Dynamic Content-based Cloud Data Integration System with Privacy and Cost Concern

Yuan Tian
Kyung Hee University
Republic of Korea
Yongin-si, Gyeonggi-do
ytian@khu.ac.kr

Biao Song
Kyung Hee University
Republic of Korea
Yongin-si, Gyeonggi-do
bsong@khu.ac.kr

Eui-Nam Huh
Kyung Hee University
Republic of Korea
Yongin-si, Gyeonggi-do
johnhuh@khu.ac.kr

ABSTRACT
Although a lot of research is currently taking place in the cloud computing technology itself, there is an equally urgent need for understanding the cost effective issues surrounding cloud. In this paper we present a dynamic content-based inter-cloud data integration system with privacy and cost concern. Users can evaluate the privacy risk value based on the content (input data or intermediate searching results) which are generated during run-time, in the meanwhile, comparing with existing data sharing techniques, our method is more practical as the cost for technical supporting privacy must be considered in the commoditized cloud computing environment.

1. INTRODUCTION
Anyone who has interest in information technology will discover that it is impossible to avoid coming across the term "cloud computing". As one of the 2010 top 10 strategic technologies, cloud computing dynamically provides high-quality cloud-based services and applications over the internet.

Generally, cloud computing covers two major trends in information technology. One is the efficiency of IT, whereby the power of could computing can be utilized more efficiently through highly scalable hardware and software resources. The other one is the economic of IT, whereby cloud computing is used as a competitive tool through rapid deployment, parallel batch processing, use of compute-intensive business analytics and mobile interactive applications that respond in real time to user requirements [2]. Similar to real world utilities, nearly all the services provided in cloud computing are billed on a utility computing basis. Based on usage and quality of service expectations, consumers just need to pay for those services like water, electricity, gas and telephony [1], rather than investing heavily to maintain their own computing infrastructure.

Cloud computing fundamentally shifts the traditional "desktop as a platform" principles theories to "internet as a platform", in the meanwhile, more problems arise than ever before [7].

For example, the disruptions of cloud computing services, insecure data access, and how to build cost-effective cloud computing. Yet cloud computing is also associated with a range of severe and complex privacy issues, which should be particularly considered, such as risks relating to malicious insiders as well as the nefarious use of cloud computing [3].

In order to meet the growing requirement for privacy management, many related technologies with different degrees of privacy control have been proposed. In [4], Yau et al provided a comprehensive privacy preserving repository for data integration and sharing. The main contribution of their work is that the data processing is kept securely in both data sharing services and the repository, but they did not consider the cost for hashing and sending all data to the repository, which could be the critical defect and hinders the implementation of their method to the real-world. Compare with the method mentioned in [4] which fully supports the privacy protection, Ye et al [5], proposed a user-side privacy protection model for search users, which is able to prevent part privacy breaches according to the number of noise injections. However with the increasing number of noise queries, the consumption of their system also increases dramatically.

While a lot of research is currently taking place in the technology itself, there is an equally urgent need for understanding the cost effective issues surrounding cloud computing [8, 9]. Thus we proposed a cost-based inter-cloud data integration privacy-aware system in our previous work [6]. In general, it is up to customers to decide a strategy about how to get a service fulfilled on the basis of their personal feeling of the importance of their data, and the appropriate cost for processing data. However, the previous approach only supports static searching, which means, once users declare their privacy preferences, they are not able to modify the rules they defined previously during the searching process. In this paper, we improve our system by using a content-based searching. If any sensitive information is found in the intermediate searching results, our system can dynamically give a new allocation strategy based on the modification. Our proposed system has the following major benefits:

1. The repository cloud can only collect data from data sharing services and integrate the data to satisfy users' integration requirements. Except the information needed to be revealed for

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CEAS '11, September 1–2, 2011, Perth, Western Australia, Australia.
Copyright 2011 ACM 978-1-4503-0788-8/11/09…$10.00.

data integration, the repository cloud will not have extra information about the data and cannot use it for other purposes.

2. We adopted a cost evaluation model to help users find an optimal solution, which ensures maximal privacy protection with minimal costs. Our proposed system is a huge improvement on cost-effectiveness in cloud computing compared with other privacy protection technologies.

3. Based on the pre-defined privacy policy, our system can dynamically adjust the allocation strategy if the intermediate results contain any sensitive data.

The structure of the rest of the paper is as follows. Section 2 introduces a scenario that is used as a running example throughout the paper. A component-based view of our system is presented in Section 3 and the design approach is given in Section 4, and the last section of this paper presents our conclusions.

2. A MOTIVATING EXAMPLE

This section presents a scenario used throughout the paper and we use this motivating example to show how our system works. The scenario is a revised version of the case study proposed in [4, 10].

In a healthcare system there are multiple collaborated clouds which participate in processing, sharing and integrating data. We assume for the purpose of obtaining the menu which is proposed to ulcer suffers, the clouds may contains the data from medical research institutes, hospitals and pharmacies. For simplicity, only four databases are considered: a disease record database *T1(Disease, Patient)* which stores patient's names and corresponding diseases which they are suffering, an identification information database *T2(ID, Patient)* which stores patients' names and their IDs, a hospital database *T3(ID, Drug, Menu)* storing hospitalization information and a pharmacy database *T4(Disease, Drug)* which stores popular drugs for each disease.

Table 1. The databases which used in the motivating example

Disease	Patient
Tuberculosis	Bree
Aids	Bob
Ulcer	Ada
Diabetes	Alice

(a)

ID	Patient
10001	Alice
10002	Tom
10003	Susan
10004	Bree

(b)

ID	Drug	Menu
10001	D1	M3
10003	D2	M3
10004	D3	M2
10005	D1	M1

(c)

Disease	Drug
Ulcer	D4
Cancer	D2
Flu	D1
Diabetes	D3

(d)

(a) Disease records T1 (b)Identification Information T2 (c)Hospital T3 (d)Pharmacy T4

Now we are going to express our motivating example by the following four SQL queries which shown in Table 2.

Table 2. The integrated query

Q1 -> Tmp1 Select T1.Patient From T1 where T1.Disease="Ulcer"	Q2 -> Tmp2 Select T2.ID From Tmp1,T2 where Tmp1.Patient=T2.Patient
Q3 -> Tmp3 Select T4.Drug From T4 where T4.Disease="Ulcer"	Q4 Select T3.Menu From Tmp2, Tmp3 and T3 where T3.ID=Tmp2.ID and T3.Drug=Tmp3.Drug

Query Q1, Q2 and Q3 generates three temporary tables Tmp1, Tmp2 and Tmp3 respectively, and the final results are from the last query Q4.

As some queries may need other queries' results as inputs, the repository randomizes those results by using the Hush function, which avoids the need for the repository to know that results but still keeps the mapping relation between data. For example, we replace the Q1's results {Bob, Alice} by {H(Bob), H(Alice)} which protects Q1's results without disturbing Q2, as the hashed name usually remains unique, the repository can easily evaluate Q2 by comparing H(Tmp1. Patient) and H(T2.Patient). Since H(Susan) is not in the Q1's hashed result{H(Bob), H(Alice)}, the repository can find the patient Susan whose ID is 10004 is not an Aids nor diabetes patient.

3. THE PROPOSED SYSTEM

Our proposed system aims to mediate between users' privacy preferences and the cost for privacy protection. The overall architecture of our system is illustrated in Fig.1.

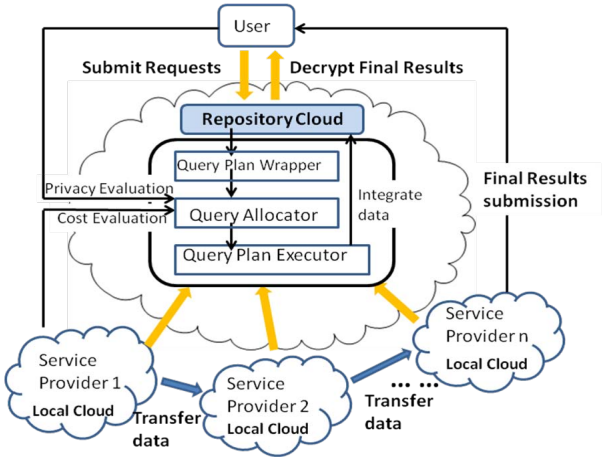

Fig.1 System Architecture

The user sends his integration requirements to the repository cloud and only the required data for users' integration request is collected. The query plan wrapper in the repository cloud will correctly construct a query plan for users' integrated query, decompose the query into a set of sub-queries, and discover corresponding service providers. The data which was conducted or collected in the query plan wrapper are then sent to the query allocator.

Each sub-query could be executed in the local cloud or the repository cloud. For the former case, the sub-query will be executed in the service provider which stores the corresponding database. Thus, only the results for the sub-query are returned to the repository cloud or transferred to the successive service provider. The price for processing data in the local cloud is relatively cheaper, however, it may not be secure as both the data

storing and processing are done in the same service provider. While the latter one is to fetch data in the local cloud, randomize all of those data and then send to the query plan executor for further processing. Executing queries in the query plan executor guarantees secure protection as the data is stored and processed separately. Therefore, the cost for processing data in repository cloud is high as it requires randomizing all the data in the local service provider and transfers it back to the repository cloud.

In order to help users find a balance between privacy insurance and the cost for privacy protection, the query plan executor decides where the sub-query should be executed. Before sending those sub-queries to the query plan executor, three separating phases are proposed:

- ✓ Privacy evaluation: based on user's query, allows users to express their privacy preferences by setting risk values to decide the importance of data and the relationship between those data.
- ✓ Cost evaluation: here, the pricing mechanism is that the service provider in repository cloud should estimate a price for processing data, whereas other providers which are located outside the repository cloud provide the costs for both processing data and uploading data to the repository cloud.
- ✓ Query Allocation: the query allocator chooses an optimal strategy according to user's preferences in the above two steps, and sends it to the query plan executor to enforce that method. At last, the requested data is sent back to the user from the query plan executor.

4. SYSTEM DESIGN

In this section, we presented the process of our system implementation. Before discussing the functionality of our system, some basic definitions are introduced in order to establish a common ground of concepts.

Firstly, the query plan wrapper converts user's integrated query to a query plan graph *G*. For a integrated query, the query plan graph *G={V, E, C}* is a labeled directed acyclic graph. Among which, $V=\{v_1, v_2, ..., v_m,\}$ is a set of nodes where each v_i represents an intermediate search result; $E=(e_1,e_2,...,e_l)$ is a set of edges where each edge $e_{ij}=(v_i,v_j)$ represents a data integration relation between v_i an v_j. $C=(c_1,c_2,...,c_l)$ is a set of labels attached to each $e_{ij}\in E$ and each label $c_{i,j}$*=(op, attr1, attr2)* $\in C$ specifies that the data of v_i and v_j is integrated by the data integration operator *op* between v_i's attribute attr1 and v_j's attribute attr2. Generally, the operator *op* can be any binary comparison operator chosen from {=, ≠, >, <} or any aggregate operator chosen from {*SUM, AVG, MAX, MIN*}.

The motivating example in Section 2 can be represented by the query plan graph shown in Fig. 2.

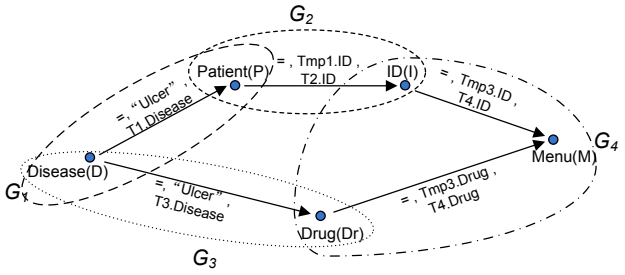

Fig.2 The query plan graph from the motivating example

The query plan graph is decomposed to several sub-graphs $\{G_1,G_2,G_3,...,G_n\}$ where each sub-graph represents a sub query in Table 2. The overlapping of sub-graphs shows that the immediate result from a sub-query is used as the input for another sub-query.

We use a matrix D_{m*m} to represent the disclosure condition of the searching results and the relationships between them. For each $d_{ii}\in D$, $d_{ii}=0$ denotes the information of v_i is not disclosed, while $d_{ii}=1$ denotes the information of v_i can be known by a service provider. Similarly, $d_{ij}=0$ denotes the relationship between v_i and v_j is not disclosed while $d_{ij}=1$ shows the relationship between v_i and v_j that can be known by service provider. An instance of data disclosure condition matrix is shown in Fig.3.

	D	P	I	Dr	M
D	1	1	0	0	0
P	0	1	0	0	0
I	0	0	0	0	0
Dr	0	0	0	0	0
M	0	0	0	0	0

Fig.3 An instance of a data disclosure condition matrix

As the service providers which locate in different clouds may collude each other in order to retrieve more information that they are not supposed to get, i.e., if the relationship between disease and patient is disclosed to one service provider and the relationship between a patient and their ID is disclosed to another service provider, then it is possible for them to find out the relationship between disease and ID. Thus, we define $d_{ij}=1$ if there exists a path P between v_i and v_j where the starting node of *P* is v_i, the ending node of P is v_j, and every edge in P has been disclosed.

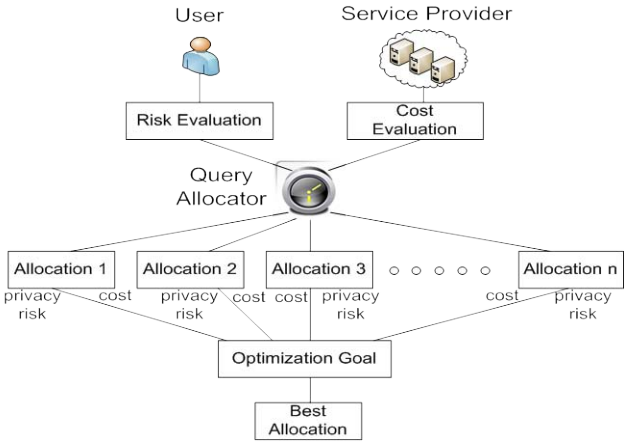

Fig.4 Process of the proposed system

As we mentioned before, each sub-query can be executed in local cloud or repository cloud, for the sub-graph G_k, we set $A(G_i)=1$ to denote the former case and $A(G_i)=0$ to denote the latter one where $A(G_i)$ is the sub-allocation decision of sub-graph G_i. As each sub-graph contains nodes and edges, if the value of the sub-graph is set as "1", the values of all the nodes and edges within the sub-graph should be also set as "1". For the nodes which may appear in several sub-graphs with the different setting values "0" and "1", we should consider this node (representing intermediate results) is executed in local cloud and set it's value as "1" as it may be disclosed in the local cloud.

The process of our proposed approach is briefly presented in Fig.4. Based on the user's query, the user and service providers first submit their evaluation for risk and cost evaluation respectively. After that, the query allocator decides an optimal allocation which provides minimum trade-off value between user's risk and service cost.

4.1 Risk Evaluation

Before submitting the user's sub-requests to the query plan executor, the user could express his privacy preference by giving a risk value to those intermediate nodes. Each node represents the results from the previous query and the values on the nodes represent to what extent that user cares about those data or the relationships between the data.

	D	P	I	Dr	M
D	4	9	9	1	1
P	–	5	∞	–	5
I	–	–	3	–	1
Dr	–	–	–	3	1
M	–	–	–	–	0

Fig.5. An instance of a penalty matrix

The risk values for the intermediate nodes are presented in a risk matrix R_{m*m} in Fig.5. The $r_{ii} \in R$ denotes the risk value on intermediate search result v_i and $r_{i,j} \in R$ denotes the risk value on the relationship between the intermediate search result v_i and v_j. Every value in the matrix is an integer ranging from 0 to 9, or +∞.

For example, from Fig.5 we can see that the user gives the "Menu" information away and set the risk value to "0" as he may think this information is not that important. Whereas he cares which patients are suffering from which diseases as this kind of data disclosure may offend a patient's privacy, so the risk value on the relationship between "Patient" and "Disease" is set to infinity.

	D	P	I	Dr	M
D	1	1	0	0	0
P	0	1	0	0	0
I	0	0	0	0	0
Dr	0	0	0	0	0
M	0	0	0	0	0

*

	D	P	I	Dr	M
D	4	9	9	1	1
P	–	5	∞	–	5
I	–	–	3	–	1
Dr	–	–	–	3	1
M	–	–	–	–	0

= 18

Fig.6 Risk Penalty Value

Given the disclosure condition matrix D and risk matrix R, the risk penalty value RPV can be calculated by the formula RPV= $\sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \times r_{ij}$. An instance for showing how to calculate the risk penalty value is presented in Fig. 6.

4.2 Cost Evaluation

At the same time, the service providers should fix the service price concerning different situations. Our pricing mechanism includes four kinds of costs:

1) Data randomization cost

As we mentioned before, the sub-query could be executed in the repository cloud in order to assure the data can be stored and processed separately. Thus, all the data in the local service provider needs to be randomized before transmitting to the repository cloud. The service provider evaluates this price according to the amount of data.

2) Cost for uploading all data

The user who wants to process the data in the repository cloud needs to upload all of the randomized data. This cost is for submitting those randomized data.

3) Query execution cost

The cost is for executing a sub-query in the local cloud or repository cloud. The query execution cost varies based on the amount of data.

4) Cost for uploading intermediate result

This cost is for transmitting the results of the sub-query after executing it locally. According to the user's preference, the results can be transferred to the next local server provider or to the repository cloud for further processing.

Fig.7 shows an example of our pricing model. We use notation $C_r(G_i)$ to denote the cost for processing a sub-query G_i in the repository cloud while $C_l(G_i)$ represents the cost for processing a sub-query G_i in the local cloud. If the user decides to execute a sub-query in repository cloud, the service provider should firstly randomize the whole data and send it to repository cloud to execute the sub-query. So $C_r(G_i)$ includes the cost for randomization data, uploading all data, and executing a query.

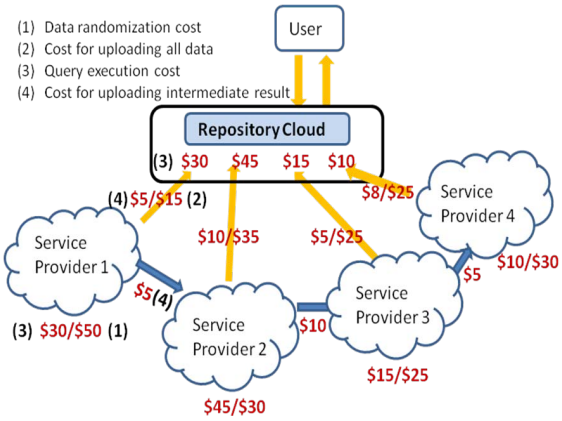

Fig.7 Pricing Mechanism

For example, from Fig.7 we can calculate that $C_r(G_1) = \$50 + \$15 + \$30 = \$95$, which means the total cost for executing G_1 in the repository cloud is \$95. On the other hand, if G_i is executed in the local cloud, only the results of that sub-query will be submitted to the repository cloud or transferred to the successive service provider. So $C_l(G_i)$ includes the cost for executing the query and uploading intermediate results. We can deserve $C_l(G_1) = \$30 + \$5 = \$35$ in the example in Fig.7, so the total cost for executing G_1 in service provider 1 is \$35.

4.3 Query Allocation

4.3.1 Static Query Allocation

A static query allocation can be decided using the following method. Based on the above analysis, the query allocator provides several strategies for the user considering both price and privacy. User could select a service with the lowest cost allocation which subjects to privacy risk constraint, or a service with lowest privacy protection which subjects to cost constraint, or a service considering trade-off between cost and privacy penalties. Based on the user's choice, every possible allocation is examined and the best allocation will be selected.

Given a set of sub-allocation decision $\{A(G_1), A(G_2), ..., A(G_n)\}$, the disclosure condition matrix D can be generated by combining all the sub-allocation decisions as follows:

$$\exists(v_i \in G_k \ \&\ A(G_k) = 1) \xrightarrow{\forall i,k} d_{ii} = 1$$

$$\exists(e_{ij} \in G_k \ \&\ A(G_k) = 1) \xrightarrow{\forall i,j,k} d_{ij} = 1$$

$$\exists(d_{ii} = 1 \& d_{jj} = 1 \& P = \{d_{ik_1} = 1, d_{k_1k_2} = 1, \ldots, d_{k_t j} = 1\}) \xrightarrow{\forall i,j,k_1,\ldots,k_t} d_{ij} = 1$$

Using RPV= $\sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \times r_{ij}$ we can get the risk penalty value of the allocation. Then the total cost is calculated by using $\sum_{i=1}^{n} C(G_i)$ *where* $\begin{cases} C(G_i) = C_r(G_i) & when\ A(G_i) = 0 \\ C(G_i) = C_l(G_i) & when\ A(G_i) = 1 \end{cases}$

As there are four sub-queries in our motivating example, $2^4 = 16$ allocations are provided in total. In Table 3, we list all the possible allocations associated with the cost and privacy penalty. Suppose that the user prefers to select a service that produces the lowest privacy penalty value while costing less than \$250. From Table 3 we can see that the allocation which meets this requirement is {0, 0, 1, 1}, which means, sub-queries 1 and 2 should be executed in the repository cloud while sub-queries 3 and 4 should be executed in the local cloud.

Table 3. Allocation Results

$\{A(G_1), A(G_2), A(G_3), A(G_4)\}$	Cost	Penalty
{0, 0,0,0}	335	0
{0,0,0,1}	288	8
{0,0,1,0}	290	8
{0,0,1,1}	205	14
{0,1,0,0,}	280	∞
{0,1,0,1}	233	∞
{0,1,1,0}	235	∞
{0,1,1,1}	188	∞
{1,0,0,0}	275	18
{1,0,0,1}	228	26
{1,0,1,0}	230	22
{1,0,1,1}	183	28
{1,1,0,0}	220	∞
{1,1,0,1}	173	∞
{1,1,1,0}	175	∞
{1,1,1,1}	128	∞

4.3.2 Dynamic Inter-cloud Data Integration System

Although the previous system can help the user to find the best trade-off between cost and privacy risk, it cannot provide sufficient flexibility regarding privacy risk evaluation. In many scenarios, the query can be executed several times with different input data. The user may want to evaluate the privacy risk value based on the input data or intermediate searching results which are generated during run-time. For example, the user may only want to hide some particular patients' information from the repository cloud and local cloud during the searching process. However, he cannot predict whether the names or IDs of those patients will appear in the intermediate searching results or not. Using a single static value to define the privacy risk of corresponding information disclosure in R_{m*m} is hard and inefficient for the user. In that case, a dynamic risk value is needed to meet user's privacy requirement. Thus, we improve our system to support the dynamic risk evaluation. Meanwhile, a corresponding allocation approach is required to handle the dynamic risk changes.

4.3.2.1 Dynamic risk evaluation

To define dynamic risk value, user should provide a set of rules to the repository cloud that can help it to calculate the risk value dynamically. A dynamic risk evaluation rule has to solve three

problems: i) which risk values need to be evaluated dynamically, ii) based on what the repository cloud can change in the risk values, and iii) how to compute the risk values. Based on the above thought, we define that $DR =< R'_{m*m}, SD, RC >$ is a dynamic risk evaluation rule attached to v_i where R'_{m*m} is the m by m matrix representing which risk value will be evaluated dynamically, SD is the user defined sensitive data set in v_i, and RC represents the dynamic risk computing formulas.

In detail, the R'_{m*m} contains only "1" and "0" where "1" represents that the risk value of the corresponding node or edge will be changed dynamically according to this rule, and "0" means the risk evaluation of the corresponding node or edge is not affected by this rule. For instance, if the user only wants to change the risk value of the edge "patient-ID", based on what information appears in the searching results of "patient", the R'_{m*m} can be defined as in Fig. 8.

	D	P	I	Dr	M
D	0	0	0	0	0
P	0	0	1	0	0
I	0	0	0	0	0
Dr	0	0	0	0	0
M	0	0	0	0	0

Fig.8 An instance of a data disclosure condition matrix

Normally, the dynamic risk evaluation rule attached to v_i only affects the risk values of v_i, it's subsequent nodes and edges (in DAG) since the dynamic allocation approach can be applied to the new risk value in the current searching process. However, if the searching tasks will be repeated multiple times, the dynamic risk evaluation rule attached to v_i can also affect other nodes or edges. The new risk values cannot be used in the current searching process, but can be used in future ones.

The sensitive data set SD may contain several sub-sets representing different sensitivity levels. For instance, the user may define $SD = \{SD_1\}$ meaning that there is only one sensitive data sub-set, or $SD = \{SD_1, SD_2\}$ denoting that there are two sub-sets with different sensitivity level. However, user does not have to define the exact sensitivity value in SD. Instead, those sub-sets are used in RC to create different privacy risk values. In each sub-set of SD, there are several sensitive data items provided by user. The data items should be hashed before they are delivered to repository cloud. Regarding v_i, the hashing function applied to intermediate results and that applied to sensitive data set should be same. Thus, the repository cloud is able to compare the data in both sets.

The risk computing RC may also contain several formulas as well where each formula uses one or more sensitive data sub-sets and intermediate searching results ISR as input parameters and produces a privacy risk value. A typical formula $f_{r_{ij}}(SD_k, ISR) \in RC$ for computing the risk of r_{ij} can be defined by user as follow:

$$r_{ij} = f_{r_{ij}}(SD_k, ISR) = \begin{cases} \infty & if\ SD_k \cap ISR \neq \phi \\ 0 & otherwise \end{cases}$$

Using the above formula, the repository cloud compares the data in ISR and the data in SD_k. If any data item is matched, the repository cloud dynamically adjusts the privacy risk value r_{ij} to ∞. Otherwise, r_{ij} is set to 0. Since the maximum possible risk value of r_{ij} is ∞, the repository cloud should take ∞ as the default risk value of r_{ij}. The default risk value is used to establish static allocation before starting a searching process.

4.3.2.2 Dynamic allocation

After evaluating the privacy risk values, the repository cloud has to change the allocation according to the new risk values. Given that the risk value of a node v_i or an edge e_{ij} is changed, the repository cloud only changes the allocation for the searching tasks containing its sub-sequent nodes or edges in the current searching process. The allocation method is the same as the static allocation method which enumerates all possible situations and chooses the best one among them. If the searching tasks are repeated multiple times, the repository cloud may need to adjust the allocation for the entire integrated searching task before starting a new searching process.

4.3.2.3 Example

We provide an example to illustrate how the dynamic inter-cloud data integration system works. Suppose the static setting of the system is the same as we presented before. Now, we add one dynamic risk evaluation rule and attach it to "patient". Let $DR =< R'_{m*m}, SD, RC >$ be the rule, and the instance in Fig. 9 be the matrix R'_{m*m} in that rule. Let $SD_1 \in SD$ be the only sensitive data set, and $\{H(Alice)\} \in SD_1$ be the only sensitive data item in that set. There is only one formula in RC:

$$r_{23} = f_{r_{23}}(SD_1, ISR) = \begin{cases} \infty & if\ SD_1 \cap ISR \neq \phi \\ 0 & otherwise \end{cases}$$

The above formula means that if $H(Alice)$ is found in ISR, the risk value of the edge "patient-ID" should be set as ∞, if it is not then the value is 0.

	D	P	I	Dr	M
D	4	9	9	1	1
P	–	5	0	–	5
I	–	–	3	–	1
Dr	–	–	–	3	1
M	–	–	–	–	0

Fig.9. New penalty matrix

Now, suppose that ISR has been produced during runtime and the contents of ISR are $\{H(Tom), H(Susan)\}$. As $SD_1 \cap ISR \neq \phi$ is not satisfied, the risk matrix should be changed and presented as Fig. 9.

Based on the new penalty matrix, the repository cloud re-evaluates the allocation results, which is presented in Table 4.

Table 4. Allocation Results

$\{A(G_1), A(G_2), A(G_3), A(G_4)\}$	Cost	Penalty
{0,0,1,0}	290	8
{0,0,1,1}	205	14
{0,1,1,0}	235	13
{0,1,1,1}	188	19

Since G_1 and G_3 do not contain the sub-sequent nodes or edges of "patient-ID", their allocation results ($A(G_1) = 0$ and $A(G_3) = 1$) are not changed. Thus, the repository cloud only checks four possible allocations and decides that the previous allocation result $(0,0,1,1)$ should be replaced by the new one $(0,1,1,0)$ since the penalty of $(0,1,1,0)$ is lower than the cost of $(0,0,1,1)$ while the cost of both allocations are less than $250.

5 Conclusions

With the offer from cloud computing providers, we are able to utilize pay-as-you-go resources together with our own and shared resources. However, we should also consider which part of the data should be executed in cloud computing systems in order to balance the trade-off between costs. In this paper, a dynamic content-based inter-cloud data ingesting system is presented. Users can evaluate the privacy risk value based on the content (input data or intermediate searching results) which are generated during run-time. Our proposed system strikes a balance between the privacy requirements from users and the cost for data protection and processing. In contrast to existing data sharing techniques, our method is more practical as the cost for technical supporting privacy must be considered in the commoditized cloud computing environment.

Acknowledgment

"This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2011-(C1090-1121-0003)

References

[1] I. Foster, C. Kesselman (Eds.), The Grid 2: Blueprint for a new computing infrastructure, morgan Kaufmann, San Francisco, CA, USA, 2003.

[2] W. Kim, cloud computing: Today and Tomorrow, *Journal of Object Technology* 8 (1) (2009) 65–72.

[3] Top Threats to Cloud Computing, cloud security alliance, https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

[4] Stephen S. Yau et al, a privacy preserving repository for data integration across data sharing services, IEEE transactions on services computing.

[5] Shaozhi Ye, Felix Wu, Raju Pandey, Hao Chen, Noise Injection for Search Privacy Protection, cse, vol. 3, pp.1-8, 2009 International Conference on Computational Science and Engineering, 2009.

[6] Yuan Tian, Biao Song, Eui-nam Huh, Inter-cloud Data Integration System Considering Privacy and Cost, ICCCI 2010, Volume 6421/2010, pp.195-204.

[7] Steve Mansfield-Devine, Danger in the clouds, Network security, http://www.webvivant.com/dangers-in-the-cloud.html (Accessed on May 15, 2010)

[8] Chee Shin Yeoa, Srikumar Venugopalb, Xingchen Chua, and Rajkumar Buyya, Autonomic metered pricing for a utility computing service, Future Generation Computer Systems, 2009.

[9] Tarry Singh, Pavan Kumar Vara, Smart Metering the Clouds, 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, IEEE Computer Society, Washington, DC, USA, pp. 66-71.

[10] Gerardo Canfora, Elisa Costante, Igino Pennino, and Corrado Aaron Visaggio, A three-layered model to implement data privacy policies, Computer Standards & Interfaces, Elsevier Science Publishers B. V. , Amsterdam, The Netherlands, The Netherlands, 2008, pp. 398-409.