Different types of Data Privacy Preserving Repository and Schemes

Mrs.V. M. Deshmukh
Professor, CSE
msvmdeshmukh@rediffmail.com

Miss. A. P. Ghatol Assistant
M.E (1st year, CSE)
ashwinipghatol@gmail.com

Prof.Ram Meghe Institute of Technology and Research,Badnera.

***Abstract*—Current data sharing and integration among various organizations requires a central and trusted authority to collect data from all data sources and then integrate the collected data. This process tends to complicate the update of data and to compromise data sources privacy and security. In this paper, we are going to discuss about different types data privacy preserving repository and schemes. With the repository, data sharing services can update and control the access and limit the usage of their shared data, instead of submitting or sharing the whole data to authorities may in an organization or over cloud computing. There are major differences between the proposed repository and the existing one such as not the whole data will be share, user will not be able to get information to use for other purpose. The repository will promote data sharing and integration. We also highlight a scheme called Security Policy Integration and Conflict Reconciliation (SPICR) layer which will also maintain the security and integrity of the data in an unambiguous environment. With this proposed layer, data sharing services can control the access, limit the usage of their shared data, and improves data sharing and efficiency of the data-centralized repository making the system scalable with little human intervention.**

***Keywords*— Privacy concerns of service-oriented solutions, privacy management, services composition, cloud; Security policy integration, Conflict reconciliation; Access control policy.**

I. Introduction

Much effort has been devoted to facilitating data sharing and integration among various organizations. Existing data sharing and integration systems are usually implemented as centralized data warehouses collecting and storing data from various data sources where the data sources and data warehouses expect to sign business agreements in which the scope of the shared data and corresponding privacy policies are specified. This will specify that all shared data will be kept confidential and will not be disclosed to other unrelated third parties or be used for other purposes. While this solution works well for a single organization or a federation of organizations, where trust relations have been well established, serious problems will arise when some data warehouses cannot be trusted by data sources. In such cases, data sources will refuse to share their data because they have no control of its usages and disclosures once the data is shared. In fact, data warehouses indeed can reveal or abuse the shared data. Furthermore, even if data warehouses adhere to the agreement, there is no guarantee that they have sufficient capability to protect the data. This technique is in work when the environment of computing is homogenous. But during the ubiquitous computing environments, they pose there certain unique challenges for generating an integrated congruous security policy set. In this paper we will present a privacy preserving repository to accept integration requirements from users, help data sharing services share data and safeguard their privacy, collect and integrate the required data from data sharing services, and return the integration results to users. Our repository will focus on the matching operations and has the following major benefits:

1. The data sharing services can update and control the access and usages of their shared data. That is, data-sharing services can update their data whenever necessary and determine who and how their shared data can be used.
2. The data is shared based on the need-to-share principle, which means that the released information of the data is sufficient to support users' integration requirements, but contains no more information of the data.
3. The repository's capability is limited to collecting data from data sharing services and integrating the data to satisfy users' integration requirements. Except the information needed to be revealed for data integration, the repository will not have extra information about the data and cannot use it for other purposes.

The privacy preserving repository which is presented has the same work over the cloud.

We will also define a SPICR layer to accept integration requirements from multiple set users belonging to different organizations and help the repository and data sharing services share data and safeguard their privacy. The centralized repository collects and integrates required data from data sharing services, and returns the integration results to users. Our layer will focus on the following major benefits such as:

1) The SPICR resolves conflicts among dynamic set of users and generates non-ambiguous and congruence set of security policies and allow the repository to share data on a need-to-share principle.

2) The SPICR allows negotiation among conflicting policies.

We show how efficiently the SPICR facilitate user collaborations among participating to access and use the appropriate resources through direct interaction with the resource owners.

II SYSTEM ARCHITECTURE AND ASSUMPTIONS

A) PRIVACY PRESERVING REPOSITORY:

In existing data integration systems, it is assumed that there is a central and trusted authority collecting all data from data sharing services and computing integration results for users based on the collected data. In our system, as shown in Fig. 1, the repository collects only the required data for user's integration requests from the data warehouses or clients. Fig. 1.a shows privacy preserving repository for data integration across data sharing services and fig. 1.b shows privacy preserving repository for securing data across cloud. It is assumed that the repository will correctly construct the query plans for users' integration requirements, decompose query plans, discover and fetch data from distributed data sharing services, integrate all data together, and, finally, return the final results to users. Furthermore, it is also assumed that the repository is granted the access to the shared data by all data sharing services, and all shared data is well protected. Because the data sharing services use the context-aware date sharing algorithm, the repository cannot learn extra information from the inferential relations of the information it obtains during the integration process. The repository consists of two components: the query plan wrapper and the query plan executor. The query plan wrapper is responsible for analysing integration requirements and constructing query plans for the query plan executor. Since the wrapper development and optimization have been extensively studied we assume that the query plan wrapper can select data sharing services and construct a query plan graph from users' integration requirements. Based on this assumption, it will focus on how to decompose the query plan graph into a set of small sub graphs for each data sharing service to guide data sharing services to prepare shared data.The query plan executor is responsible for executing query plans to fetch data from data sharing services and producing the final results..

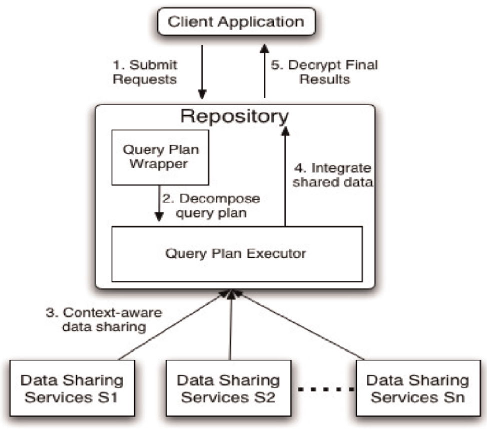

Fig. 1.a) Privacy preserving repository for data integration across data sharing services.

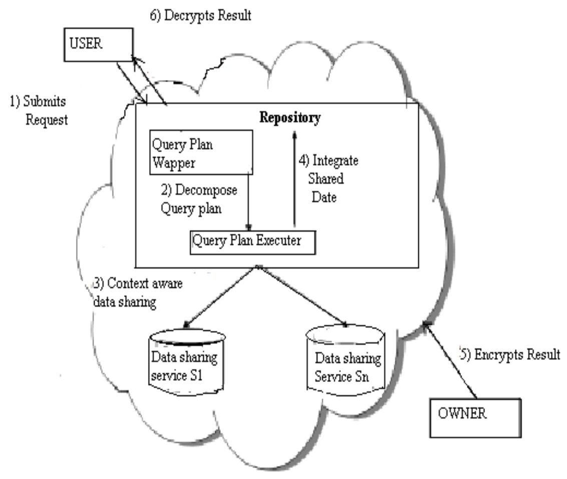

Fig. 1.b) Privacy preserving repository for securing data across cloud

B) SPICR LAYER:

Our SPICR layer supports the repository to collect only the required data for user's integration requests. We assume that our SPICR layer will correctly evaluate the users' request to generate integration requirements and pass it to the repository to construct the query plans for users' integration requirements, decompose query plans, discover and fetch data from distributed data sharing services, integrate all data together, and, finally, return the final results to users. Fig. 2 shows the SPICR layer.

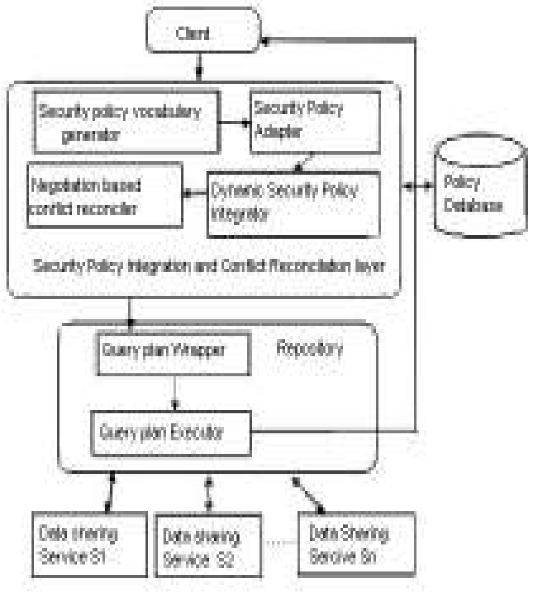
Client
Security policy vocabulary generator
Security Policy Adapter
Negotiation based conflict reconciler
Dynamic Security Policy integrator
Policy Database
Security Policy Integration and Conflict Reconciliation layer
Query plan Wrapper
Repository
Query plan Executor
Data sharing Service S1
Data sharing Service S2
Data Sharing Service Sn

Fig.2 the SPICR Layer

An easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

II. Overview of Approaches

A. Privacy Preserving Repository

The goal is to develop a repository to facilitate the data integration across data sharing services. In this section, we will present the process of the data integration via our privacy preserving repository REP. Before going in to the process we have to know what the query plan and the privacy preserving repository means. They are defined as follows.

Definition 1 (Query Plan): A query plan P is a partially ordered set of queries {p1; p2; ------; pmg} with two properties:

1) Each pi can be evaluated only after all of its precedent queries have been evaluated.

2) Each pi can use the data directly from data sharing services or its precedent queries' outputs as inputs.

The final result of P is the outputs of pi with no successive queries, and all other queries outputs are intermediate results [6].

The above definition indicates that a query plan P has a much richer structure than a single query or a set of independent queries. First, there is a partial order relation among queries in P. Second, only the outputs of queries in P without successive queries constitute the final result and all other intermediate results should be protected.

Definition 2 (Privacy Preserving Repository): For a query plan P = {p1; p2; ------; pmg} and a repository REP,REP is a privacy preserving repository for data integration if REP executes P in a privacy preserving manner then:

1) *REP only has P's final result encrypted with user's public key and has no information on P's intermediate results;*
2) *and REP cannot use the data shared for P to evaluate any other queries* [7].

The process of the data integration via our privacy preserving repository REP can be summarized as follows:

Step 1: The user sends his/her public key pk and the requirements about data integration to our repository REP.

Step 2: The query plan wrapper of REP analyses the user's integration requirements and converts them to a query plan graph G, and then decomposes G to a set of sub graphs {G1;G2; _ _ _;Gmg }using the *Decompose Algorithm* and sends the sub graphs to the query plan executor. Every sub graph Gi represents the context of one data sharing service for conducting context-aware data sharing.

Step 3: For every Gi, the query plan executor looks for the corresponding data sharing service Si and sends Gi to Si, which prepares the data using the *Context-Aware Data Sharing Algorithm* and returns all randomized data to the query plan executor.

Step 4: The query plan executor executes the *Integrate Algorithm* on all returned data to execute the G and outputs the results *FinalRes* of user's request, which is encrypted with the user's public key pk.

Step 5: REP sends *FinalRes* to the user who then decrypts it with his/her secret key sk [10].

B. SPICR Layer:

Our approach consists of nine major steps.

Step1: The user provides his /her public key and the integration requirements of data. Apply homomorphic encryption algorithm for all values of selected attribute.

Step2: The Security Policy vocabulary generator checks the policy specifications for the requested user. The generator resolves ambiguity among the security policies using approach.

Step3: The Security policy Adapter adapts the existing security policies for the extended user set. It decides whether to grant the user to access the data or deny the user.

Step4: The Dynamic Security policy Integrator integrates security policies to generate a new set of security policies that is not present in either of collaborating organization.

Step5: The Negotiation-based conflict Reconciler resolves the conflict between the security policies and makes a compromise by selecting the weakest security policy in the policy hierarchy.

Step6: Once the user is granted access to the data , the query plan wrapper of repository REP generates a query plan graph G, and then decomposes G to a set of sub graphs {G1,G2,…,Gm}using the Query Decomposition Algorithm and sends the sub-graphs to query plan executor. Every sub-graph Gi represents the context of one data sharing service for conducting context-aware data sharing.

Step7: For every Gi, the query plan executor looks for the corresponding data sharing service Si and sends Gi to Si

Step8: The query plan executor executes the Integrate Algorithm on all returned data to execute the G and outputs the results final result of user's request, which is encrypted with the user's public key.

Step9: REP sends final result to the REP who then decrypts it with his/her secret key [10].

IV EXPLANATION OF THE PROPOSED MODELS

A) Privacy Preserving Repository:

Query Plan Decomposition:

The query plan graph G contains the information about all the data sharing services but each data sharing services only needs to know its related data hence the query plan wrapper will decompose G and send only the query plan sub-graphs to their corresponding data sharing services. For the given query plan graph G=(V, E, C) with m nodes, the decomposition algorithm which will construct a sub-graph Gi for each node vi by extracting vi's adjacent nodes and corresponding edges and the labels attached to edges . The sub graph Gi of vi as (Vi; Ei; Ci; ri), where Vi consists of all vi's adjacent nodes, Ei all the adjacent edges, Ci all the labels attached with Ei, and ri contains all random numbers assigned to Ei are denoted. Gi represents all data integration operations of the data sharing service represented by vi [1].

Context-Aware Data Sharing:

In *Decomposition algorithm*, the query plan graph G is decomposed to a set of sub graphs. For each data sharing service, its sub graph Gi consists of the information about other data sharing services whose data will be integrated with its own data and how the data will be integrated together. Hence, we call the sub graph of vi the context of the data sharing service of vi in current G. Because data sharing services are aware of its context in the whole data integration process, they can determine which information should be shared and how to limit the usage of the shared data. The *Context-Aware Data Sharing Algorithm* help data sharing services share information with the repository. It focus on the matching operations to determine whether two records are matched according to the equality test between their attribute values. The matching between two data records is replaced by the matching between their hash values. Hash functions low conflict probability ensures the correctness of the hash-based matching and hash functions one-way property enables a third party to match two data records without revealing their values. Thus, the hash function is a simple solution for privacy preserving data matching [1].

Data Integration:

When the repository receives the shared information from all data sharing services, the repository follow the query plan graph G and integrate the received information together to compute the integration results for the user this is done with the help of the *Integration Algorithm*.

The integration algorithm work as follows:

Initialize: Before the repository REP evaluates an edge, it first retrieves the edge's label information from G to find out which attributes are to be matched. Meanwhile, REP collects all attributes shared by the edge tail node for its own out-edges as AttrOut.

Match: In this step, REP scans and matches tail nodes' records with the records from head nodes.

Remove random factors: Assume that the edge's tail node's record rec passes the evaluation of the edge. To use rec to evaluate the tail node's out-edges, REP first needs to remove random factors from the shared information about rec for all attr belongs to AttrOut

Collect outputs: If the edge's tail node is the sink node t, REP collects the outputs of the whole query plan in this step [2].

B) SPICR Layer:

The working of the SPICR layer is explained below:

Step 1: Providing key and encryption algorithm: In the first step the user provides his /her public key and the integration requirements of data to SPICR layer. Then we apply encryption algorithm i.e. homographic encryption algorithm for all values of selected attribute. As the algorithm it deals with dynamic environment having secure multi-party computation.

Step 2: Checking User Integration Requirements Vocabulary: Then the Security Policy vocabulary generator checks the policy specifications for the requested user. The generator resolves ambiguity among the security policies. Information access may require restrictions based on the content and context related to the access requests. It uses a specification language to generate security policy specifications for each collaborating organization, where a security policy can be specified as a quadruple with the components derived from the ontology containing the key concepts needed for specifying security policies.

Step 3: Security Policy Adaptation: The Security policy Adapter module adapts the existing security policies for the extended user set. It decides whether to grant the user to access the data or deny the user. When an access to a resource is requested by a user from other participating organizations, the resource owner should make the decision on whether to grant the request to the resource. However, since the requester is from another organization, the resource owner may not have much knowledge about this requester. So it evaluates a stranger's trustworthiness based on where he comes from and how he is trusted in the organizations he belonged before.

Thus, it will use a similarity-base security policy adaptation algorithm to help the resource owner make a decision based on evaluating whether the participating organizations that the user comes from have similar security policies, and whether the user request will be granted under those security policies. However, for dynamic collaborations in ubiquitous computing environments, we present an alternative approach based on policy adaptation and negotiation to achieve dynamic security policy integration with minimum human intervention.

Step4: Generating composite Set of security Policies: The Dynamic Security policy Integrator module integrates security policies to generate a new set of security policies that is not present in either of collaborating organization. It uses a negotiation-based approach to generating a congruous set of security policies which can be accepted by all participating organizations. Participating organizations will first provide their inputs towards the generation of the policies, and then make compromise in order to resolve possible conflicts and reach an agreement. This is used to address possible conflicts which arise from the integration, and to specify new security policies for the resources generated by the collaboration. In both cases, no single organization can claim the sole ownership of the generated resources, and thus no security policies of any participating organization should take priority over those of other organizations.

Step 5: Resolving Conflict among Security Policies based on Negotiation: The Negotiation-based conflict Reconciler resolves the conflict between the security policies using Negotiation-based Policy Algorithm for conflict reconciliation and makes a compromise by selecting the weakest security policy in the policy hierarchy. Thus, instead of rejecting a possible collaborator because of some conflicting security policies, collaborating organizations often prefer to take certain risk and move forward with the collaboration by relaxing their security policies and adopting some weaker security policies to resolve the conflicts. For a specific resource, each organization specifies a chain of security requirements instead of just one security requirement. The head of the chain is the most desirable security policy travelling through the chain from the head, the security requirements become weaker and weaker. The SPICR layer interacts with the Security Policy Database that maintains the policies along with user credentials, roles and their permissions in the form of XML schemas. Once the credential is evaluated, the role is mapped and the context is extracted, the user is granted access to the data. The SPICR generates information for data integration process. This information will help the repository figure out what data should be retrieved from data sharing services and how to integrate the data together, and will help data sharing services share their data without revealing more information than the evaluation of G needs.

Step6: Query Decomposition: The query plan wrapper of repository "REP" generates a query plan graph G based on XML specification i.e. QPSL (Query plan specification language), and then decomposes G to a set of sub-graphs {G1,G2,…,Gm}using the Query Decomposition Algorithm and sends the sub-trees to query plan executor. Every sub-graph Gi (operator,attribute1, attribute2) of one data sharing service for conducting context-aware data sharing.

Step7: Context-aware Data sharing: To further protect the privacy of such information; we use a Context-Aware Data Sharing algorithm to randomize the result. We focus on the matching operations to determine whether two records are matched according to the equality test between their attribute values. Basically, the matching between two data records can always be replaced by the matching between their hash values. The process uses context-aware data sharing algorithm.

Step8: Data Integration: When the repository receives the shared information from all data sharing services, the repository should follow the query plan graph G and integrate the received information together to compute the integration results for the user. The integration process uses the Integration.

Step9.Decryption: REP sends final result to the REP who then decrypts it with his/her secret key [3] [4].

V. Conclusion:

In this paper, the presented privacy preserving repository to integrate data from various data sharing services and an approach to security policy integration and conflict reconciliation for collaborating organizations in ubiquitous computing environments. The repository only collects the minimum amount of information from data sharing services or from the client across the cloud which is based on user's integration requests, and data sharing services can restrict our repository to use their shared information only for user's integration requests, but not other purposes. For the SPICR layer, a resource owner can adapt its existing security policies to cover the new users from collaborating organizations using the similarity indices as a selection criterion and make a decision based on their inputs. Collaborating organizations with conflicting security policies can try to resolve the conflicts by gradually making security compromise and relaxing their security policies until the conflict reconciliation process reaches certain compromise threshold associated with that particular collaboration. Thus, allow the user to integrate data with the help of repository without any conflicts.

VI Future Work:

Future research along this topic includes how to extend the expressiveness of our specification language, enable our repository to support more types of data integration operations, and improve of our repository's performance for much larger scale of data size. A possible approach for performance improvement is to enable the precomputation of data, which allows the data sharing services to obtain some preliminary information about their data for accelerating data sharing. In future we will be coming up with a secure development scheme of the hash function and its implementation in public key cryptography for maintaining the privacy of data in the cloud. The future research work also includes about how to evaluate the security of the SPICR layer and on the design of policy languages.

References

[1] Stephen S. Yau, Fellow, IEEE, and Yin Yin, "*A Privacy Preserving Repository for Data Integration across Data Sharing Services*", IEEE Transactions on Services Computing, vol. 1, no. 3, July- September 2008.

[2] Ranjita Mishra, Sanjit Kumar Dash, Debi Prasad Mishra, Animesh Tripathy paper "*A Privacy Preserving Repository for Securing Data across the Cloud*"

[3] Beneyaz A. Begum, Rajesh K. Thakur, Prashanta K.Patra paper "*Security Policy Integration and Conflict Reconciliation for Data Integration across Data Sharing Services in Ubiquitous Computing Environments*". Int'l Conf. on Computer & Communication Technology.

[4] S.S. Yau and Z. Chen, *"Security Policy Integration and Conflict Reconciliation for Collaborations among Organizations in Ubiquitous Computing Environments"*, Proc. Fifth Int'l Conf. Ubiquitous Intelligence and Computing, pp. 3-19, 2008.

[5] Kamber, et al., *"Data Mining Concepts and Techniques"*, 2nd Edition,Elsevier, New Delhi, 2009, p12-13

[6] R. Agrawal, A.V. Evfimievski, and R. Srikant, *"Information Sharingacross Private Databases,"* Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 86-97, 2003.

[7] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, *"Order-Preserving Encryption for Numeric Data,"* Proc. ACM Int'l Conf. Management of Data (SIGMOD '04), pp. 563-574, 2004.

[8] M. Bellare, A. Boldyreva, and A. O'Neill,*"Deterministic and Efficiently Searchable Encryption,"* Advances in Cryptology (CRYPTO '07), pp. 535-552, 2007.

[9] Stallings, W., *"Cryptography and Network Security",* 3rd Edition, Pearson Education, New Delhi, 2003, p8.

[10] Rakshit, A. , et. Al, *"Cloud Security Issues",* 2009, IEEE International Conference on Services Computing

[11] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, *"Above the clouds: A Berkeley view of cloud computing,"* EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[12] P. Mell and T. Grance, *"Draft-NIST working definition of cloud computing - v15,"* 21. Aug 2005, 2009.

[13] M. Bellare, A. Boldyreva, and A. O'Neill, "*Deterministic and Efficiently Searchable Encryption",* Advances in Cryptology (CRYPTO'07), pp. 535-552, 2007.

[14] Ribeiro, C., Zuquete, A., Ferreira, P., "*SPL: An access control language for security policies with complex constraints"*. In: Proc.Network and Distributed System Security Symp(NDSS 2001), pp. 89–107 (2001).

[15] Agrawal, D., Giles, J., Lee, K.W., Lobo, J.: *"Policy Ratification"*. In: Proc. 6th IEEE Int'l Workshop on Policies for Distributed Systems and Networks (POLICY), pp. 223–232 (2005).

[16]Bhatt, R., Joshi, J.B.D., Bertino, E., Ghafoor, A.: "*Access Controlin Dynamic XML-Based Web Services with XRBAC"*. In: Proc.1stInt'lConf. on Web Services (2003), http://www.sis.pitt. edu/~jjoshi/ICWS_XRBAC_Final_PDF.pdf.

[17] Yau, S.S., Chen, Z.: *"A Framework for Specifying and Managing Security Requirements in Collaborative Systems"*. In: Yang, L.T., Jin, H., Ma, J., Ungerer, T. (eds.) ATC 2006. LNCS, vol. 4158, pp. 500– 510. Springer, Heidelberg (2006).

[18] "*OASIS, eXtensible Access Control Markup Language (XACML)*"Version2.0,OASIS standard (2005), http://docs. oasisopen.org/xacml/2.0/access_control-xacml-2.0-core-pec-os.pdf.