December 2004

Sharing and Access Right Delegation for Confidential Documents: A Practical Solution

S.M. Yiu
University of Hong Kong

S.W. Yiu
Hydra Limited, Wanchai, Hong Kong

L. Lee
University of Hong Kong

Eric Li
University of Hong Kong

Michael Yip
University of Hong Kong

Sharing and Access Right Delegation for Confidential Documents: A Practical Solution

S. M. Yiu [1]   S. W. Yiu [2]   L. K. Lee [1]   Eric K. Y. Li [1]   Michael C. L. Yip [1]

[1] Department of Computer Science
The University of Hong Kong
Pokfulam Road
Hong Kong
Email: {smyiu, lklee, kyeli, clyip2}@cs.hku.hk

[2] Hydra Limited
Wanchai
Hong Kong
Email: ryiu@hydra.com.hk

Abstract

This paper addresses a practical problem in Document Management Systems for which no existing solution is available in the market. To store confidential documents in a Document Management System, a common approach is to keep only the encrypted version of the documents to ensure the confidentiality of the contents. However, how to share these encrypted documents and how to delegate the access rights of these documents are not straightforward while these operations are common in most companies. In this paper, we discuss the issues related to this problem and provide a practical and easy-to-implement solution for solving the problem. Our solution has been shown to be feasible by a prototype implementation. We also show how to extend our solution to take advantage of the hierarchical structure of a company to make it more scalable. As a remark, this problem was initiated by a local company based on its actual needs to fulfil a set of business objectives and requirements.

Keywords

Document management systems, access right delegation, encrypted documents, public-key infrastructure

1. INTRODUCTION

Document Management Systems (e.g. Bertino 1990) are Information Systems specially designed for handling and managing documents. In these systems, documents are usually stored in a database for ease and efficient retrieval. This kind of systems is getting popular as it helps to save storage space for paper records and can enhance the communication and dissemination of information. In real applications, the management of these documents is complicated by practical requirements. Not all documents are accessible to everyone in the company. Some are strictly confidential and extra security measures must be taken to protect the contents of these documents. On the other hand, sharing of these documents is unavoidable and in most companies, it is also common for a manager to delegate his/her access rights of these documents to a subordinate to help handling the documents. Consider the following scenario. In a company, let *D* be a confidential document (e.g. a bid document) that should only be accessible by some senior staff, say Bob, Mary, and John. However, John is having a business trip and would like to delegate his access right to his subordinate May for handling the document. Of course, later on, when John returns, he would like to revoke May's access right to *D*. It is not straightforward how the sharing as well as the delegation of access rights can be done while keeping the contents of the documents as secure as possible.

At a first glance, this problem is related to access control, one of the common techniques is to make use of an Access Control List (ACL) (Barkley 1997, Li et al. 2002, Qian 2001, Stiegler 1979) which states clearly which user is allowed to access which resources (documents) together with the allowable operations (e.g. read, write, delete) in the form of a data structure (e.g. a table). Whenever a user wants to access a document, the system will check against the ACL and see if the user is granted the access. In fact, using ACL, the sharing, delegation, and revocation can be done easily by adding and removing the corresponding user in the ACL. In our example, the ACL will probably contain four entries for the document *D* for Bob, Mary, John, and May. When John comes back from his trip, the entry of May for *D* will be deleted. However, from the security point of view, ACL alone does not provide a satisfactory solution to the problem. The reasons are as follow. If only the plain version of a document is stored in the database, the attackers may be able to get hold of the document if the attackers can break in the system. In fact, if the document is transmitted through the network, the attacker can easily obtain a copy of the document without breaking into the database. For highly confidential documents, we certainly need another level of security. Also, the ACL is usually manipulated by the database administrator who has the full control and access right to all documents in the database. So, the security of the system relies on a single person and it violates the principle of security that separation of duties should be implemented to avoid collusion of users. To conclude, the use of ACL alone does not solve the problem.

On the other hand, to ensure the confidentiality of the contents of the documents, one can use encryption. That is, we only store the encrypted version of the documents so that even if the attackers are able to get hold of the documents, it is difficult for them to understand the content without the knowledge of the corresponding decryption key. To perform encryption, one can make use of the Public Key Infrastructure (PKI). For more information about PKI, see Kelm (2000), Kent and Polk (1995). In PKI, each entity has a public and private key pair. The private key is supposed to be kept secret by the user and the public key will be authenticated by a trusted certificate authority (CA). To encrypt a document for a user, say Bob, we can use the public key of Bob to do the encryption and the encrypted document can only be decrypted using Bob's private key. However, using encryption alone also has problems. For examples, in our case, we may need to encrypt the document *D* four times, each with the public key of Bob, Mary, John, and May respectively. We have to store four different encrypted versions of the document. It is obvious that the solution is not scalable and will waste a lot of storage space. Also, when compared to symmetric key encryption and decryption in which the same key is used for encryption and decryption (e.g. Data Encryption Standard (DES) and Advanced Encryption Standard (AES), see NIST (1999, 2001)), public-key encryption and decryption are slow and may not be appropriate for daily operations. Note also that a pure symmetric encryption approach is also not appropriate as the key distribution and management process will be rather complicated.

Our Contributions: In this paper, we combine the techniques of ACL and encryption to provide a feasible solution for solving the problem of sharing and access right delegation for confidential documents. Our solution provides two levels of security to the system and divides the responsibility to two parties, the database administrator (DBA) and the security officer (SO). To speed up the encryption and decryption process, we make use of the concept of "session keys" such that the same document is only encrypted once based on a symmetric key encryption algorithm using the session key. For those legitimate users who can access the document, we encrypt the corresponding session key using their public keys. In other words, a user must have the appropriate private key to decrypt the session key and must be granted access to the document in the ACL before he/she can actually access the document. In this way, separation of duties is achieved; namely, the DBA can only manipulate the ACL, but is not allowed to touch any of the session keys while the SO is responsible for generating the session keys, but is not allowed to touch the ACL. So, unless the two parties collude, either one cannot break the system and gain access to the plain version of the documents.

The rest of the paper is organized as follows. Section 2 highlights the major requirements for the access right delegation and sharing of the confidential documents. These requirements are based on the actual requirements from a local company. Our proposed solution is given in Section 3. We show in Section 4 how our basic scheme can be modified to take advantage of the hierarchical structure of a company to make it more scalable for large-sized companies. Section 5 concludes and discusses some of the possible future work.

Remarks: There is no existing solution in the market that can solve the problem. Existing database and software packages only provide encryption functions and allow users to store an encrypted version of the document in the database. However, how we can share the documents and how access right delegation can be performed to satisfy our requirements (see Section 2) are not provided and discussed. On the other hand, there are other works on delegation (e.g. Gasser and McDermott 1990, Mambo et al. 1996a, Mambo et al. 1996b, Varadharajan et al. 1991, Ruan and Varadharajan 2004). However, it is not trivial how these approaches can be applied in our case as most of them focus on the delegation of a signing right only. In fact, most of these schemes are complicated and the practicality is still an open question. So, it is desirable to have a more practical and easy to implement solution.

2. REQUIREMENTS

In this section, we list the assumptions and the requirements related to the sharing and access right delegation of a set of confidential documents. The list is extracted from the requirement specifications of a local company. Before we give the details of the requirements, we first distinguish two concepts in our application: delegation and access right granting. In the absence of encryption requirements, delegation can simply be achieved via access right granting. In the case of encrypted document delegation, as in our application, a delegatee needs to be able to have both the proper access right and decryption key in order to get access to the delegated document.

There are three major assumptions for this system. Firstly, each user and each document in the system are assigned a security level. In general, if the security level of a user is lower than that of the document, the user is not allowed to access the document unless the user was delegated to do so (see the requirements below for details). For simplicity, in this paper, we only assume that there are two levels of security: confidential and unclassified in which the unclassified documents can be accessed by every user. In real applications, we can have more levels.

Secondly, each user has generated a public key and a private key based on PKI. The public key is stored in a certificate signed by the Certificate Authority (CA) and is accessible by everyone through a database. The private key is supposed to be stored securely by the user, for example, it can be stored in a smart card or other hardware tokens.

Thirdly, there are three types of access rights for the documents: read, update, and delete. In the real applications, we have two more types: downgrade and upgrade which modify the security levels of the documents. The owner (creator) of the document is automatically assigned all three types of access rights to the document.

Next, we present the list of requirements and we mainly focus on the requirements related to confidential documents.

Requirements

- Confidential documents must be stored and transmitted in encrypted form.
- The same (confidential) document can be accessible by multiple users [*].
- The system should support delegation of access rights at the *Global level* and the *Document* level. To make sure that delegation is done in a restricted manner, for each user (the delegator), a list of pre-defined delegatee(s) is set. Delegation can only be done for pre-defined delegatee(s). Any changes in the list of pre-defined delegatee(s) must go through a dedicated procedure. In fact, in our application, this delegation requirement can be used to support the "acting" operation which is common in major companies.
    - *Global level delegation*: The delegator can delegate all or selected access rights (read, update, and delete) to his/her delegatee(s). The delegation will be applied to all documents accessible by the delegator including those documents created in the future. However, the delegatee can only access the documents with a security level not greater than that of the delegatee. This requirement is to fit a real case scenario in which a manager usually delegates his/her secretary to handle most of the documents.
    - *Document level delegation*: The delegatee can be granted access rights on a document basis even if the document has a higher security level than that of the delegatee.
- Delegation can be transferable. For example, if Bob delegates his access rights to John, then John can further delegates his access rights to Mary. In other words, if there is a document *D* which is only accessible by Bob, but not accessible by both John and Mary. After the delegation, both John and Mary can access *D*.
- Delegation is revocable. The revocation will be done through the whole delegation tree. In other words, if Bob has delegated his right to John and John has further delegated his right to Mary, then when Bob revokes the delegation to John, the delegation to Mary should be revoked as well.

[*] Note that the issue of concurrent access, e.g. the read write conflict, is out of the scope of this paper and can be handled by standard techniques and the underlying database management system. Our main focus is to derive a scheme so that the same encrypted document is accessible by multiple users without creating different copies of the encrypted document.

- The system should allow user A to assign access rights of a document to multiple users provided that user A has the corresponding access rights (not by delegation) and the other users have a higher or equal security level as that of the document.

Note that not all requirements are discussed in the paper. We only highlight the critical requirements that are related to the design of our solution. For example, the requirement that multiple versions of documents should be supported is omitted in our discussion. In fact, we can modify the scheme presented in the paper to satisfy these additional requirements, however, the details are out of the scope of this paper.

3. OUR PROPOSED SOLUTION

In this section, we describe our proposed solution. We will first highlight the key features of our solution, then will provide details on the database design and the procedures.

Session key encryption for documents: To avoid encrypt the same document more than once, we make use of the technique of session key. For each document, we generate a session key and encrypt the document using the session key. For each user who is allowed to access the document, we encrypt the session key using the public key of the user. In this case, only the user who has the corresponding private key can extract the session key in order to read the document.

Access control table and delegation table: We make use of two tables in the database to store the access control list and the delegation information. These two tables control the access to the documents in the database. Together with the private key of the user, we have two levels of security.

Separation of duties: In our solution, there are two independent entities involved, the database administrator (DBA) and the security officer (SO). DBA has the full control and access to the access control table and delegation table, however, DBA is not given the session key to any of the documents. In other words, DBA alone cannot read the contents of the documents. On the other hand, SO keeps the encrypted session keys for all documents. However, SO does not have an access entry in both the access control and delegation tables. In other words, either DBA or SO alone is not able to read the contents of the documents.

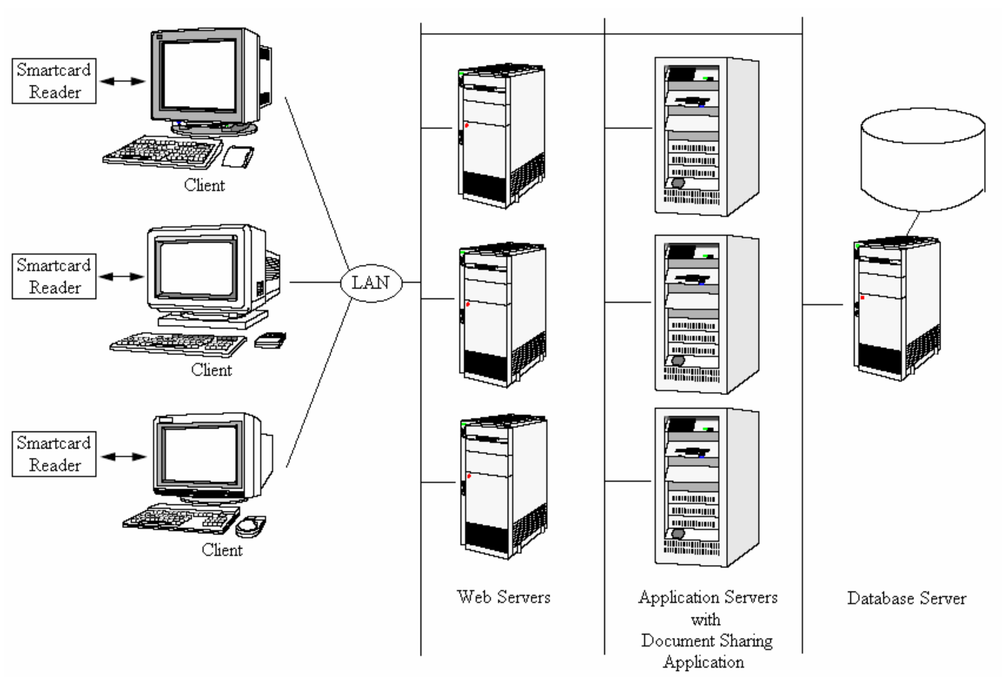


Figure 1: System Architecture

3.1 DETAILS

Figure 1 shows the system architecture of the design. Users access the system using a web browser. Since we have to make sure that only the encrypted version of the documents are transmitted over the network, so the encryption and decryption are done on the client side. The client side carries out the necessary cryptographic operations through a Java Crypto Engine. On the other hand, the application servers work with database server that keeps the access control table and the delegation table for the following functions: authentication, authorization, and document handling. Note that there is an administrator console for the security officer to perform functions specific to SO which is not discussed in details in this paper.

The system keeps two tables for controlling access to the documents in the database: the access control table and the delegation table. The structures of these two database tables are shown in Figure 2. For the access control table, if Bob has access right to the document *D*, then there is a record for Bob and *D* in the table. On the other hand, for the delegation table, if Peter is a pre-defined delegatee of Bob, then there will be a record for Peter and Bob in the table. For global level delegation, we can set the Doc_ID to some pre-defined value, say –1, and the delegated access rights will be stored in the same record. When the system starts, records for pre-defined delegatees are created in the delegation table with all access rights set to "No". For document level delegation, a separate record with the related ID of the document will be created in the table after the delegation has been performed.

Access Control Table

Field Name	Data Type	Description
A_ID	Number	Primary key
Doc_ID	Number	Document unique ID
User_ID	Number	User unique ID
read	Yes/No	Grant read access?
upd	Yes/No	Grant update access?
del	Yes/No	Grant delete access?

Delegation Table

Field Name	Data Type	Description
D_ID	Number	Primary key
Valid_from	Date	Starting date of delegation
Valid_to	Date	Ending date of delegation
Delegator_id	Number	Delegator ID
Delegatee_id	Number	Delegatee ID
read	Yes/No	Grant read access?
upd	Yes/No	Grant update access?
del	Yes/No	Grant delete access?
Doc_ID	Number	Document ID (for document level delegation)

Figure 2: The Access Control Table and the Delegation Table

User Bob creates a document *D*: On the client side, a session key is generated. The document is encrypted by the session key using a symmetric key encryption algorithm (e.g. triple DES). Access rights are granted to the appropriate users. The corresponding entries will be created in the access control table. For these users (including SO and all the pre-defined delegatees of these users), we encrypt the session key using their public keys. These encrypted session keys will also be stored in the database. Note that SO will have the encrypted versions of all session keys. Also, all session keys will not exist as plain text in the system.

User Bob grants an access right of document *D* to the user Peter: Recall that this operation can only be done if Bob has the corresponding access right to document *D* and Peter has a security level greater than or equal to that of *D*. An entry for Peter and *D* with the appropriate access right will be created in the access control table. The session key for *D* is decrypted using Bob's private key and the session key is encrypted by the public keys of Peter and all his per-defined delegatees.

User Bob delegates an access right of document *D* to the user Peter (Document level delegation): A new entry for Peter and *D* with the appropriate access right is created in the delegation table. The session key for *D* will be encrypted using Peter's public key and the public keys of all Peter's pre-defined delegatees.

User Bob delegates globally an access right to the user Peter (Global level delegation): Note that in this case, Peter must be a pre-defined delegatee of Bob. The entry for Bob and Peter in the delegation table will be updated. The session keys for all documents that are accessible by Bob will be encrypted using Peter's public key. Note that we can have a trade off between the security and the efficiency of the system by creating the encrypted session keys of all documents accessible by Bob for Peter when the system starts.

To check if the user Bob can access document *D*: We first consult the access control table and see if there is an entry for Bob and *D* with the appropriate access right. Otherwise, we check the delegation table and see if Bob is delegated an appropriate access right to *D*. If yes, the encrypted version of *D* can be sent to Bob. Recall that Bob can only read the content of *D* if Bob can provide his private key to decrypt the document. The decryption is done in the client side so that the transmitted document is in its encrypted form.

User Bob revokes the delegated access right from the user Peter: We update the corresponding record in the delegation table. Then, for security purpose, we delete all relevant encrypted session keys. However, for efficiency purpose, we can choose to keep these encrypted session keys.

Due to the limitation of space, not all the functions are listed and discussed in details. For examples, there are functions designed for the security officer (SO) such as updating the list of pre-defined delegatees of a user. Also, there are other functions such as deleting a document that we do not discuss in the paper. However, based on the functions that we have discussed, one can see how our solution works. Note also that for messages sent from the client to the server, we can make use of digital signature or hash values to ensure the integrity.

3.2 IMPLEMENTATION

We have implemented our solution. Figure 3 shows the software architecture of our implementation. The architecture is a standard 3-tier architecture. The Web server is Apache Group HTTP Server with SSL support by OpenSSL. The Application Server is Jakarta Tomcat Server while the Database Server is Oracle 8i Enterprise Database. Jave 2 Platform, Enterprise Edition (J2EE) is used for programming the system. For the communication between the clients and the servers, Java Applets are used in the client side and JavaServer Pages (JSP) are used to present the Applets to clients. Java Servlets and JavaBeans are used in the server side for business logic and interaction with the Oracle database. The Java Cryptography Extension (JCE) is used for cryptographic operations. In order to utilize the cryptographic services, Bouncy Castle, one of the service providers suggested by Sun, is used as a service provider with the standard JCE.

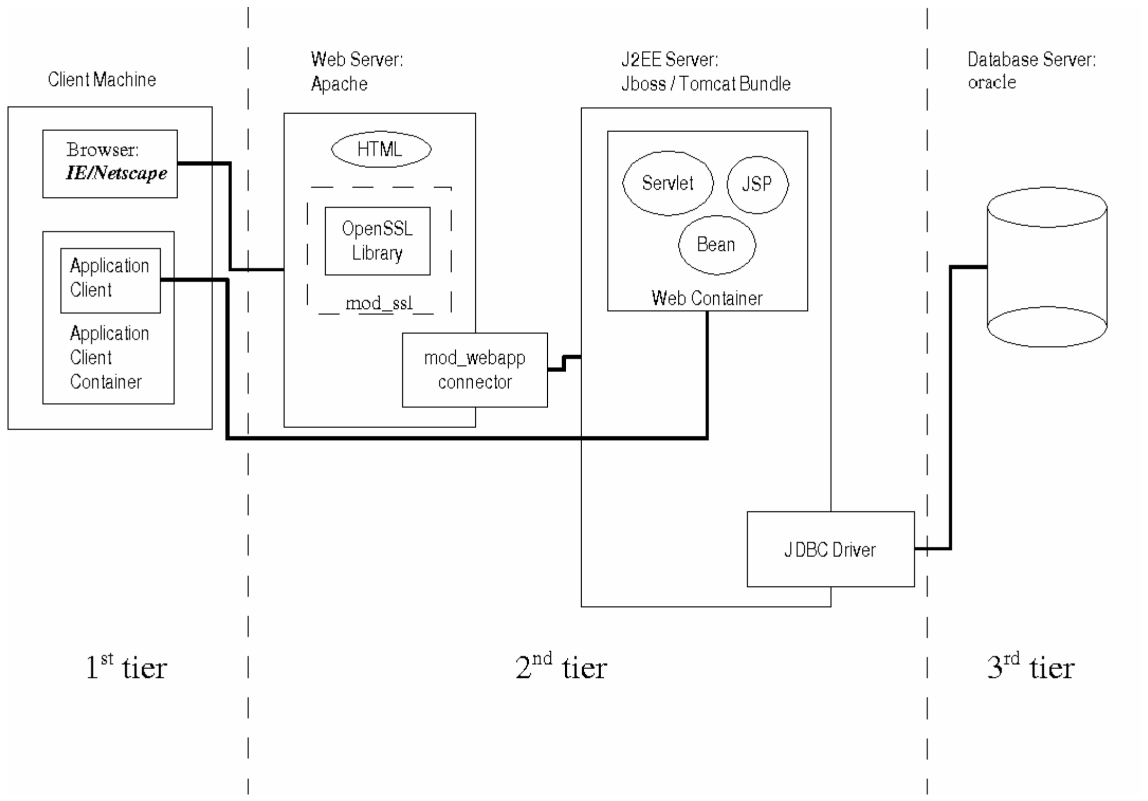
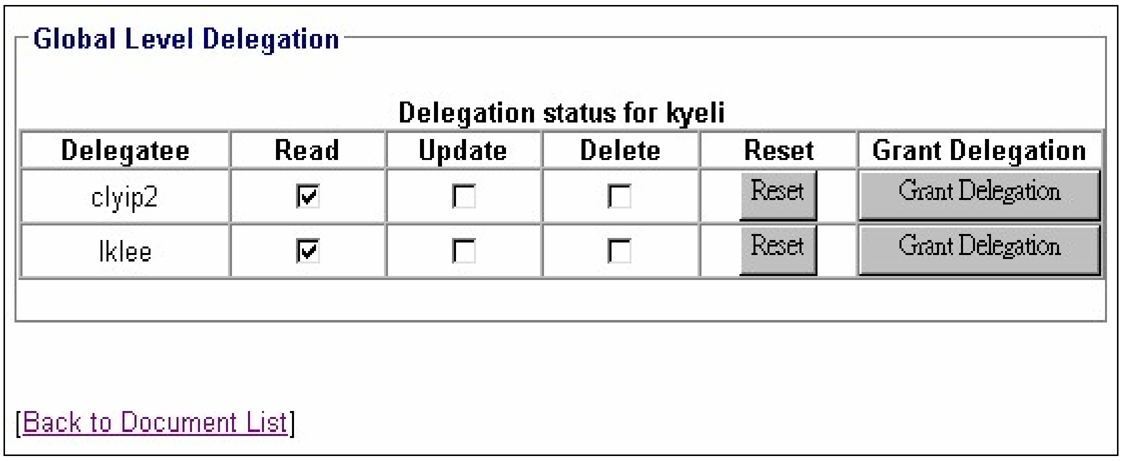


Figure 3: Overview of Software Architecture

Figures 4 and 5 show some of the screen shots of the prototype. We have created a medium-sized company with about 50 users in our prototype for testing. Our preliminary study shows that the performance of the system is good (each of the operations can be done in about 15 seconds). From the testing, we realized that if the delegation chain is long, the performance of the system degrades as the system needs to follow the chain and perform updates for each user on the chain. Fortunately, long delegation chain is not likely to occur frequently, so in general, the performance of the system is reasonable.

4. MODELING COMPANY HIERARCHY USING VIRTUAL USER CONCEPT

There is a scalability issue in the solution presented in Section 3 for company with many users. If the company is big, say with 1000 users, and there is a document that should be accessible by a majority of users, then the corresponding session key has to be encrypted hundred times and the number of encrypted session keys to be stored can be huge. In this section, we extend our basic scheme to take advantage of the hierarchical structure of a company to make it more scalable. We make use of the concept of "virtual user".



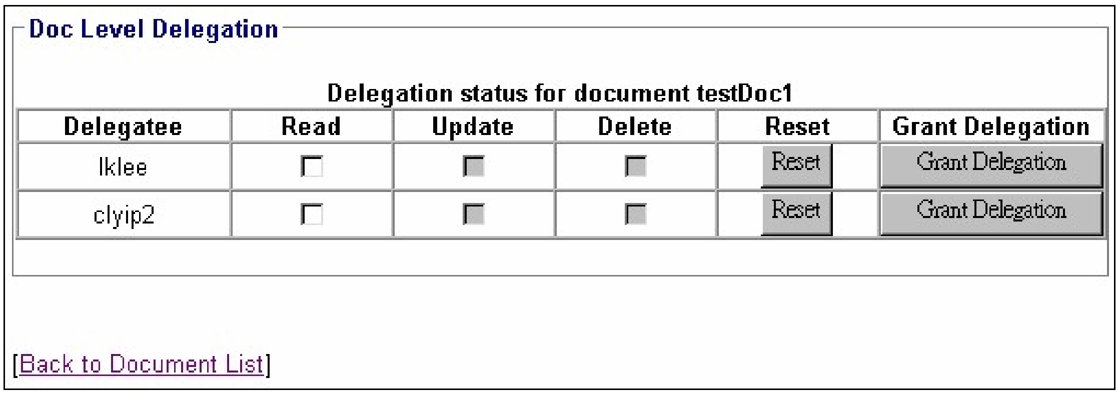
Global Level Delegation

Delegation status for kyeli

Delegatee	Read	Update	Delete	Reset	Grant Delegation
clyip2	☑	☐	☐	Reset	Grant Delegation
lklee	☑	☐	☐	Reset	Grant Delegation

[Back to Document List]

Figure 4: Delegation (Global-level)



Doc Level Delegation

Delegation status for document testDoc1

Delegatee	Read	Update	Delete	Reset	Grant Delegation
lklee	☐	☐	☐	Reset	Grant Delegation
clyip2	☐	☐	☐	Reset	Grant Delegation

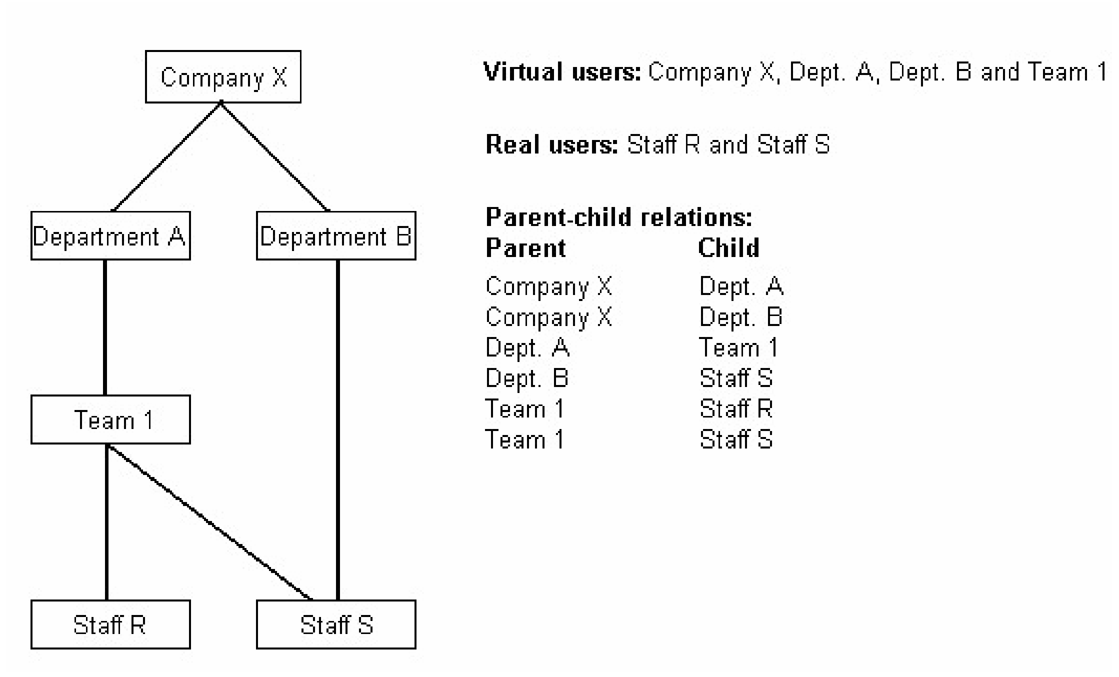
[Back to Document List]

Figure 5: Delegation (Document-level)

Figure 6 shows an example of a hierarchy of a company. We assume that the users in a company can be grouped into teams, teams can be grouped under different departments, and all departments are under the same root "Company" (see Figure 6 for an example). We call the logical units (teams, departments, company) "groups". We make use of the virtual user concept to represent each group. In other words, each group is regarded as a virtual user and will be assigned a public and private key pair. Documents can be made accessible by a group (that is, a virtual user). Conceptually, all entities under the group should be able to access the documents accessible by that group with respect to the security level of the individual entity and the document to be accessed.

The links between components in the company's hierarchical structure are modelled as parent-child relations between virtual users, and between virtual user and real user (i.e. staff of the company). These parent-child relations can be easily captured in a database table. With groups being modelled as virtual users, granting access rights of a document to a virtual user (e.g. a department) is essentially the same as granting access rights of a document to a real user (i.e. the staff in a company).

Since every virtual user would have its own public/private key pair, there is a problem in storage and management of these key pairs. Key management for the public key of virtual user is the same as that for real user (i.e. the public key certificate of each virtual user is being stored in the database so that every one in the company can have access to it.). However, for the private key of virtual user, special management approaches need to be taken. Consider again the example shown in Figure 6. If a user creates a new document *Y* and grants all access rights of document *Y* to Department *B*, then staff *S* should also have all access rights of document *Y* since staff *S* belongs to Department *B*. However, in the database, there will not be an encrypted session key of document *Y* corresponding to staff *S*, but only an encrypted session key of document *Y* corresponding to Department *B*. In other words, staff *S* needs to have the private key of Department *B* in order to decrypt document *Y*. In view of this, we want to have a scheme such that the private keys for all virtual users would be stored in an encrypted form in the database. The encrypted private key of a virtual user can be accessed and decrypted by a real user only if he/she has a “belongs-to” relationship with the virtual user in the hierarchical model. In some cases in which policies prohibits the storage of encryption key in harddisk, hardware encrypion module can be used.



Virtual users: Company X, Dept. A, Dept. B and Team 1

Real users: Staff R and Staff S

Parent-child relations:

Parent	**Child**
Company X	Dept. A
Company X	Dept. B
Dept. A	Team 1
Dept. B	Staff S
Team 1	Staff R
Team 1	Staff S

Figure 6: Modeling company hierarchy using virtual user concept

With the above key management scheme, if a real user has a “belongs-to” relationship with a virtual user (e.g. user *A* belongs to Team 1, and Team 1 belongs to Department *B*), then the real user *A* will be able to access and decrypt the encrypted private keys of the virtual users Team 1 and Department *B*, thus the real user *A* can decrypt the documents accessible by both Team 1 and Department B provided *A* has the right security level.

Our designed scheme for key management is as follows. By default, the system will always have a virtual user representing the whole company. When the Security Officer adds a group (i.e. virtual user) to the system, the group is being added as a child to one or more of the other groups in the system. For example, at the very beginning, there is only the default virtual user (i.e. the company) in the system, then when a new virtual user, say Department *A* is being added; it will automatically become the child of the company. After that, suppose another new virtual user, Team 1 is being added; now team 1 may become the child of either the company or Department *A*, or both of them. Therefore, every virtual user (except the default one) is guaranteed to have at least 1 parent. Whenever a new virtual user is being added, the public key certificate of the new virtual user will be stored in the database, whilst the private key of the new virtual user will be encrypted by the public key of the Security Officer and being stored in the database. Besides that, the encrypted private key of the parent of the new virtual user (i.e. this encrypted private key is the one being encrypted by Security Officer’s public key at the time when the parent of the new virtual user is being added to the system) will be retrieved from the database. After that, the encrypted private key of the parent will be decrypted by Security Officer’s private key, and then the private key of the parent will be encrypted by the public key of the new virtual user and stored into the database. As a result, for every virtual user (except the default one) in the system, it will have the encrypted private keys of its parents. The addition of a real user (i.e. a staff) to the system is similar to the addition of virtual user to the system, except that the private key of the real user will not be encrypted by the Security Officer’s public key and stored in the database.

5. CONCLUSIONS

In this paper, we discuss a practical security related problem in document management systems. In particular, we consider how the sharing and access right delegation of encrypted documents can be done in such systems. The problem we studied is based on the requirements from a client of a local company. There is no existing solution in the market that can solve the problem. We propose a feasible and practical solution to solve the problem. Our solution combines the technique of access control list (ACL) and session key symmetric encryption. We have implemented our solution as a prototype. The preliminary testing shows that the performance of our system is quite good even for a medium-sized company with about 50 users. To make the solution more scalable, we also show how to take advantage of the hierarchical structure of a company and introduce the concept of virtual users in order to make the solution more effective.

For future work, there are a number of possible directions. Firstly, our solution is the first step towards solving the problem, one obvious direction is to design a better and more secure scheme to solve the problem. Secondly, there are no appropriate practical indexing schemes available for indexing encrypted documents that allow efficient keyword search. Some of the related works include Boneh et al. 2004, Golle et al. 2004, and Song et al. 2000. However, it is not trivial how their proposed schemes can be used in a document management system and whether the schemes are practical and scalable.

REFERENCES

Barkley, John F. (1997) Comparing Simple Role Based Access Control Models and Access Control Lists, *Proceedings of the Second ACM Workshop on Role-Based Access Control*, 127-132, VA, USA, November.

Boneh, Dan, Crescenzo, Giovanni Di, Ostrovsky, Rafail, and Persiano, Giuseppe (2004) Public Key Encryption with Keyword Search, *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2004*, 506-522, Interlaken, Switzerland, May.

Bertino, Elisa and Montesi, Danilo (1990) Design and Development of a Document Management System for Banking Applications: An Example of Office Automation, *Proceedings of the International Conference on Database and Expert Systems Applications*, 500-507, Vienna, Austria.

Gasser, M. and McDermott, E. (1990) An Architecture for Practical Delegation in a Distributed System, *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, 20-30, CA, USA, May.

Golle, Philippe, Staddon, Jessica, Waters, Brent (2004) Secure Conjunctive Keyword Search over Encrypted Data, *Proceedings of the Second International Conference on Applied Cryptography and Network Security (ACNS 2004)*, 31-45, Yellow Mountain, China, June.

Kelm, Stefan (2000) The PKI Page, URL http://www.pki-page.org/, Accessed 06 July 2004.

Kent, S. and Polk, T. (1995) IETF Public-Key Infrastructure Working Group Charter, URL http://www.ietf.org/html.charters/pkix-charter.html, Accessed 06 July 2004.

Li, Shou-peng, Wu, Shi-zhong, and Guo, Tao (2002) The Consistency of an Access Control List, *Proceedings of the 4th International Conference on Information and Communications Security*, 367-373, Singapore, December.

Mambo, M., Usuda, K., and Okamoto, E. (1996a) Proxy Signatures: Delegation of the Power to Sign Messages. *IEICE Trans. Fundamentals*, E79-A(9), 1338-1354.

Mambo, M., Usuda, K., and Okamoto, E. (1996b) Proxy Signatures for Delegating Signing Operation, *Proceedings of the Third ACM Conference on Computer and Communications Security*, 48-57, New Delhi, India, March.

NIST (1999) The Data Encryption Standard, URL http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, Accessed 06 July 2004.

NIST (2001) The Advanced Encryption Standard, URL http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, Accessed 06 July 2004.

Qian, J. (2001) ACLA: A Framework for Access Control List (ACL) Analysis and Optimization, *Proceedings of the IFIP TC6/TC11 International Conference on Communications and Multimedia Security Issues*, Darmstadt, Germany, May.

Ruan, Chun and Varadharajan, Vijay (2004) A Weighted Graph Approach to Authorization Delegation and Conflict Resolution, *Proceedings of 9th Australasian Conference on Information Security and Privacy*, 402-413, Sydney, Australia, July.

Song, D., Wagner, D., and Perrig, A. (2000) Practical Techniques for Searches on Encrypted Data, *Proceedings of IEEE Symposium on Security and Privacy*, 44-55, CA, USA, May.

Stiegler, Helmut G. (1979) A Structure for Access Control Lists, *Software Practice and Experience*, 9(10), 813-819.

Varadharajan, V., Allen, P., and Black, S. (1991) An Analysis of the Proxy Problem in Distributed Systems, *Proceedings of the IEEE Symposium on Security and Privacy*, 255-275, CA, USA, May.

COPYRIGHT