Protecting Sensitive Information in Directory Services Using Virtual Directories

William Claycomb[1] and Dongwan Shin[2]

[1] Sandia National Laboratories, P.O. Box 5800, MS 0823, Albuquerque, NM, 87185-0823, USA
wrclayc@sandia.gov
[2] Computer Science Department, New Mexico Tech, Socorro, NM, 87801, USA
doshin@nmt.edu

Abstract. Directory services are commonly used to store information related to individuals, and often act as a source for security services, such as authentication and access control, in collaborative applications within/across organizations. Hence, there is an urgent need to protect the sensitive information they contain. Existing solutions offer minimal protection against insider attacks, a growing threat to both government and industry data services. In this paper we present a solution for data protection that leverages virtual directories and data encryption to provide a user-centric approach to data protection, delegation, and collaboration. A security architecture is presented, along with the discussion of the benefits and vulnerabilities of our approach. We also discuss a proof-of-concept implementation and performance testing results.

Keywords: Access controls, Cryptographic controls, Data encryption, Public key cryptosystems, Privacy, Information resource management, Data dictionary/directory.

1 Introduction

Directory services are used to store information about objects within an organization, such as users, computers, etc., and are organized in a hierarchical structure. Often, directory services (or simply, *directories*) are used as authoritative data sources for many collaborative applications that require user information, such as instant messaging, workflow systems, and social applications. In some cases, the information contained in a directory is considered confidential, such as employee ID number, clearance level, or other *personally identifiable information (PII)*. While techniques exist to protect this information, they do not adequately prevent a user with administrative privileges from unauthorized access. Moreover, many organizations have several directories, some containing duplicate information. This can arise from inadequate information planning, applications requiring proprietary data sources, or the need to protect specific information at different levels. Determining authoritative data sources and synchronizing data across directories is a challenging and ongoing task for many corporations.

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 244–257, 2009.

Each object in a directory is described by a set of *attributes*. Examples include name, address, email, or manager name. Access control lists (ACLs) or marking attributes *confidential* [1] are two commonly practiced techniques to protect attribute data in a directory from ordinary users. In general, however, directories are used to share information, and rarely enforce access controls beyond simple user authentication (only users with accounts on the system may access the data). An insider threat, someone with authorized access, could potentially retrieve personal information about every object in the directory. The malicious activities possible with such information could include selling information to competing companies, foreign governments, or spammers, or even worse - the targeted attack of specific individuals within the company, such as domain-level administrators, known as *context aware attacks*, or *spear phishing* [2].

The threat of unauthorized access of sensitive data by employees or other authorized users, known as "dedicated insiders", is well documented [3,4,5]. In January 2008, the U.S. Secret Service and CERT issued a report titled "Insider Threat Study: Illicit Cyber Activity in the Government Sector" [3]. This study outlines a multi-year project, started in 2002, that explores the activity and threats posed by insiders. Among the key findings of this study are the following:

- Most of the insiders had authorized access at the time of their malicious activities
- Access control gaps facilitated most of the insider incidents, including:
  - The ability of an insider to use technical methods to override access controls without detection
  - System vulnerabilities that allowed technical insiders to use their specialized skills to override access controls without detection

In this paper we present a solution for data protection that leverages the concept of a virtual directory and data encryption to provide a user-centric approach to sensitive information protection, delegation, and collaboration. Specifically, we discuss an architecture for protecting individual attributes in directory services from unauthorized access. In standard configurations for directory data usage, clients communicate directly with directory services using the Lightweight Directory Access Protocol (LDAP). Clients connect to a specific port on a specific server, and may authenticate using various methods, including providing a username and password, if necessary. Our architecture is based on a middle layer placed in between the client and server, called a virtual directory, to handle LDAP transactions between them. A data protection component within the virtual directory is introduced and it relies on information provided by the client to encrypt sensitive information. While other solutions have proposed encrypting attribute information, our architecture provides this capability without requiring additional software or hardware on either the client or destination server.

The remainder of this paper is organized as follows. Section II presents an analysis of background material and related solutions. Section III describes our architecture in detail. In Section IV, we analyze the results of our implementation testing. Section V discusses the architecture, including various advantages, as well as attack models. Section VI concludes this paper with a glimpse of future work.

2 Background and Related Work

2.1 Directory Services

Directories are collections of information related to objects in an organization. These objects often include users, computers, or contacts. *Directory Services* are the services which make this data available for use by others. Frequently, the intention is to provide a single point of access for various applications and individuals to find information about users and other objects for different purposes including collaboration within/across organizations. The information contained within the directory may come from direct input, and can be manually maintained, but also may be referenced and managed indirectly from other corporate data repositories, such as databases and other information stores. Commonly used directory services are Microsoft Active Directory [6], IBM Tivoli [7], Apple Open Directory [8], Novell eDirectory (formerly called Novell Directory Services) [9], OpenLDAP [10], Fedora Directory Server [11], and Sun Java System Directory Server [12].

2.2 Protecting Information in Directory Services

A few techniques exist for protecting the information stored within a directory itself. In general, access control lists (ACLs) can be used to implement some form of protection in most directories. For instance, in OpenLDAP, ACL protection can be applied to individual objects, groups of objects, specific LDAP filters, or a list of attributes [13]. Other techniques are almost exclusively implementation-specific.

Microsoft Active Directory [6] provides additional access control features through the use of *confidential attributes* [1]. This is a setting applied to the *searchFlags* component of individual attributes, and is only supported on Microsoft Windows Server 2003 SP1 and later. When processing confidential attributes, the directory server checks for additional access control rights associated with the requesting user. This particular type of access, called "CONTROL_ACCESS," is granted to administrative accounts by default, but can be delegated to other accounts individually.

Another approach to protecting attributes is encrypting them. Fedora Directory Server [11] has the capability to encrypt all instances of specified attributes. This means that for every object containing such attributes, the data in that attribute is encrypted using a symmetric key known to the directory server. Various encryption methods can be configured, and different attributes can be encrypted using different ciphers. Encryption and decryption are handled by the directory server itself, so access to attributes is not controlled by this method. However, data would be protected from unauthorized access if the directory data was stolen or otherwise compromised.

A third approach to protecting directory attributes is described in [14]. This method is not dependent on a particular directory implementation. Rather, it uses public key infrastructure (PKI) to allow users to control the encryption of attributes related to their own directory information. This solution describes different methods for using PKI to ensure either data authenticity alone, or data

authenticity combined with confidentiality. Specific solutions are proposed for scalability and usability purposes.

Additionally, [15] proposes encrypting directory information for users based on a *unique-id* chosen for each user. This method applies primarily to public directory servers, and does not address the issue of preventing unauthorized access so much as it addresses the issue of preventing access to the entire directory. For instance, a company could share contact information publicly, and provide selected clients with appropriate unique-ids, without worrying that the entire directory would be scanned for email addresses. One important aspect of the work is to choose a unique-id well, so that it cannot be easily guessed, but can still be easily shared with authorized users.

2.3 Metadirectories and Virtual Directories

One way to protect personal information is to reduce the number of different data stores containing personal information. This can be a complicated task, particularly for businesses with many disparate data sources. The International Data Corporation (IDC) and Gartner Groups have found that large corporations may have in excess of 100 data stores containing user information [16]. Additionally, proprietary systems often do not interact with other data sources. Consolidating data into a single data source is often not possible, due to constraints on who may have access to specific information. Technology has emerged to address these issues, specifically by referencing the underlying data sources and presenting end-users with customized views of the data they require, and by synchronizing data between different data sources. Two similar but distinct methods of handling these tasks are *virtual directories* and *metadirectories*.

Metadirectories. Analyzing the origin of the word "metadirectory," we see the Greek phrase "meta-" which means "after," or "beyond." In modern English, this term often describes abstraction. Thus, a metadirectory is an abstraction of an actual directory. In this sense, it acts as a directory in some instances, by providing user interaction via LDAP, but does not act as a directory in other instances, because it is not the actual authoritative source of directory information. A metadirectory is used to abstract data from other directories into a single source, which can be used for two purposes.

The first use is for end user reference. Users may access the data collected by a metadirectory via LDAP. In particular, this not only reduces the number of data sources an end user connects to, but enables customization of directory data for individual uses. Therefore, in one sense a metadirectory is a real directory - information is actually stored locally, and is queried directly by end users.

However, this repository is not the authoritative source of the data. Rather, data synchronization must occur between the metadirectory and source directories to ensure consistency and accuracy of the data. It is the synchronization of data which is its second purpose. When different data sources must store the same information, it is desirable to have a single source of authoritative data, which is synchronized with other data sources. For example, if the HR department is the authoritative source for a user's telephone number, but the company

directory application, which uses its own data source, also requires a telephone number to be stored, a metadirectory could be used to automatically synchronize the data from the original source (HR). A more advanced use of metadirectories is for *user provisioning*, which is a modified version of synchronization, where new user accounts are created and prepared for use, based on data in an authoritative source, such as an account database. Examples of metadirectory implementations include Microsoft Identity Lifecycle Manager 2007 [17], Sun ONE Meta-Directory [18], and Critical Path Meta-Directory Server [19].

Virtual Directories. "A virtual directory functions as an abstraction layer between applications and data repositories." [20] In contrast to metadirectories, *virtual directories* do not maintain the data in a standalone data source (though some offer *data caching*, which does store data locally for improved performance). Rather, virtual directories reference various data sources and present a consolidated view to the end user. This has the advantage of not requiring data synchronization - the data presented is always real-time, directly from the source. Most virtual directory implementations have the additional capability of acquiring data from sources other than directories, such as databases, and presenting this information to end users via LDAP.

Virtual directory instances can be highly customized to modify, or *transform*, data prior to client use. Additionally, some products offer data synchronization as well, which when coupled with a virtual directory instance, could be used for user provisioning in much the same way as a metadirectory. Virtual directory products currently in use include Radiant Logic's RadiantOne [21], Symlabs Virtual Directory Server [22], and Oracle Virtual Directory [23].

3 Our Approach

Our approach to protecting sensitive information in directory services is to encrypt that information using user-controlled keys and to provide access to that data using user-controlled delegation. This user-centric approach follows current trends in computer security and privacy, but should not interfere with more traditional approaches to access control. Our approach also maintains usability with existing client applications and source directories. To better understand the overall picture of our solution, it is first important to understand various key components.

3.1 Data Encryption

Encrypting sensitive information to protect it from misuse is hardly a new concept. In the simplest application towards protecting information in directory services, an end user would simply encrypt sensitive information and then store the encrypted data in a directory.[1] To share information, the user would share the

[1] Note that encryption here is orthogonal to that of secure LDAP (or LDAPs). The former is for data protection in data stores while the latter is for network communication protection. It is assumed that LDAPs is supported for better security in our approach.

encryption/decryption key with another user, who would obtain the encrypted form from the directory and decrypt it locally.

However, this approach presents several usability and security problems. First of all, the confidentiality of the data relies entirely on the shared key. If a malicious user were to obtain this key, or if an authorized user were to share it with an unauthorized party, the information could be compromised. Data confidentiality could be provided by using an asymmetric encryption algorithm, such as RSA, but this still does not protect the data from unauthorized access.

Secondly, this approach requires the user to perform encryption and decryption before and after retrieving the information from the directory. At best, this could be accomplished by a custom application, which interfaced directly with the client LDAP application. At worst, existing client LDAP applications would need to be rewritten to incorporate encryption and decryption. This is an undesirable situation for which a simple solution exists: add a third party, between the client and server, to handle encrypting and decrypting the data.

The third party component could be a custom component written specifically for the purpose of handling encryption and decryption of information between the client and directory. However, we find it much more useful to leverage the existing technology of virtual directories to provide the third party component to the model. The benefits of doing so are numerous, and will be discussed in detail later.

3.2 System Architecture

If we consider a virtual directory as the container of the third party component - the one responsible for encrypting and decrypting data - we must consider several key aspects, namely: how does the virtual directory obtain key information from the client, how does the virtual directory perform pass-through authentication to destination directories, and how does the virtual directory manipulate the data in the destination directory? The system architecture, shown in Figure 1, is proposed to address those questions.

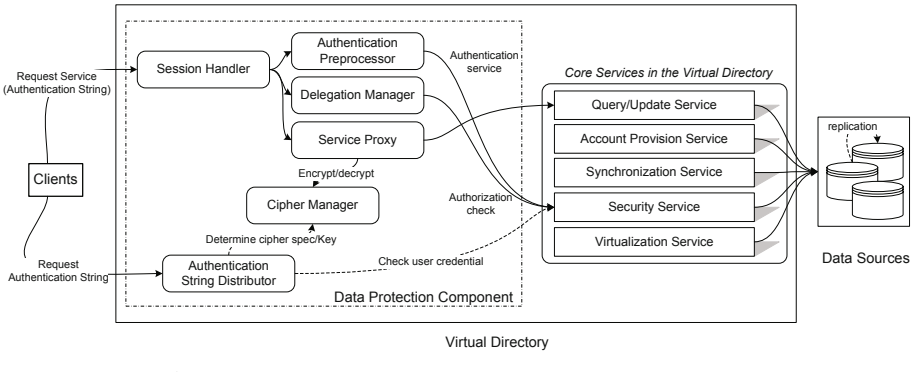

Fig. 1. System Architecture for Protecting Sensitive Information in Directory Service using Virtual Directory

Obtaining Client Key Information. When LDAP communications occur between a client and server, several standard pieces of information are transmitted. These components are generally configured by the client application, and can be changed by the end user. They are: username, password, and destination server name and port. We leverage these components to pass encryption information to the server as follows. The destination server and port are replaced with the destination server name and port of the virtual directory. This configures the client to communicate with the virtual directory, instead of the original destination directory. Note that the original destination directory is transparent to the client through virtualization, which is one of the core services in virtual directories, as shown in Figure 1. The password remains the same as the original password used to authenticate to the original destination directory.[2] We replace the username component with a string which is the concatenation of the following: the client username, ID_c, the hash of the original user password, $H(pwd_c)$, and a symmetric key between the client and virtual directory, K_{cv}. The last two components are encrypted using a secret key known only to the virtual directory server, K_v. The addition of these last components requires additional setup, performed by the Authentication String Distributor shown in Figure 1 with access to the virtual directory key, K_v, and is also discussed in detail later. The resulting string is called an *authentication string (AS)*:

$$ID_c|\{K_{cv}|H(pwd_c)\}_{K_v}$$

Performing Pass-Through Authentication. We do not ignore traditional authentication and access control methods with this solution. Unless configured for anonymous authentication (also called *anonymous bind*), the destination LDAP server will expect a client to authenticate prior to data retrieval. Some Virtual Directory implementations allow a static username and password to be used for every transaction, but this defeats the purpose of fine-grained access control. Rather, we will pass the original client username and password, obtained from the AS and password provided by the client, to perform an initial bind prior to data retrieval. If this bind is not successful, then no data transmission occurs between the virtual directory and the client.

Storing the Data. Once the user has successfully authenticated to the destination directory, we use the transformation capabilities of the Authentication Preprocessor module in our architecture to extract the user's symmetric key, K_{cv}, and password hash, $H(pwd_c)$. The password hash is used as an additional measure of security against an attack where a malicious administrator may change the user's password and, using the original authentication string, masquerade as the user. While this step may seem redundant, it is necessary because of the nature of LDAP clients. Many LDAP clients allow users to cache login information, including the username. An attacker would need to have no knowledge of the client secret key, K_{cv}, if he used a cached authentication string and a newly-reset password. However, if the client were configured to prompt for a password

[2] Pass-through authentication is more commonly practiced than single-sign-on, in virtual directories.

every time, while still retaining a cached authentication string, the user's password hash could be checked against the password hash encrypted by the virtual directory's secret key in the AS. In this instance, a changed user password would cause a failure, because its hash would not match the original hash in the AS.

Once verified, the user's secret key, K_{cv}, is used to perform encryption or decryption of data stored in the directory. The protocol for reading an encrypted attribute is shown in Figure 2, and the protocol for writing an encrypted attribute is shown in Figure 3.

3.3 Collaboration and Delegation

One of the key components to our approach, as shown in Figures 2 and 3, is the capability of the user to delegate access to attributes, enabling collaboration with other users. We modify a traditional Access Control List (ACL) model, by identifying the access control entry principal by password hash. If another user is delegated permission to access a particular attribute, the corresponding password hash, $H\{pwd_c\}$, must exist (read and/or write) in the ACL attached to the attribute when stored in the destination directory. Alternatively, if the attribute owner attempts to access the attribute, identified by $H\{pwd_o\}$, full access is granted. The ACL is managed by the virtual directory server, and again would require additional interaction by the attribute owner to manage. This is supported by the Delegation Manager in our system architecture.

Client	Secure comm.	Virtual Directory Server	Secure comm.	Dest. server
Auth. string + pwd'_c	⟶	Authenticate client using $ID_c \vert \{K_{cv} \vert H\{pwd_c\}\}_{K_v}$		
		Verify $H\{pwd_c\} = H\{pwd'_c\}$		
Recv. auth.	⟷	If match, continue authentication	⟷	Authenticate client, using ID_c and pwd'_c
Request	⟶	Receive request		
		$\{data \vert H\{pwd_O\} \vert ACL\}_{K_{cv}}$	⟵	$\{data \vert H\{pwd_O\} \vert ACL\}_{K_{cv}}$
		Decrypt using K_{cv} provided by client, and check $H\{pwd_c\} = H\{pwd_O\}$ or $H\{pwd_c\} \in ACL_R$		
	⟵	If match, return to client		

Fig. 2. Reading an encrypted attribute

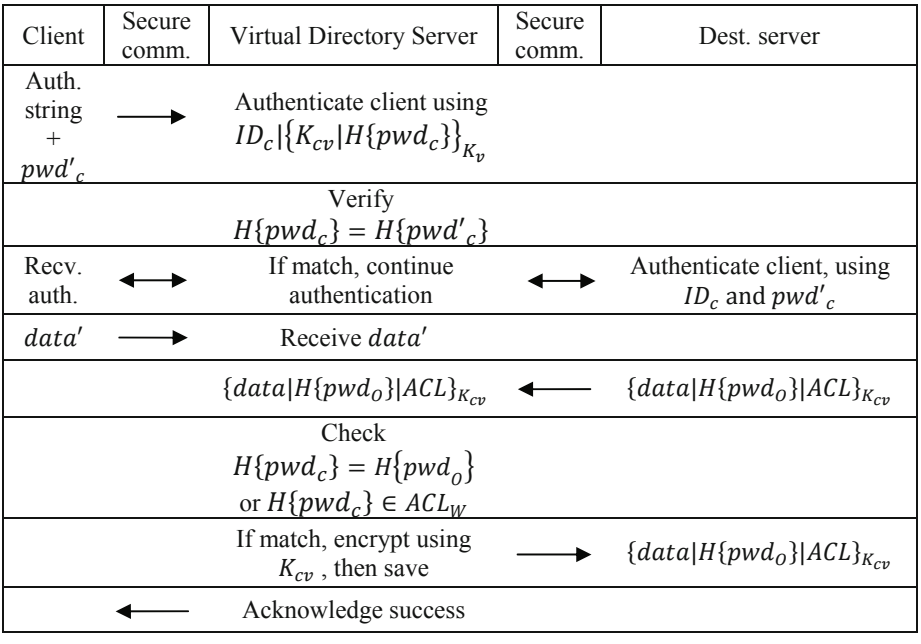

Fig. 3. Writing an encrypted attribute

4 Implementation and Performance Testing

The system architecture has been implemented for testing and usability purposes. Directory servers were represented using Microsoft Active Directory Administration Mode (ADAM) [24]. The Virtual directory component was modeled via a custom application on a separate system, and clients were simulated using directory services functions in Microsoft Visual Studio .NET 2008.

To accurately compare results of testing different configurations of using virtual directories, three separate ADAM instances were created, to represent the following situations:

- No virtual directory - communication directly between the client and destination directory server
- A virtual directory handing communication between client and destination directory server, but processing no encrypted attributes
- A virtual directory handing communication between client and destination directory server, and processing some encrypted attributes

Data was simulated using real-world directory objects from a corporate Active Directory instance. For each test, 10,000 user objects were created, with 25 attributes populated for each user. Testing both with and without the virtual directory server, as well as with and without encrypted attributes was performed. When using encryption to protect attributes, three attributes per user were

stored encrypted. The time to perform each operation was recorded, as well as the overall size on disk of each directory instance.

Creating New Objects. Creating new objects in a directory service, known as *account provisioning*, involves two distinct steps: creating the new object, and populating the attributes of that object. For testing purposes, this was measured as one atomic operation. Table 1 shows the average time necessary for each testing configuration to create new accounts.

Table 1. Average new account creation time (ms)

Configuration	Time (ms)
No virtual directory - no encryption	92
Virtual directory - not encrypting	99
Virtual directory - encrypting	205

Modifying an Non-Encrypted Attribute. When modifying an attribute, the virtual directory server is able to detect whether or not the attribute is encrypted. If the attribute is not encrypted, the virtual directory simply passes through the modification request from the client to the destination directory server. The time to modify an non-encrypted attribute is shown in Table 2.

Table 2. Average time to modify a non-encrypted attribute (ms)

Configuration	Time (ms)
No virtual directory - no encryption	6
Virtual directory - not encrypting	12
Virtual directory - encrypting	12

Modifying an Encrypted Attribute. To modify an encrypted attribute, the virtual directory is required to decrypt the authentication string, extract the shared secret key of the client, K_{cv}, and check to see if the requesting client is either the data owner, or listed as an authorized user of that particular object attribute. In some cases, the performance is dependent on whether or not the attribute is blank or has been previously populated. The time to complete these tasks is shown in Table 3.

Table 3. Average time to modify encrypted attributes (ms)

Configuration	Time (blank attribute (ms))	Time (populated attribute (ms))
No virtual directory - no encryption	5	6
Virtual directory - not encrypting	12	12
Virtual directory - encrypting	106	100

Table 4. Average time to retrieve an attribute (ms)

Configuration	Time (non-encrypted attribute (ms))	Time (encrypted attribute (ms))
No virtual directory - no encryption	3	3
Virtual directory - not encrypting	6	6
Virtual directory - encrypting	6	98

Table 5. Directory size on disk (MB)

Configuration	Beginning size (MB)	Final size (MB)
No virtual directory - no encryption	6.3	56.6
Virtual directory - not encrypting	6.3	56.6
Virtual directory - encrypting	6.3	77.6

Retrieving an Attribute Value. Retrieving an attribute also depends on the particular configuration and whether or not the attribute is encrypted. The time to retrieve an object attribute is Table 4.

Directory Size on Disk. The disk space necessary to store a directory services instance can be easily measured when using ADAM. Table 5 shows the beginning and ending size of the file used to store the directory for each test configuration. The final file size was recorded after all test accounts had been created and all test attributes modified.

5 Discussion

Analyzing our solution should be approached from several angles. First, what are the advantages to using a virtual directory as the encryption provider? Next, what are the benefits and limitations of using encryption to protect the data in directory services? No analysis of data protection would be complete without discussing vulnerabilities and attack models. Finally, how well does the solution perform, particularly in real-world situations?

5.1 Advantages of Using Virtual Directories

Virtual directories allow us to use existing technology to overcome barriers such as application reliability and security. Additionally, many virtual directory implementations compliment existing access control methods, by specifying yet another level at which users may be granted permission to access specific objects. Another distinct advantage is that virtual directories are client and destination independent. That is, any LDAP client can be configured to use a virtual directory, and virtual directories can be connected to almost any type of directory service, as well as other types of data sources, such as databases.

One additional advantage could be gained by incorporating a metadirectory service into the solution as well. By using the data synchronization component of metadirectories, we can ensure that all instances of a particular attribute related to a certain user are encrypted. This takes data protection one step further, by eliminating the need to individually protect data in each separate data source.

5.2 Advantages of Using Encryption

Allowing the user to maintain the encryption/decryption key used in this solution is a user-centric approach [25,26,27] to data protection and identity management. In general, user-centric identity management is a method of managing user identities where the users themselves control what information is stored, the actual content of that information, and who is allowed to view the information. One motivation for this concept is privacy, accomplished by giving users the choice about what is shared, and with whom it is shared. [14]. Allowing the users to control the key provides them with complete control over the content of the data, and by including a user-specified ACL in the model, we allow users to specify who is allowed to access that data.

This is a particular advantage when considering one possible threat to conventional ACL-based access control. Administrative users may have permissions to modify ACLs on directory objects, and could grant themselves permission to read attributes intended to be confidential. By encrypting this data, we mitigate this particular threat.

5.3 Vulnerabilities

Of course, it's still possible for a dedicated attacker to compromise this system by gaining administrative access to the virtual directory server. This type of attack is difficult to prevent in any architecture. However, the type of attack which would compromise the data stored using our solution would be a more sophisticated attack, require more technical knowledge, and be more risky in terms of detection. No longer is a simple ACL modification necessary, now an attacker must either compromise the virtual directory application and intercept unencrypted data in transit, or he must compromise the data during transit or storage, by attacking the SSL connection. This is a much harder attack to undertake, and the risk of detection by network monitoring tools is greater.

A much simpler attack on this solution would be to compromise the user's secret key. However, this would be useless without also compromising the user's original password. Tools such as keystroke logging and administrative access to the user's computer could be used to mount such an attack, but again, this requires more technical skill, and comes with a higher risk of detection.

5.4 Performance Analysis

Examining the performance tables shown in Section IV seems to reveal a large difference between the time it takes to manage encrypted attributes versus the

time to manage unencrypted attributes. This is hard to avoid - encryption is not computationally easy - but we believe this large difference is not functionally detrimental to the overall performance of the directory. Often, attributes which need to be protected are rarely accessed. A difference of 100ms is hardly noticeable when the attribute is only accessed a few times per day.

More significant to the performance of the solution in the real world is the user interaction required. An initial interaction with the Authentication String Distributor is necessary to establish the authentication string. This could be done via secure web services, for example, but still requires user configuration of the local LDAP client. Additionally, any authorized password change would require a new authentication string to be issued.

Collaboration and delegation would also be an application management issue. To add a user to the object ACL, the owner would need to interact with the Delegation Manager, and would need to have access to the hash of the delegatee's password. Again, password changes would require a modification of the ACL stored on the directory object. For large-scale delegation, this could become unwieldy. However, for sharing information with a few select sources, the benefits of this solution appear to outweigh the administrative overhead.

6 Future Work and Conclusion

We have presented an architecture for protecting sensitive information in directory services, which often work as a data hub for collaborative applications. This architecture leverages the existing technology of virtual directories as a layer between client and directory service applications. This middle layer is responsible for handling communication between client and server, and manages encryption and decryption routines with information provided by the client. The client provides the information using standard LDAP client fields, requiring only a reconfiguration - not a recode - of client applications. By allowing users to control and protect encryption keys, we enable a user-centric model of data protection, and we reduce the threats posed by dedicated insider attacks.

Future work will include additional real-world implementation and testing. Integration with existing PKI infrastructure may also pose an interesting approach, and could help to eliminate some of the overhead associated with user key and password management. Finally, by examining existing data stores and the applications that utilize them, we may come to a better understanding of how sensitive information is distributed over an enterprise-level environment, and may discover new approaches to information protection based on such knowledge.

References

1. How to mark an attribute as confidential in windows server 2003 service pack 1, `support.microsoft.com/kb/922836`
2. Jakobsson, M.: Modeling and preventing phishing attacks. In: Phishing Panel at Financial Cryptography (February 2005)

3. Kowalski, E., Cappelli, D., Conway, T., Willke, B., Keverline, S., Moore, A., Williams, M.: Insider threat study: Illicit cyber activity in the government sector, U.S. Secret Service and CERT, Tech. Rep. (January 2008)
4. Keeney, M., Capelli, D., Kowalski, E., Moore, A., Shimeall, T., Rogers, S.: Insider threat study: Computer system sabotage in critical infrastructure sectors, U.S. Secret Service and CERT/SEI, Tech. Rep. (May 2005)
5. Shaw, E., Ruby, K., Post, J.: The insider threat to information systems. Security Awareness Bulletin 2-98 (September 1998)
6. Windows server 2003 active directory, `www.microsoft.com/windowsserver2003/technologies/directory/activedirectory/default.mspx`
7. Ibm tivoli directory server, `www-306.ibm.com/software/tivoli/products/directory-server/`
8. Mac os x server open directory, `www.apple.com/server/macosx/opendirectory.html`
9. Novell edirectory, `www.novell.com/products/edirectory/`
10. Open ldap, `www.openldap.org/`
11. Fedora directory server, `directory.fedoraproject.org/`
12. Sun java system directory server, `www.sun.com/software/products/directory_srvr/home_directory.xml`
13. Carter, G.: LDAP System Administration. O'Reilly, Sebastopol (2003)
14. Claycomb, W., Shin, D., Hareland, D.: Towards privacy in enterprise directory services: A user-centric approach to attribute management. In: Proceedings of the 41th IEEE International Carnahan Conference on Security Technology, Ottawa, Canada (2007)
15. Berger, A.: Privacy protection for public directory services. Computer Networks and ISDN Systems 30, 1521–1529 (1998)
16. Chacon, M.: Unifying diverse directories. Network Magazine 16, 70–75 (2001)
17. Microsoft identity lifecycle manager 2007 (2007), `www.microsoft.com/windowsserver/ilm2007/default.mspx`
18. Sun one meta-directory, `www.sun.com/software/products/meta_directory/home_meta_dir.xml`
19. Critical path meta-directory server, `www.criticalpath.net/pdf/MetaDirectory.pdf`
20. I. Radiant Logic, Using virtualization to leverage your investment in active directory, Radiant Logic, Inc., Tech. Rep.
21. Radiant logic, Inc., `http://www.radiantlogic.com/main/`
22. Symlabs virtual directory server, `http://symlabs.com/products/virtual-directory-server`
23. Oracle virtual directory, `http://www.oracle.com/technology/products/id_mgmt/ovds/index.html`
24. Windows server 2003 active directory application mode, `www.microsoft.com/windowsserver2003/adam/default.mspx`
25. Koch, M., Worndl, W.: Community support and identity management. In: Seventh European Conference on Computer-Supported Cooperative Work - ECSCW 2001, Bonn, Germany (September 2001)
26. Koch, M.: Global identity management to boost personalization. In: Ninth Research Symposium on Emerging Electronic Markets, Basel, Switzerland (September 2002)
27. Josang, A., Pope, S.: User centric identity management. In: Asia Pacific Information Technology Security Conference, AusCERT 2005, Austrailia (2005)