id=15&lang=KO&get_params=dc:D75058GC10,p_preview:N
13. (2013) dunnhumby. Dunnhumby:customer science company. http://www.dunnhumby.com/
14. Kaplan AM, Haenlein M (2010) Users of the world, unite! the challenges and opportunities of social media. Bus Horiz 53(1):59–68
15. University of California (2013) Uci machine learning repository. http://archive.ics.uci.edu/ml/datasets.html
16. The world bank (2013) The world bank data catalog. http://datacatalog.worldbank.org/
17. Mao W (2001) Modern cryptography. In: Selected Areas in Cryptography VIII (SAC'01. Citeseer
18. Ateniese G, Kevin F (2006) Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans Inf Syst Secur (TISSEC) 9(1):1–30
19. Simmons G, Armstrong GA, Durkin MG (2011) An exploration of small business website optimization: enablers, influencers and an assessment approach. Int Small Bus J 29(5):534–561

20. Bayardo RJ, Agrawal R (2005) Data privacy through optimal k-anonymization. In: Proceedings 21st International Conference on Data Engineering, 2005. ICDE 2005, pp 217–228
21. Chow R, Golle P, Jakobsson M, Shi E, Staddon J, Masuoka R, Molina J (2009) Controlling data in the cloud: outsourcing computation without outsourcing control. In: ACM Proceedings of the 2009 ACM workshop on Cloud computing security, pp 85–90
22. Kamara S, Lauter K (2010) Cryptographic cloud storage. In: Financial Cryptography and Data Security. Springer, pp 136–149
23. Coull S, Green M, Hohenberger S (2009) Controlling access to an oblivious database using stateful anonymous credentials. In: Public Key Cryptography-PKC 2009. Springer, pp 501–520
24. Camenisch J, Dubovitskaya M, Neven G, Zaverucha GM (2011) Oblivious transfer with hidden access control policies. In: Public Key Cryptography-PKC 2011. Springer, pp 192–209
25. Frikken K, Atallah M, Li J (2006) Attribute-based access control with hidden policies and hidden credentials. IEEE Trans Comput 55(10):1259–1270
26. Waters B (2011) Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Public Key Cryptography-PKC 2011. Springer, pp 53–70
27. 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, ACM, pp 89–98
28. Sahai A, Waters B (2005) Fuzzy identity-based encryption. In: Advances in Cryptology-EUROCRYPT 2005. Springer, pp 457–473
29. Blaze M, Bleumer G, Strauss M (1998) Divertible protocols and atomic proxy cryptography. In: Advances in Cryptology EUROCRYPT'98. Springer, pp 127–144
30. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In: IEEE, INFOCOM, 2010 Proceedings IEEE, pp 1–9
31. Liu Q, Wang G, Wu J (2014) Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. In: Information Sciences, 2014, vol 258. Elsevier, pp 355–370
32. Bethencourt J,Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, IEEE, 2007. SP'07, pp 321–334
33. Müller S, Katzenbeisser S, Eckert C (2009) Distributed attribute-based encryption. In: Information Security and Cryptology-ICISC 2008. Springer, pp 20–36
34. Wang G, Liu Q, Wu J (2010) Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In: Proceedings of the 17th ACM conference on Computer and communications security, ACM, pp 735–737
35. Wang G, Liu Q, Guo M (2011) Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. Comput Secur 30(5):320–331
36. Patel B, Crowcroft J (1997) Ticket based service access for the mobile user. In: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking, ACM, pp 223–233
37. Freedman MJ, Nissim K, Pinkas B (2004) Efficient private matching and set intersection. In: Advances in Cryptology-EUROCRYPT 2004. Springer, pp 1–19
38. 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-Verlag, London, pp 573–584
39. 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-Verlag, Berlin, pp 223–238
40. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of the 29th conference on Information communications, INFOCOM'10. IEEE Press, Piscataway, pp 534–542
41. James M, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Byers AH (2011)The next frontier for innovation, competition, and productivity, Big data
42. Google (2013) Google app engine. https://cloud.google.com/products/app-engine
43. The Android open source project (2013) Netbeans android plugin. http://plugins.netbeans.org/plugin/19545
44. Pervez Z, Ahmad A, Masood A, Lee S (2013) Privacy-aware searching with oblivious term matching for cloud storage. Supercomputing 63(2):538–560
45. Allcock B, Bester J, Bresnahan J, Chervenak AL, Kesselman C, Meder S, Nefedova V, Quesnel D, Tuecke S, Foster I (2001) Secure, efficient data transport and replica management for high-performance

data-intensive computing. In: Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, 2001, IEEE, MSS'01, pp 13–13
46. Lewko A, Okamoto T, Sahai A, Takashima K, Waters B (2010) Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Advances in Cryptology-EUROCRYPT 2010. Springer, pp 62–91
47. Chase M (2007) Multi-authority attribute based encryption. In: Theory of Cryptography. Springer, pp 515–534
48. Li J, Huang Q, Chen X, Chow SS, Wong DS, Xie D (2011) Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, pp 386–390
49. Goh E-J, Shacham H, Modadugu N, Boneh D (2003) Sirius: Securing remote untrusted storage. NDSS 3:131–145
50. Venkatesh VP, Sugavanan V (2009) High performance grid computing and security through load balancing. In: IEEE, International Conference on Computer Engineering and Technology, 2009. ICCET'09, vol 1, pp 68–72