A Technical Overview of Novell SecretStore 3.2

Feature Article

NOVELL APPNOTES

Cameron Mashayekhi
Senior Software Engineer
Novell, Inc.
cameron@novell.com

This AppNote presents a technical overview of Novell SecretStore, covering the architecture of both client and server platforms, a discussion of how SecretStore is used in single sign-on scenarios, and some of the new features in this latest release of SecretStore.

Contents:

Topics	Secure Identity Management (SIM), SecretStore, single sign-on, authentication, network security
Products	Novell SecretStore 3.2
Audience	network application developers
Level	beginning
Prerequisite Skills	familiarity with user authentication mechanisms
Operating System	NetWare 5 and above
Tools	NDK
Sample Code	no

Introduction

Because networks are an integral part of today's computing environments, it has become crucial to solve the complex problems of access, storage, and retrieval of sensitive data from a secure storage from any point on the network. As an infrastructural component of the Secure Identity Management (SIM) provisioning architecture, Novell's SecretStore service is designed to address these issues.

The inherent function of this kind of storage is to protect the confidentiality and integrity of the data stored in it against unwanted modification and disclosure. This type of data, commonly called *secrets*, are stored so that the owner—or a security system on behalf of the owner—can use them. Consequently, the methods of access, storage, and retrieval of the secrets should also be protected.

SecretStore is designed to place login credentials in eDirectory and securely store them there. Single sign-on services such as Novell SecureLogin, Novell iChain, and Novell Portal Services, as well as third-party applications, can then access and use these credentials (secrets) on behalf of the authenticated user.

This AppNote presents a technical overview of Novell SecretStore, covering the architecture of both client and server platforms, a discussion of how SecretStore is used in single sign-on scenarios, and some of the new features in the latest release of SecretStore. A follow-up AppNote will cover the new SecretStore programming interfaces.

SecretStore Architecture

The implementation of SecretStore is based on a hidden set of attributes added to eDirectory objects. These objects represent entities that must utilize the SecretStore service. By default, these attributes are added to the User object as extensions to the eDirectory schema. However, the same extensions can be added to any desired type of object.

This obviously implies that a SecretStore is created for each object, and each user can have his or her own SecretStore. Secrets in SecretStore are encrypted using keys generated per user by FIPS-140 (Federal Information Processing Standard) certified NICI (Novell International Cryptographic Infrastructure) when the user's SecretStore is created. The algorithm and key size selection are guided by requiring the strongest algorithm with the largest key size available through NICI on the server.

Hidden Attributes

By definition, hidden attributes in eDirectory are not accessible from client workstations or other servers through commonly available means such as Novell administrative utilities or other applications built on Novell's published eDirectory interfaces.

The only ways to access SecretStore are through the following:

- SecretStore administrative utilities (for example, SecretStore Manager, SecretStore Status, or snap-ins to ConsoleOne)
- Different types of SecretStore connectors that are built on the SecretStore Service APIs and interfaces

As a result, the SecretStore service controls and regulates the proper access to and disclosure of the protected data.

Interfaces

The interfaces for the administrative utilities and SecretStore connectors are based on SecretStore NCP-92 and/or LDAP extensions on Windows workstations. Client applications running on NetWare and UNIX platforms (Linux, Solaris, and AIX) are only LDAP-based extensions. NCP-92 is SecretStore's encrypted transport for secure transmission of data between a client and a server.

The NCP-92 transport utilizes NICI for encrypting inbound and outbound data to and from SecretStore. This strong encryption is based on the common algorithm and strongest session key that is negotiated between NICI running on the client and server. The key and algorithm are negotiated at the end of the user's successful Client32 authentication to eDirectory. At the end of each session the keys are destroyed. Each session with a different server has its own keys.

An LDAP extensions-based connection can be established through an SSL LDAP bind with the server. The bind can be done by the application prior to calling SecretStore, or by SecretStore if the application provides the appropriate credentials when calling the SecretStore interface. If a connection over LDAP is not SSL based, SecretStore rejects it. The strength of encryption on the wire is dependent on the SSL-negotiated algorithm for the session.

Architectural Diagrams

Figure 1 illustrates the client NCP and LDAP protocol stacks on a client workstation.

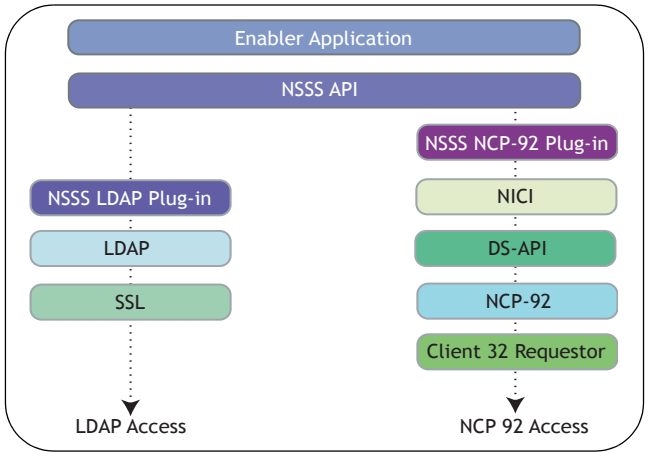

Figure 1: *Novell SecretStore C client architecture.*

Note: The NCP-92 stack is available only on clients running Microsoft Windows operating systems, when Novell Client 32 is installed.

Figure 2 illustrates the server NCP and LDAP protocol stacks on a server platform.

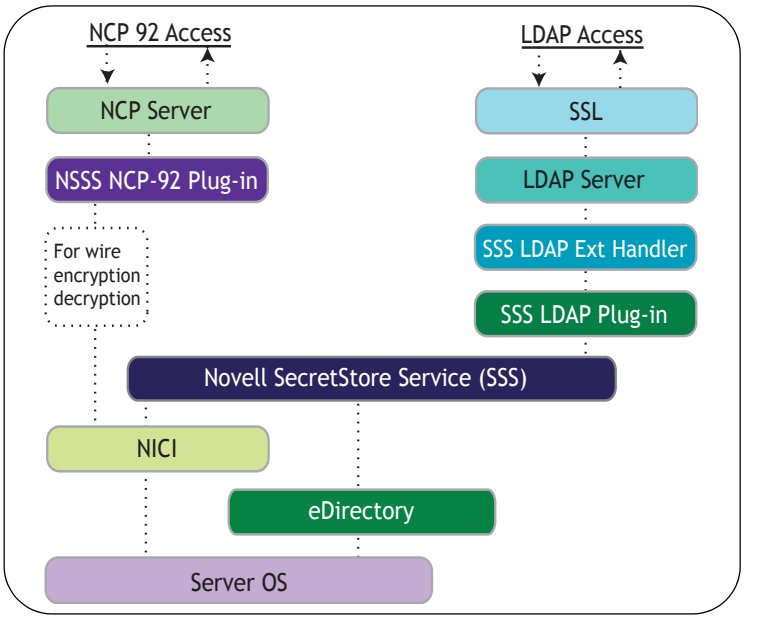

Figure 2: *Novell SecretStore C server architecture.*

Note: The NCP-92 stack is available only on NetWare server operating systems.

SecretStore at Work: Single Sign-on

One of the fundamental applications of SecretStore is single sign-on. SecretStore-enabled applications can use SecretStore to store the authentication credentials that network applications require to authenticate the user. After a user is authenticated to eDirectory, the enabled client component of the application can retrieve these previously-stored credentials from SecretStore and use them to automatically authenticate the user without his or her active involvement.

For example, user IDs and passwords, biometrics, or other credentials can be stored and used later by the user to access secure Web sites, e-mail applications, and other systems. Basically, SecretStore is used to store these credentials so that the user can securely reuse them to authenticate to applications on subsequent attempts to access these sites and services.

Rationale Behind Single Sign-on

Due to the distributed nature of network-based applications and services, there is a constant need to authenticate clients of these applications and services before granting the clients access to the resources they need. Currently, there is no standard set of universally-known services that can authenticate users and grant them universally-accepted identities, credentials, and clearances. Consequently, each distributed application or service must authenticate its users independently.

To begin using network services, users typically authenticate to some access portal. These access portals can be network servers, Internet portals, or Internet Service Providers. After the initial authentication, the network users must authenticate again and again when they access different systems, Web sites, e-mail servers, and so on.

Research shows that users often forget their passwords, which results in extra costs and lost productivity associated with administrative password resets, not to mention the security risks and vulnerabilities involved in this process. Also, the process of starting a session with the network can become cumbersome when a user has to authenticate a number of times to the different applications and services commonly used during that session.

To simplify having to remember passwords for signing on to most or all of the applications and services they access, users commonly select the same password for multiple applications. This is called *password synchronization*. As a result, the would-be attacker has only to target the weakest link in the chain of applications that the user uses. Once the attacker cracks that weakest link and finds the password, he can gain access to all of the user's applications by masquerading as the user, no matter how intricate the security systems on those applications are.

Single sign-on can help solve this problem by allowing users to authenticate once to eDirectory and thereby obtain access to SecretStore services. For authenticated users, SecretStore-enabled applications known as *connectors* can securely capture and store authentication credentials for these network services. On users' subsequent requests for access, these enabled applications can authenticate the users to these services by securely replaying these credentials back to the target service, thereby eliminating the need for user interaction with those applications.

With SecretStore, users do not have to remember multiple passwords and the problem of password synchronization is eliminated. Users can go ahead and set a different password for each application or service, knowing that single sign-on will take care of all the authentication. Authenticated users can use SecretStore's secure management applications to access and manage these secrets for a task as simple as a database look-up or update.

Administrators of eDirectory or SecretStore can be the same person, or they can be designated separately. These administrators manage users' SecretStores for common operations, such as populating the store ahead of time for the users for the first time. The highly secure design of the SecretStore service allows for the designated administrators to access users' SecretStores for administrative operations, such as adding, removing, or unlocking these stores, without allowing the administrators to actually read the users' secrets.

Benefits of Single Sign-On

The implementation of a single sign-on solution can reduce the costs associated with the following issues:

- Compromised security as a result of weak and synchronized passwords
- Administrative password resets due to forgotten passwords for applications
- The time and effort required for users to sign on to a multitude of services and applications during their workday
- Keeping track of the credentials used for authenticating to these services to maintain a secure environment

As a foundation for single sign-on implementations, SecretStore is designed to provide a secure solution for Novell customers facing these issues.

Two Methods of Single Sign-on

There are two basic methods to single sign-on using the SecretStore service:

- Application Connectors
- Universal Connectors

These methods should use the new Shared SecretStore, which allows applications to share secrets stored by different services.

Application Connectors. The use of application connectors was the original and older method of enabling individual applications to access SecretStore. One popular example of an application connector is the "Connector for Lotus Notes" that was part of the now-obsolete Novell Single Sign-on product line. While SecretStore has survived as an enabling technology, application connectors have not proven as successful. Unless the application to be enabled by this technology accommodated an authentication API, implementation of single sign-on without the application developer's assistance could be difficult or impossible.

However, there have been some very successful custom application connector implementations that are still in use today. For example, Novell GroupWise and the Novell Client for Windows have built-in connectors that can provide single sign-on when SecretStore is available.

Universal Connectors. Universal connectors are currently the most efficient, cost effective, and common way of utilizing SecretStore for applications to single sign-on to target services. Universal Connectors can automatically intercept any prompt for authentication and then interact with the user to collect credentials and store them in SecretStore. These stored credentials are then passed to the application on subsequent invocations. The passed credentials authenticate the user without active interaction. This also allows for populating SecretStore for known applications beforehand. The greatest benefit of universal connectors is that they can often provide single sign-on to an application without requiring modifications to the application code itself.

Novell SecureLogin (NSL) is an advanced example of a universal connector. NSL relies on the architecture and methods of operation in the Windows operating system, browsers, and terminal emulators to recognize the prompts for authentication. The user or the administrator uses the built-in wizards or scripting language capability of the product to enable NSL to recognize the target applications, browsers, and service authentication dialogs.

Upon recognizing the authentication dialog, NSL can read secrets (for example, user IDs and passwords) from SecretStore and fill in the proper fields in those dialogs to complete the authentication process. By default, NSL ships with many scripts that recognize popular applications, thereby helping users get a head start in the process.

Universal connectors such as NSL enhance their performance and roaming capabilities by synchronizing with SecretStore during the user's login session with eDirectory. During this period (known as *connected mode*), NSL uses the SecretStore client to download the user's relevant secrets to a secure and encrypted cache on the client. This process allows NSL to minimize wire access and results in a better performance.

Periodically, the cache and SecretStore are synchronized in the background. As a result, the user can use NSL from different workstations to single sign-on to desired applications and services. If the security policy and configurations in the customer's environment so allow, the secrets cache can be saved to a securely encrypted file when the user's session is terminated. Otherwise, the cache is destroyed upon session completion.

A local copy of SecretStore allows NSL to provide roaming capability for the users in an eDirectory *disconnected mode*. NSL also provides the capability to access applications and single sign-on by using other directory systems which do not have the enhanced security provided by the patented SecretStore technology. As a universal connector, NSL allows synchronization between these other directories and SecretStore as needed.

Upon termination of a roaming period and the start of a new user session with eDirectory, NSL synchronizes its offline cache with SecretStore to bring SecretStore and the local cache up to date.

Proxy portals are another category of universal connectors. Using these, a trusted, secure service can do the following:

- Act as a connector to authenticate the user to the applications and services
- Allow for populating users' SecretStores ahead of time so that the users can subsequently access applications

SecretStore's Single Sign-On Process

SecretStore's single sign-on process requires the universal connectors or the regular connectors to acquire the authentication credentials before or after a user successfully authenticates to the target application or service. These credentials are then stored as secrets in SecretStore.

Clearly, the user must be authenticated to eDirectory before any access to SecretStore is possible. Upon subsequent invocations of applications or requests for services, the universal connectors can read the secrets from SecretStore, pass them to the targets, and bypass any need for user interaction with the application or service.

Figures 3, 4, and 5 describe the process of single sign-on authentication and show how an enabled application can interface with SecretStore, read and write secrets, and authenticate the user.

For purposes of comparison, Figure 3 illustrates how a user might authenticate to a network application that isn't enabled for single sign-on.

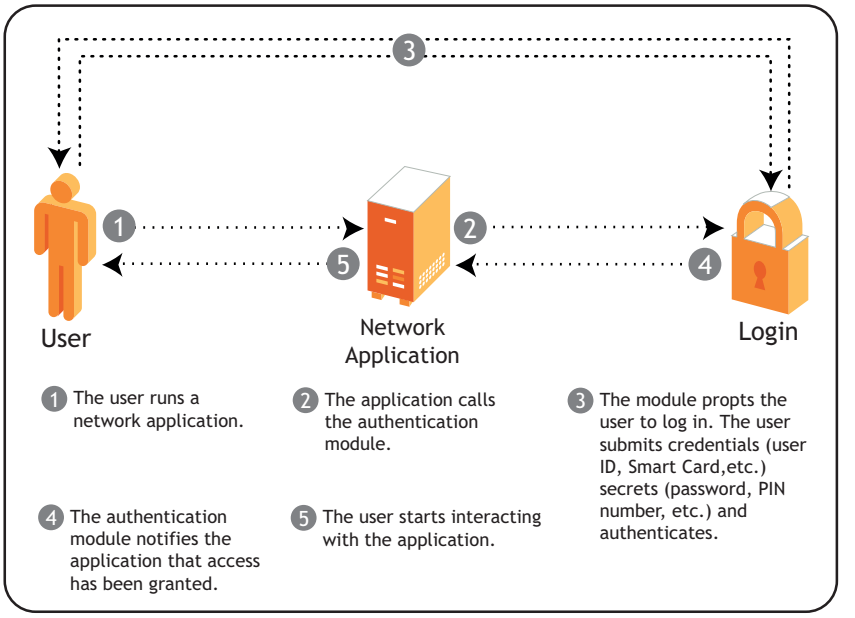

Figure 3: *An application's successful authentication before single sign-on.*

Figure 4 illustrates the first-time authentication to an application that has been enabled for single sign-on.

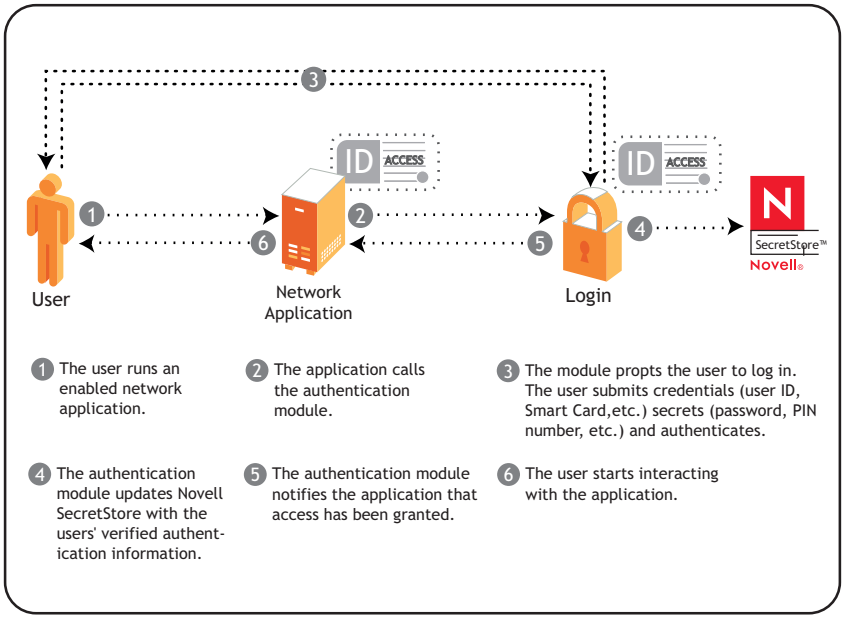

Figure 4: *First-time successful authentication to a single sign-on enabled application.*

Figure 5 illustrates the processes involved in subsequent user authentication to a single sign-on enabled application.

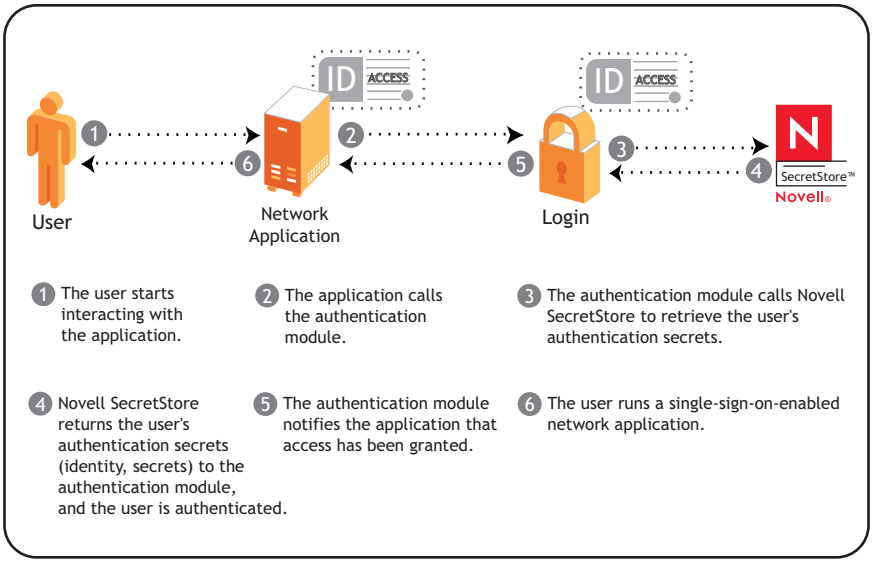

Figure 5: *Subsequent user authentication to a single sign-on enabled application.*

As seen in Figure 5, the interaction of the authentication module with the user has been eliminated. That subsystem interacts with SecretStore to provide single sign-on. A user must be logged in and authenticated to eDirectory before SecretStore can be accessed. The clearances for entities trying to access SecretStore are controlled by *Discretionary* and *Mandatory Access Controls* implemented in eDirectory and their utilization by SecretStore.

New Features in Novell SecretStore

This section highlights two new features in the latest release of Novell SecretStore: enhanced password protection and shared secret format.

Enhanced Protection

In the current eDirectory authentication scheme based on passwords, it is possible for the eDirectory administrator to launch an attack on a user's secrets. The administrator can do this by changing the user's eDirectory password through the available administrative means, logging in as the user with the new password, and attempting to read the secrets in the user's SecretStore.

To prevent such an attack, SecretStore's *enhanced protection* feature can be enabled. Then, using the SecretStore administrative utilities, users can set a user-specific Master Password for SecretStore.

With enhanced protection enabled, a user's SecretStore is locked whenever the eDirectory password is reset. This action protects users from an administrative attack. Once a user's SecretStore is locked, a valid owner can unlock it in one of two ways:

- Provide the previous valid eDirectory password.

  In the case of forgotten passwords, this method will not work—which brings us to the second option.

- Use SecretStore's Master Password.

Note: The Master Password for the user's SecretStore should be set *before* the user starts using SecretStore in enhanced protection mode or *before* SecretStore is locked.

As an example scenario, consider a password reset. Suppose user Markus forgets his eDirectory password. He contacts the network administrator, who resets the forgotten password. Upon the first login with the new eDirectory password, Markus encounters SecretStore's Master Password feature. Once he provides the master password, he regains regular access to network and Web services.

Configuring Enhanced Protection. An administrator can configure SecretStore's enhanced protection feature by using the SecretStore snap-in to ConsoleOne. Using this snap-in, the administrator can define one or more *SecretStore Administrators*. A SecretStore Administrator is different from the regular administrator of the system.

This extended feature of enhanced protection is designed to further improve security in highly-secure environments. With a two-administrator scheme, the eDirectory password is reset by a network administrator, but SecretStore can only be unlocked by a separate SecretStore Administrator. SecretStore configuration allows for defining administrators for different parts of the eDirectory tree.

Through configuration, it is also possible to define security policies such as Graded Authentication labels for applications under the SecretStore Policy container or related policy override hierarchies. The SecretStore Policy container is located within the Security container of the eDirectory tree. SecretStore periodically caches these policies to be applied to the user's access in addition to eDirectory access controls. The SecretStore snap-in to ConsoleOne provides the means for configuring SecretStore policies and behavior in detail.

You may be wondering what would happen if SecretStore is locked due to a forgotten password, but no MasterPasswords were set and no SecretStore administrators were designated ahead of time. In this case, the user can use ConsoleOne or SecretStore Manager to remove the locked secrets, thereby unlocking the SecretStore.

The SecretStore Master Password, providing the previous eDirectory password, and defining SecretStore administrators are various ways to unlock the secrets in a secure way without losing the data.

Shared Secret Format

Because SecretStore is the common security infrastructure used by products such as Novell iChain, Novell Portal Services, and Novell SecureLogin, the need for these secure applications to share user secrets has become crucial.

In the past, it was possible for all of the above applications and services to create similar sets of credentials in SecretStore to authenticate the user to the Novell GroupWise e-mail system. In addition, the GroupWise client running in connector mode could store its own credentials for the user in SecretStore. As a result, the user ended up with four secrets for GroupWise in SecretStore. These secrets, while similar, had different Secret IDs. When using one of these applications, the user could make changes to the credentials in GroupWise (a password change, for instance), and the application would store the updated credentials in SecretStore. This would have caused the secrets for the other three applications to go out of synchronization.

To solve this problem and provide the ability for applications and services to share common data, the Shared Secret format has been introduced in the latest releases of SecretStore. The related interfaces have also been introduced and published to the developer community. Currently, applications that comply with the Shared Secret format using the new SecretStore interfaces can create Shared Secret IDs that other applications can read and write to.

Coupled with the Shared Secret ID, the same interfaces are used to write secrets in the Shared Secret format. This allows applications to share the actual secrets instead of having multiple copies of similar secrets in the store. Shared Secret provides for all of these applications to use and modify the same Shared Secret for the target application. Applications will no longer go out of synchronization, and redundant secrets will not need to be stored in SecretStore.

Conclusion

Novell SecretStore provides a secure, eDirectory-based infrastructure for applications and services to store user authentication secrets. Applications and services are enabled to utilize this service either as connectors or through the use of universal connectors. The most common use of SecretStore is single sign-on. As a key component of Novell's Secure Identity Management provisioning architecture, SecretStore is designed to provide a secure solution for customers who want to take advantage its single sign-on capabilities.

For Additional Information

For the latest information on deploying Novell SecretStore on your network, refer to the installation and configuration manuals supplied on the product CD.

The Novell Developer Kit (NDK) downloads and documentation for SecretStore 3.0.2 are available at the following URLs:

- For C: *http://developer.novell.com/ndk/ssocomp.htm*
- For Java: *http://developer.novell.com/ndk/nssoj.htm*

As of this writing, SecretStore 3.2 code and documentation are available as an early access release.

The following AppNotes and Developer Notes may also be of interest:

- "Understanding Novell's Single Sign-On"
  http://developer.novell.com/research/appnotes/2000/february/02/
- "SecretStore: Novell Single Sign-on Version 1.1"
  http://developer.novell.com/research/devnotes/2000/april/02/d000402.htm
- "SecretStore Single Sign-on"
  http://developer.novell.com/research/devnotes/1999/november/05/

For other AppNotes on security-related topics, refer to the Novell Research Web site at *http://www.novell.com/appnotes*.