On Monitoring Information Flow of Outsourced Data

Anne V.D.M. Kayem
Department of Computer Science
University of Cape Town
Private Bag X3
Rondebosch
7701
South Africa
Email: akayem@cs.uct.ac.za

***Abstract*—Data outsourcing is an Internet-based paradigm that allows organizations to share data cost-effectively by transferring data to a third-party service provider for management. Enforcing outsourced data privacy in untrustworthy environments is challenging because the data needs to be kept secret both from unauthorized users and the service provider (SP). Existing approaches propose that the data owner(s) encrypt the data before it is transferred to the service provider to preserve confidentiality. Access is only granted to a user initiated program if the key presented can decrypt the data into a readable format. Therefore the data owner can control access to the data without having to worry about the management costs. However, this approach fails to monitor the data once it has been retrieved from the SP's end. So, a user can retrieve information from the SP's end and share it with unauthorized users or even the SP. We propose a conceptual framework, based on the concept of dependence graphs, for monitoring data exchanges between programs in order to prevent unauthorized access. The framework has a distributed architecture which is suitable for data outsourcing environments and the web in general. Each data object contains a cryptographic tag (like an invisible digital watermark) that is computed by using a cryptographic hash function to combine the checksum of the data and the encryption key. In order to execute an operation with a data object the key presented for decryption must match the one associated with the user's role and generate a cryptographic tag that matches the one embedded into the data. Tracing data exchanges, in this way, can leverage data privacy for organizations that transfer data management to third party service providers.**

I. Introduction

The emergence of the Internet made the power of distributed computing a reality that organizations have and will continue to tap on to provide an information sharing environment that strives to be dynamic and reliable with guaranteed quality of service. Data outsourcing is an interesting information sharing application in which organizations transfer the management of data intensive applications like data publishing and data warehousing to third-party service providers (SP) for management. The advantage of this is that it reduces their management costs and makes the data available so that users are able to access information according to their needs. Yet, information sharing in this context raises challenges pertaining to trust and data privacy that need to be addressed in order to ensure that sensitive information is not accessible to unauthorized parties.

The necessity of trust and data privacy makes it important to protect the data from access by unauthorized users including the SP [1], [2]. For example, health insurance companies that wish to outsource data concerning their clients' personal information must protect the information even from the SP in order to avoid breaches of data privacy laws. Therefore, the data owners need some way of protecting the data that is sent to the SP and also preventing users who are authorized to view the data from sharing it with unauthorized parties.

A. Problem Statement

Approaches based on using cryptographic keys to protect the data propose that the data owners encrypt the data before it is transfered to the SP [3], [4]. This ensures that the data is not visible to the SP and the data owner can decide who gets to see the data by sharing the keys only with users that are authorized to view the data. As well, in order to minimize the cost of updating security policies, the SP can apply a second layer of encryption to the data so that security policy updates are handled by updating the affected keys and re-encrypting the data only on the SP's end [1], [2]. This is in contrast to having to request and have a new encrypted copy re-transmitted by the data owner which is expensive in terms of bandwidth. However, while cryptographically supported access control approaches protect the data from being read by the SP, preventing authorized users from retrieving the data and then proceeding to share its contents with unauthorized users including the SP is a challenging problem.

For instance, consider the scenario depicted in Fig. 1 in which an organization such as a health insurance firm outsources the storage of client data. The company wants to be able to allow medical students or other parties to retrieve data and perform statistical calculations without revealing the identity of the clients concerned. Consider the case of two users, Alice and Bob, who both have access to different parts of the data. Alice can only access records concerning clients who are older than 70 while Bob can only access data concerning patients who are chronically ill.

In order to find out which clients are both over 70 and chronically ill, Alice can decide to persuade Bob to share

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

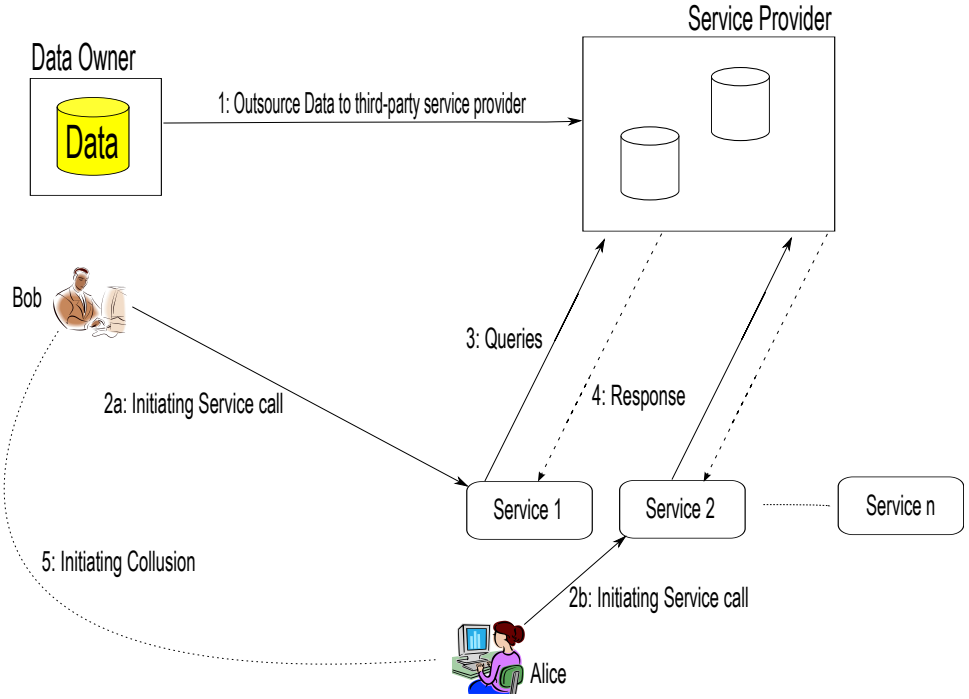

Fig. 1. Collusion between Alice and Bob to access unauthorized data

his data with her in exchange for taking a peep at her data. Current approaches based on cryptography do not prevent cases of illegal data sharing such as the one we have just described. Therefore, security policies for controlling the flow of information between users and/or the SP, are need in order to prevent unauthorized exchanges of information.

B. Contributions

This paper presents a conceptual framework that uses cryptographic tags to monitor data exchanges between users or user initiated programs, like services, in order to prevent unauthorized data sharing. In the framework, data owners transfer management of their data to a third party SP from where it can be accessed by users. Access to the data on the SP's end can either be requested directly or orchestrated by a service that is invoked by a user with a role(s) that authorizes him/her to retrieve the data requested. In the case where the outsourced data is protected by a double encryption scheme, all the SP requires is that the service presents a key that matches the one associated with the role of the user responsible for the invocation. The user will then use the key that the data owner assigned to him/her to decrypt the retrieved data into a readable format. Illegal information flows between users or services are prevented essentially by labeling each data object with a cryptographic tag (similar to an invisible digital watermark) that is computed by using a cryptographic hash function to combine the checksum of the data and the encryption key. Embedded in each service is an Information Flow Control Module (IFCM) that monitors the usage of the data ensuring that the data is only manipulated in ways that are authorized by the access control mechanism. The IFCM relies on two knowledge bases that contain the security policies as well as the rules for verifying the cryptographic tags. The security policies are used to construct the dependency graph to indicate which services may share the information that has been retrieved, while the rules for verifying the cryptographic tags are used to determine whether a data object can be read by a user. This approach to tracing data exchanges between services can leverage privacy of data that organizations transfer to third party SPs, and thereby, make customers and organizations more willing to use applications that depend on outsourced data.

The framework has a distributed architecture which makes it suitable for suitable for data outsourcing environments and the web in general. Potential applications can be found in banks and hospitals where customers' private financial and medical information is transferred to third party service providers for statistical analysis, as well as in social computing applications where there is a need to secure shared information like photographs, blogs or documents from unauthorized access.

C. Organization

The rest of the paper is structured as follows. In Section II, we present related work on the "Database-as-a-service" paradigm in relation to cryptographically supported access control to outsourced data. In Section III, we present our framework for monitoring the flow of outsourced data between services and discuss the challenges involved in implementing the framework. Section IV gives an example application scenario for the framework in order to highlight the importance of securing information exchanges between users. Concluding remarks and topics for future work are offered in Section 5.

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

II. Related Work

The "database-as-a-service" paradigm emerged with the purpose of finding ways to facilitate resource outsourcing, by data owners, in a secure and reliable way to SPs on the Internet. Most solutions focus on methods of executing queries efficiently and securely on encrypted outsourced data [3]–[6]. Typically, these methods are centered around indexing information stored with outsourced data. The indexes are useful for returning responses to queries without the need to decrypt the data (or response to the query). Only the party authorized to view the response should, in principle, be able to decrypt the result returned in response their query. The challenge in developing indexing techniques is in establishing a reasonable trade-off between query efficiency, exposure to inference, and linking attacks that depend on the attacker's knowledge [7]. Additionally, there is the issue of a malicious user with access to the SP's end being able to piece together information from parts of information gathered historically and then using this knowledge to transmit false information [2], [7], [8].

Other proposals avoid this occurrence by using design approaches based on cryptography to protect the data from being accessed either by malicious users or by the third party SPs to whom the data is outsourced. Research on tackling the issue of access control to outsourced data with cryptography began with the approach that Miklau et al. [8] proposed in 2003. In the Miklau et al. approach, different cryptographic keys are used to encrypt different portions of the XML tree by introducing special metadata nodes in the document's structure. In this way, the data remains secret to all the participants in the system and only those in possession of valid keys are able to decrypt and read meaningful information from the data. However, while this solves the problem of securing the outsourced data, and additionally imposes some form of hierarchical access control, it does not address the problem of handling key updates and more crucially the need of categorizing the data that the SP receives. As well, the problem of illegal information flow or exchanges between users is not addressed.

De Capitani Di Vimercati et al. build on this approach and specifically consider the problem of authorization policy updates which is, in a situation where access control is supported cryptographically, equivalent to the problem of key updates [2], [7]. The De Capitani Di Vimercati approach operates by using two keys. The first key is generated by the data owner and used to protect the data initially by encrypting the data before it is transmitted to the SP. Depending on the authorization policies the SP creates a second key that is used to selectively encrypt portions of the data to reflect policy modifications. The combination of the two layers provides an efficient and robust solution to the problem of providing data security in outsourced data environments. However, as with previous solutions this solution does not provide a method of controlling data exchanges between users once the information has been retrieved from the SP's end.

The literature on cryptographic access control approaches to addressing the problem of secure data access and cost effective key management has been investigated in the context of distributed environments like the Internet [9]–[13]. Examples of applications in which access can be controlled using these more conventional cryptographic key management approaches to enforcing access control hierarchically include, pay-tv, sensor networks and social networking environments [14]–[16]. However, the case of controlling illegal exchanges of outsourced data in an untrustworthy scenario, differs from these cases in the sense that the SP cannot be trusted to enforce access control policies in the manner that the data owners specify. Moreover, because of the sensitivity of the data, it is important that users are not able to share the data with unauthorized users in order to forestall cases of collusion between users and SPs. Using cryptographically supported access control has the advantage that the SP can categorize the data received and if the SP is compromised, data confidentiality is maintained if users do not reveal their keys to the malicious party. In the next section, we present our framework for monitoring information flows between users and/or services to guarantee outsourced data privacy.

III. A Framework for Secure Sharing of Outsourced Data

We propose a conceptual framework for monitoring the flow of outsourced data in a service oriented architecture (SOA) where data owners transfer the management of their data to a third-party SP. In our SOA, the SP handling the outsourced data offers a service that can be invoked to retrieve parts of the data that is received from the data owner(s). Invocations of the outsourced data service can either be made directly by users or via services that make the invocation on behalf of a user. Information flow control is important in this context to prevent either the users or services from sharing data in ways that are not authorized by the security policies. Although the SOA operates in an environment like the Internet where trust is a concern, we assume that the users invoking the services belong to a trusted group of users and have clearly defined roles with associated permissions. Our conceptual framework consists of three principal components that, as shown in Fig. 2, interact with two knowledge bases to decide whether or not to authorize the transfer of data between users and/or services.

The knowledge bases in this case, the *ACM and Key Graph* repository as well as the *Hash Value* repository, allow the Access Control (AC) module, Information Flow Control (IFC) module, and the Security Policy Module (SPM) to check that the rules governing access to the data are verified. Both knowledge bases and the SPM reside at the data owner's end while the AC module resides at the SP's end and the IFC module resides at the user's /user invoked service's end. Each of the components can be implemented as a service that can be invoked according to the task the needs to be performed. In the following sections we describe how each component can be designed and integrated into a service oriented architecture in order to manage and protect the privacy of outsourced data.

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

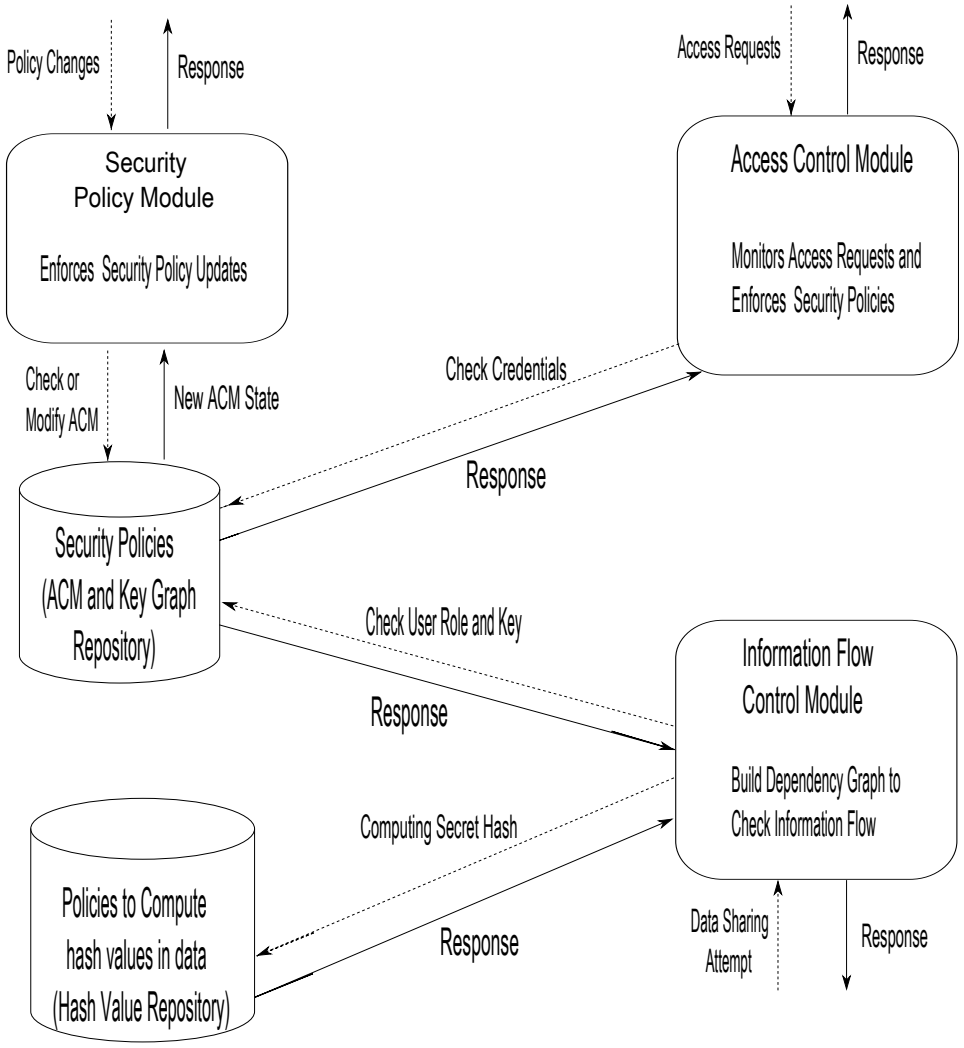

Fig. 2. Framework for Information Flow Control of Outsourced Data

A. *Security Policy Module (SPM)*

We assume that we are dealing with a two layered encryption scheme such as the one that Vimercati et al. [1], [2] have proposed for protecting outsourced data privacy. In this case, each user holds two keys that are used to first remove the encryption layer imposed by the SP and second to decrypt the data into a readable format. The SPM basically enforces the security policies that the data owner defines before the data is transfered for management to the SP. These security policies are defined by assigning authorized users to roles according to the permissions the users are granted with respect to accessing the data. For simplicity, we assume that our security policies are based on an role based access control (RBAC) model in which roles are defined in the form of a hierarchy. The permissions associated with a role are represented using an access control matrix (ACM) where the rows indicate the role and the column the data objects.

As shown in Fig. 3, in the ACM, a 1 at the intersection of a row and column indicates that a user with the corresponding role has the permission to access the data, while a 0 indicates otherwise. So, for instance (see Fig. 3), a user with the role R_0 would have the permission to access the data objects B, D and E while a user with the role R_3 can only access the data objects D and E.

	A	B	C	D	E
R_0	0	1	0	1	1
R_1	1	0	1	0	0
R_2	1	1	1	1	1
R_3	0	0	0	1	1

Fig. 3. Access Control Matrix

When the RBAC model is supported with cryptography, each one of the data objects is encrypted with a unique key that is shared with the users requiring access to the data. Holding

several keys is cumbersome in terms of security management and so it makes sense to use a hierarchy of inter-dependent keys where users are assigned a single key from which all the required keys can be derived. In order to do this, we structure the roles hierarchically as shown in Fig. 4 with the role that has the most permissions being at the top of the hierarchy and the role with the least permissions at the bottom of the hierarchy. As shown in Fig. 4, R_2 has the most permissions because it allows a user access all of the data on the system, whereas R_1 and R_3 have the least permissions.

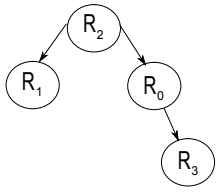

Fig. 4. An Example of a Role Hierarchy

The role hierarchy can then be expressed as a key graph where keys are assigned to a role according to the parts of the data that the user with the associated role can access. Cryptographic keys can then be generated to enforce the security policy by creating security classes to which users are assigned according to the role(s) they have with respect to accessing the data. The generated keys are then mapped onto a key graph so that a user receives only the keys that he/she requires according to the role(s) he/she has on the system. Although, in this case a user can get assigned several keys, it is possible to minimize the number of keys assigned by using a key management scheme, such as the one that Atallah et al. [17] proposed, in which all the required sub keys can be derived from a master or composite key.

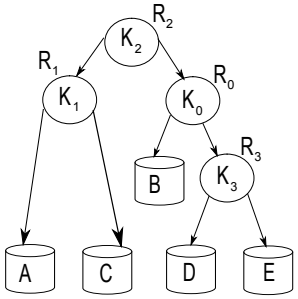

Fig. 5. Key Graph Expressing Permissions of Access to the Data

For instance, in Fig. 5 the keys K_0, K_1, K_2, and K_3 are associated with the roles R_0, R_1, R_2, and R_3 respectively. In this case, the number of keys assigned to users is minimised by using a key assignment scheme that makes it possible for a user with a key say K_2 to derive all the required keys $K_0, K_1,$ and K_3 that are required to access the data objects $A, B, C, D,$ and E. However, the key assignment algorithm does not allow a user with the key K_3, for example, to derive any of the higher level keys. For example, the key assignment scheme that Atallah et al. proposed randomly assigns keys to the nodes in the key graph and uses a function

$$Y_{R_j \preceq R_i} = K_j \oplus H\left(K_i, l_j\right)$$

where $Y_{R_j \preceq R_i}$ indicates that a user with role R_i is authorized to read data that is accessible to a user with a role R_j, $H\left(K_i, l_j\right)$ is a cryptographic hash function that is used to check that the key K_i can be used to derive K_j, l_j is a public value or label that is assigned to the node R_j, and K_i and K_j are the random keys assiigned to the nodes R_i and R_j respectively. Therefore, a user can only access a data object if the precedence rule is verified and the user's key can be used to decrypt the data into a meaningful format. All of the information in the ACM and the key graph is stored in the *ACM and Key Graph* repository.

The advantage of combining cryptography with RBAC is that it makes security policy updates easy to handle since a consistent copy of the data can be maintained on the SP's end where users can both access the data and apply updates. An example of the necessity of maintaining copy consistency, while protecting data privacy, arises in the case of medical records to which several doctors from different hospitals have access. In this case, it is important that every doctor gets a synchronized view of the patient's records to avoid situations in which a diagnosis is made with incomplete information.

B. Access Control (AC) Module

The Access Control (AC) Module basically enforces the security policies formulated both by the data owner and the SP in order to protect the outsourced data from unauthorized access. Moreover, the access control module serves as an intermediary between the user and the SP so that the contents of the data are not revealed to the SP. This module can be implemented as a service that a user invokes in order to access the outsourced data on the SP's end.

As mentioned earlier, the AC module is embedded at the SP's end and controls access to data that is doubled encrypted using the method that Vimercati et al. [2], [7] proposed. Since this involves two layers of encryption on the data, the AC module decides whether or not to grant a user or user initiated service call access to the data by analyzing each access request according to the information in the *ACM and Key Graph* repository. As shown in Fig. 2, when the AC module receives a request to access data, the AC module will begin by checking the credentials of the user against the information in the *ACM and Key Graph* repository to determine whether the role of a user matches authorizes access to the data requested. The reason for doing this is to prevent users illegally exchanging keys and performing impersonation attacks to retrieve data at the SP's end. The AC module prevents this by comparing the role of the user requesting access to the one corresponding to the node in the role hierarchy that is associated to the user's

role. A simple way of doing this, is to have some sort of secret question whose answer determines whether or not the user is who he/she claims to be. A successful credential verification, results in the AC module decrypting the data, to remove the layer of encryption imposed by the SP, and returning the data to the user.

An example of how the data is retrieved from the SP's end and returned to the user is shown in Fig. 6. In this case, a data owner transfers encrypted data A, B, C, D, and E to an SP for management. The SP double encrypts all the data received using a second set of keys a_0, a_1, a_2, and a_3 to obtain double encrypted data D_0, D_1, D_2, and D_3 that is stored at the SP's end. Finally, the SP creates an ACM in which all the rules of access are mapped to facilitate security management. The keys a_0, a_1, a_2, and a_3 are then shared with the data owner who then assigns users a set of two keys depending on the role of the user. As with the keys that the data owner generates, the SP can use a hierarchical key management scheme that assigns users single keys from which the required keys can be derived. So, as shown in Fig. 6 a user with the role R_2 will hold two keys a_2 and K_2 where a_2 can be used to derive the keys a_0, a_1, and a_3 and likewise K_2 can be used to derive the keys K_0, K_1, and K_3. In order to access the data object A, the user with the role R_2 can use the key a_2 to derive a_1 in order to decrypt D_1 to remove the layer of encryption imposed by the SP and download the data object A that is decrypted with the key K_1, derived from K_2, to read the data. We now consider what happens when a user retrieves data from the SP's end and proceeds to share it either with an authorized user or even the SP.

C. Information flow Control Module

The Information Flow Control Module (IFCM) uses the concept of dependence graphs to prevent data exchanges between unauthorized users. Embedded in each data object is a cryptographic tag (like an invisible digital watermark) that is computed by combining the checksum of the data and the cryptographic key with which the data is initially encrypted (at the data owner's end) to obtain a cryptographic hash. This hash value is then embedded into the data like a digital watermark and used to authenticate a user before access is granted to the decrypted data.

When a service call is initiated the IFCM will construct a dependence graph to determine whether any illegal information flows are likely to occur. The dependence graph is constructed by using the key graph to decide which users are allowed to share information. Users can share information when they belong in the same group and have a role that matches the one associated with the key that was used to encrypt the data. When the user has a role that is higher than the role of the user initiating the data sharing, the role of the user initiating the sharing must be one of the sub-roles of the higher role. As well, the user with the higher role must either hold the key required to decrypt the data or be able to derive the key from the one in his/her possession. This allows the IFCM to determine which users can share information and what information they can share.

As a second line of defense, embedded in each data object is a cryptographic hash that is computed by combining the checksum of the data and the cryptographic key with which the data was encrypted. This hash value allows the IFCM module to prevent users from creating covert channels to transfer data illegally to unauthorized users. In order to do this, each execution involving a data object is evaluated by comparing the user's role with the key presented to determine if the key presented belongs to the user making the access request. When this is the case, the IFCM will proceed to crosscheck the user's authorization to view the requested data by using the key presented to compute the cryptographic has associated with the data. Access is allowed if a "correct" hash is computed.

We now consider the case of a user attempting to share information that has been retrieved from the SP and decrypted. The IFCM will contact the knowledge base to compare the role of the user, on the receiving end, to the one that is associated with the key that was used to compute the cryptographic hash on the data being transfered. The comparison allows the IFCM to determine whether or not the data can be shared with the requesting user. When there is a mismatch between the requesting user's role and the key on the data, the data transfer will not be allowed to proceed.

IV. Example Scenario

As an example we consider the case of hospitals that outsource the management of their patients'data to a third-party SP in order to ensure that it can be managed in a cost effective way. Outsourcing the patients' data makes the data easily accessible via the Internet so, in an emergency a medical practitioner would only need a patient's name and health card number to get an up to date record of the patient's data.

In order to control access to the patient's data, each hospital applies a first encryption layer to their data and transfers the management of the data to an SP where a second encryption layer is applied. The keys used for the second encryption are then returned to the data owner who will assign every health care practitioner, on the board of accredited medical practitioners, a pair of keys according to their role. Access is then possible if a health care practitioner holds the keys needed to access a patient's file. Since there might be potentially many key pairs, as mentioned in Section III(C), it makes sense to use a key management scheme that assigns users a composite key. To do this, a tool for combining keys to form a composite key is embedded at the user's end. Using this tool, all of the keys the user receives from the different data owners are combined to form a composite key that can be used for all the required data accesses.

As well, in order to ensure data consistency, it makes sense to keep a single record of the data at the SP's end rather than have users download copies that are modified separately. For instance, in an emergency unit a medical practitioner might need to access a patient's records to determine if the patient has any allergies that need to be considered before an

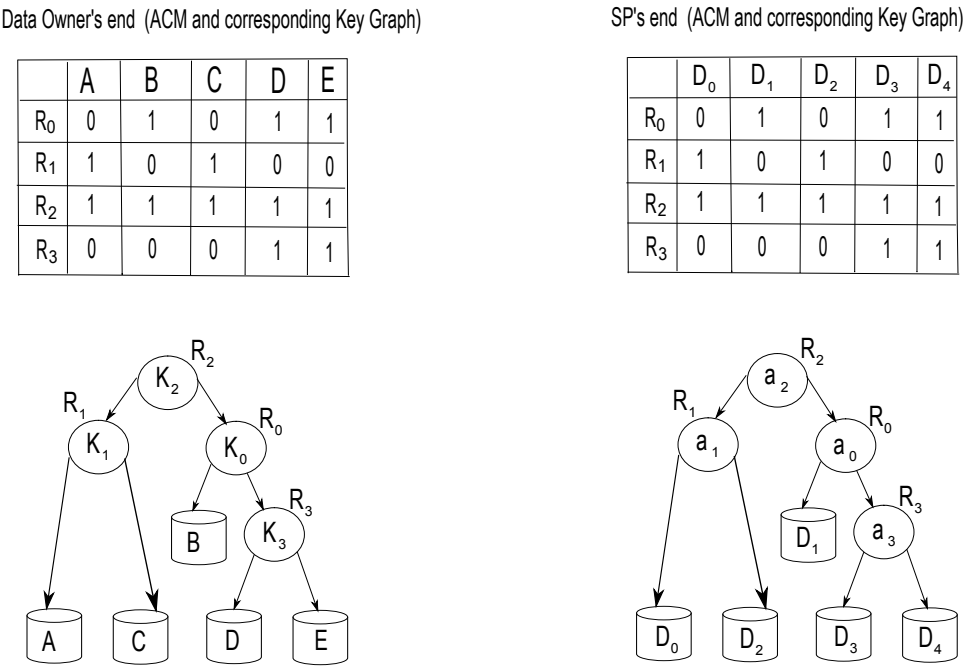

Fig. 6. Secure Management of Outsourced Data: Example

operation. If the allergy information is in a separate record on the lab technician's computer, the medical practitioner may not find out about the patient's allergies in time to carry out the operation. Therefore having a consistent record on the SP's end is important.

Using cryptographic access control to protect data like this on the SP's end is necessary if the data owners (in this case the hospitals) are to prevent the SP's from reading the data. Additionally, since sharing patient data is prohibited by privacy laws, the hospitals need to guarantee that the data is never shared with unauthorized users. The Access Control and Information Flow Control modules in the framework we have described (see Fig. 2) handle both concerns. As mentioned earlier, the Access Control Module checks user credentials to determine if they have the authorization to view the data and only releases the information if this is the case. The Information Flow Control module on the other hand monitors and prevents attempts to share the data by using a combination of dependence graphs and cryptographic hash values to decide whether or not a data object can be shared between any two parties.

V. Conclusion

We have presented a framework based on the concept of dependence graphs to prevent unauthorized sharing of outsourced data. Dependence graphs and program slicing are popular approaches to monitoring the flow information with in programs in order to prevent illegal assignments [18]. Specifically we considered the case of handling data exchanges in a scenario in which access to the outsourced data is controlled cryptographically [2]. Our discussion of the background on this topic indicated that most solutions have tended to focus on indexing techniques that strike a balance between query efficiency and controlling exposure to inference attacks as well as handling security policy updates efficiently. However, we noted that a key concern in using outsourced data in service oriented architectures on the Internet is that of preventing authorized users from sharing the data with unauthorized users. Data owners need some way of guaranteeing that data privacy will be maintained and SPs need a way of maintaining the data owners'trust.

Our framework consists of three modules namely, the Security Policy Module, the Access Control Module and the Information Flow Control Module. All three modules can be implemented as services that can be invoked separately. The Security Policy Module is implemented as a two separate knowledge bases, one on the data owner's end and one on the SP's end. The knowledge bases contain the combination of rules that implement the security policy and these rules are expressed using an RBAC model. In the RBAC model, users are assigned roles according to the permissions they have with respect to accessing the data. RBAC and cryptography are combined generating keys that are mapped onto a key graph that reflects the role assignments. This key graph is then used by the Access Control Module, that is located at the SP's end, to evaluate access requests based on the role of the user making the request and the key presented.

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

The Information Flow Control Module, essentially acts to prevent unauthorized flows of data. This is done by monitoring information exchanges between users and services using a combination of cryptographic tags, that are embedded into the data, and dependence graphs. The dependence graphs allow the Information Flow Control Module to determine whether or not a exchange of data between the two services is allowed and the cryptographic tags act as a second line of defense allowing a user to only read the data if their profile matches the key presented. Therefore, even if a user were to illegally obtain the key with which the data was encrypted, access would not be possible because the Information Flow Control Module would be unable to match the user's profile, in the knowledge bases, to the key that is presented to compute the cryptographic tag in the data.

As future work, we plan on designing and implementing all three modules in a service oriented architecture that involves outsourced data. Our first step will be to design an intelligent knowledge base that is able to make predictions about changes in security policies and adapt the role hierarchy as well as the key graph to reflect the change. An example of this arises during service compositions where services with different security requirements need to interact securely. In this case, establishing a global security policy that satisfies the minimum security requirements of the individual services is challenging both from the access control and information flow control perspective. A second step would be to develop the access control framework by designing an access control scheme that is able to handle security policy compositions efficiently. The access control policy needs to be flexible so that security policy updates can be handled without requiring data encryptions that are costly for large volumes of data. Finally, the Information Flow Control Module, needs to be formalized, implemented, and proven to be secure against illegal data exchanges when security conditions change. Finally, since trust is a concern for systems operating on the Internet, a method of enforcing a trust is needed to determine whether or not a user is who he/she claims to be.

REFERENCES

[1] S. De Capitani Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samariti, "Over-encryption: Management of access control evolution on outsourced data," in *In Proc. VLDB 2007*. Vienna, Austria, September 23-28 2007, pp. 123–134.

[2] S. De Capitani Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samariti, "A data outsourcing architecture combining cryptography and access control," in *In Proc. of the 2007 ACM Workshop on Computer Communications Security (CSAW)*. Fairfax, Virginia, USA, November 2 2007, pp. 63–69.

[3] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Encrypting sql over encrypted data in the database-service-provider model," in *Proc. of ACM SIGMOD*. Madison, Wisconsin, USA, June 2002, pp. 216–227.

[4] H. Hacigumus, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proc. of the 18th ICDE Conf.* San Jose, California, USA, 2002, pp. 29–38.

[5] G. Aggrawal, M. Bawa, P. Ganesan, H. Garcia-Molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep a secret: A distributed architecture for secure database services," in *Proc. CIDR 2005*. Asillomar, California, USA, 2005, pp. 186–199.

[6] R. Agrawal, J. Kierman, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proc. of ACM SIGMOD*. Paris, France, June 2004, pp. 563–574.

[7] A. Ceselli, E. Damiani, and S. De Capitani Di Vimercati, "Modeling and assessing inference exposure in encrypted databases," *ACM Trans. on Information and System Security*, vol. 8, no. 1, pp. 119–152, February 2005.

[8] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *Proc. of the 29th VLDB Conf.* Berlin, Germany, September 2003, pp. 898–909.

[9] M. Atallah, K. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in *In Proc. ACM Conference on Computer and Communications Security*. Alexandria, Virginia, USA, November 7-11 2005, pp. 190–202.

[10] M. Atallah, M. Blanton, and K. Frikken, "Key management for non-tree access hierarchies," in *In Proc. of ACM Symposium on Access Control Models and Technologies*. Lake Tahoe, California, USA, June 7-9 2006, pp. 11–18.

[11] J. Crampton, "Cryptographically-enforced hierarchical access control with multiple keys," in *In Proc. of the 12th Nordic Workshop on Secure IT Systems (NordSec 2007)*, 2007, pp. 49–60.

[12] R. Hassen, A. Bouabaallah, H. Bettahar, and Y. Challal, "Key management for content access control in a hierarchy," *Computer Networks*, vol. 1, no. 51, pp. 3197–3219, 2007.

[13] A. Kayem, S. Akl, and P. Martin, "On replacing cryptographic keys in hierarchical key management systems," *Journal of Computer Security*, vol. 16, no. 3, pp. 289–309, 2008.

[14] J. Birget, X. Zou, G. Noubir, and B. Ramamurthy, "Hierarchy-based access control in distributed environments," in *In Proc. of the IEEE International Conference on Communications, Vol. 1*. Helsinki, Finland, June 11-14 2001, pp. 229–233.

[15] W. Yu, Y. L. Sun, and K. J. R. Liu, "Optimizing rekeying cost for contributory group key agreement schemes," *IEEE Trans. Dependable Secur. Comput.*, vol. 4, no. 3, pp. 228–242, 2007.

[16] S. Zhu, S. Setia, and S. Jajodia, "Performance optimizations for group key management schemes," in *In Proc. of* 23^{rd} *International Conference on Distributed Computing Systems (ICDCS '03*. Fairfax, Virginia, USA, May 19-22 2003, pp. 163–171.

[17] M. Atallah, M. Blanton, N. Fazio, and K. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 3, pp. 1–43, 2009.

[18] C. Hammer and G. Snelting, "Flow-sensitive, context-sensitive, and object-sensitive information flow control based on program dependence graphs," *Int. J. Inf. Secur.*, vol. 8, no. 6, pp. 399–422, 2009.

978-1-4244-5494-5/10/$26.00 ©2010 IEEE