

ITProTM
SERIES

WindowsITPro

 **eBooks**

Keeping Your Business
SAFE from Attack:

Encryption and Certificate Services

By Jan De Clercq

Microsoft[®]



Microsoft®

Contents

Chapter 7: Windows PKI-Enabled Applications	143
Encrypting File System	143
The End-User EFS Experience	143
EFS Internals	144
EFS Data Recovery	148
New EFS Features in Windows XP and Windows Server 2003	151
EFS File Sharing	151
Web Folder and WebDAV Integration	153
Offline Files and Folders EFS Support	154
Cryptographic Changes	156
Disabling EFS	157
EFS Recovery Changes	157
Secure Mail Using S/MIME	159
S/MIME Basics	159
Exchange Server S/MIME Support	161
Microsoft Mail Client S/MIME Support	162
Enhanced Security Services	165
Leveraging Smart Cards and USB Tokens for PKI-Enabled Applications	166
Windows Server 2003 and Windows XP Smart Card Support	167
Enrolling for Smart Card-Based Credentials	168
Smart Card Logon	170
Smart Card Management Systems	171
Conclusion	172

Chapter 7:

Windows PKI-Enabled Applications

In the previous chapters, we introduced Windows Server 2003 PKI. In this chapter, we focus on three applications that can leverage your public key infrastructure (PKI) investment: the Encrypting File System (EFS), Secure MIMI (S/MIME) for secure messaging, and smart card-enabled applications.

This chapter does not cover all Windows PKI-enabled applications available today. Other PKI-enabled applications not covered in this eBook are VPN authentication, IP Security (IPSec) authentication, Secure Sockets Layer/Transport Layer Security (SSL/TLS), and code signing.

Encrypting File System

The disclosure of confidential information to unauthorized parties is a serious threat from which any organization should be protected. The EFS, a feature of the Windows 2000, Windows XP, and Windows Server 2003 NTFS file systems, provides file-system-level encryption of files and folders stored on NTFS volumes. Before Windows 2000, NT users had to use the products of other vendors to implement an encryption solution.

The End-User EFS Experience

As you can in Windows 2000, you can manually encrypt files and folders in Windows XP and Windows Server 2003 by selecting the “Encrypt contents to secure data” check box in the Advanced Attributes dialog box, or by choosing the Encrypt command on a file or folder’s shortcut menu. If you set the encryption attribute on the folder level, newly created files in the folder will be automatically encrypted. Unless you select the “Apply changes to this folder, subfolders, and files” check box, files that already existed in the folder before the encryption attribute was set will not be encrypted. The same is true for decryption.

The Encrypt/Decrypt shortcut context-menu option for Windows Explorer is disabled by default. To enable it, add the EncryptionContextMenu value with REG_DWORD data value 1 to the HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced registry key. Enabling this feature adds an encrypt/decrypt option when the user right-clicks a file or folder in Windows Explorer.

The cipher.exe tool can be used to automate and enforce encryption from the command prompt. For example, you can use cipher.exe in a user’s logon or logoff script. In a Windows 2000 and Windows Server 2003 environment, you can use Group Policy Objects (GPOs) to automatically distribute these scripts throughout the enterprise. Table 7-1 lists some sample cipher commands and their effect.

Table 7-1 Cipher switches

Command	Effect
Cipher /E /A /I <path>	Encrypts all files and subfolders in the specified file-system path and ignores possible errors.
Cipher /D /A /I <path>	Decrypts all files and subfolders in the specified file-system path and ignores possible errors.
Cipher /E /A <path>/*.txt <path>/*.bat	Encrypts all .txt and .bat files in the specified file-system paths.
Cipher /E /A /F <path>/*.*	Forces the encryption of all files in the specified file-system path. Files that were encrypted previously are reencrypted.
Cipher <path>	Shows the EFS status (U or E) of all files and/or folders in the specified file-system path.
Cipher /U	Updates all Data Decryption Fields (DDFs) and Data Recovery Fields (DRFs) of encrypted files to reflect the latest user encryption key and the latest recovery-agent recovery keys.

When you are using EFS, do not forget that the account that can encrypt data is not restricted to the owner of the data. To encrypt a file, you don't need file or folder ownership, but you need at least read/write permission to the file or folder.

EFS Internals

The software technology behind EFS is a good example of a hybrid cryptographic solution that combines the power of both asymmetric and symmetric ciphers. EFS uses a symmetric cipher (Advanced Encryption Standard [AES], Triple DES [3DES], or DESX Data Encryption Standard X [DESX]) to perform the bulk encryption, and an asymmetric cipher (RSA) to provide secure storage of the bulk encryption key. AES uses a 256-bit symmetric encryption key, 3DES uses a 128-bit key, and DESX uses a 56-bit key. AES support is available only on Windows XP Service Pack 1 (SP1) (and later) and Windows Server 2003. By default, Windows 2000 and Windows XP EFS use DESX for bulk encryption. If you have the High Encryption Pack installed, EFS will use 3DES. Later in this chapter (in the “New EFS Features in Windows XP and Windows Server 2003” section), we explain how you can configure EFS to use AES, 3DES, or DESX. The bulk encryption key is known as the file encryption key (FEK). Figures 7-1 and 7-2 illustrate the EFS operation.

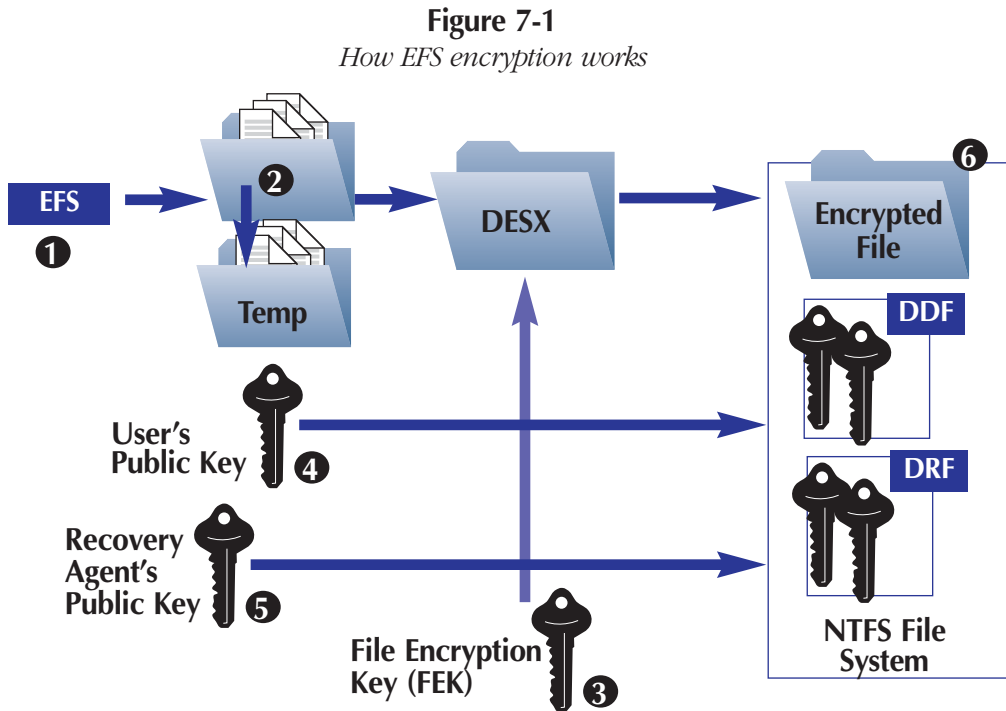


Figure 7-1 illustrates an EFS encryption operation:

1. The EFS service opens the file for exclusive access.
2. The file data is copied to a temporary file for recovery purposes.
3. Using a symmetric key cipher (in this example, DESX), an FEK is randomly generated and used to encrypt the file data blocks.
4. A DDF is created that contains the FEK encrypted with the user's public key. If multiple users can access the encrypted file (a feature that is supported only in Windows XP and Windows Server 2003), a DDF is created for each user.
5. A DRF is created that contains the FEK, this time encrypted, using the recovery agent's public key (this operation is done for every recovery agent).
6. The EFS service writes the encrypted data, along with the DDF and DRF, back to the file.
7. The temporary file is deleted from disk.

Figure 7-2
How EFS decryption works

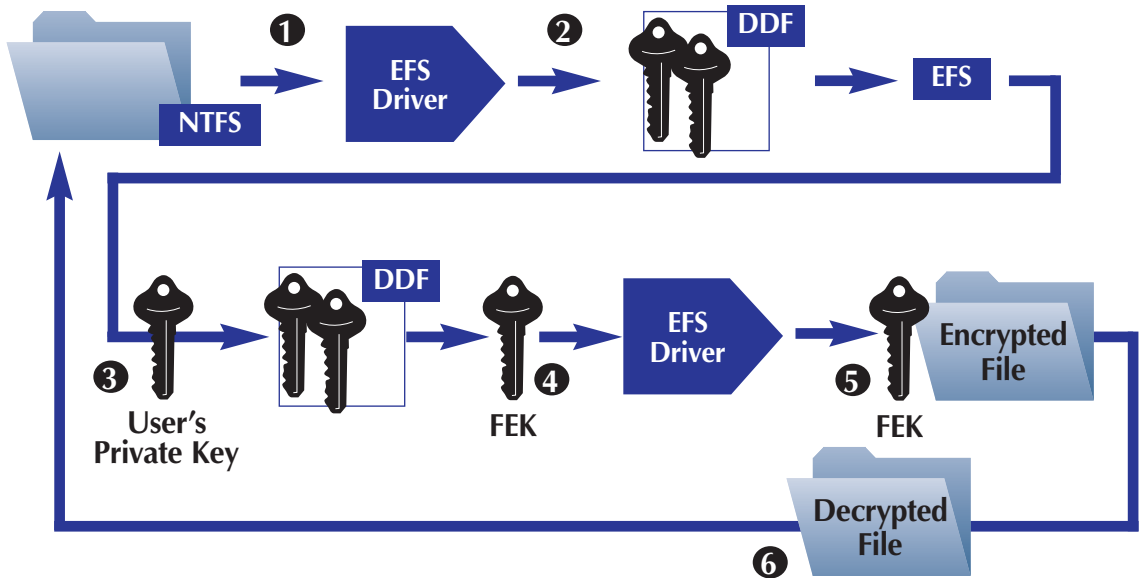


Figure 7-2 illustrates an EFS decryption operation:

1. When an application accesses an encrypted file, NTFS recognizes that the file was encrypted and sends a request to the EFS driver.
2. The EFS driver retrieves the DDF and passes it to the LSA.
3. The LSA decrypts the DDF with the user's private key to obtain the FEK.
4. The LSA passes the FEK back to the EFS driver.
5. The EFS driver uses the FEK to decrypt the file.
6. The EFS driver returns the decrypted data to NTFS, which then sends the data to the requesting application.

The processes behind EFS decryption and EFS recovery are almost identical. The only difference is that the data owner's private key is used for decryption of the FEK (as Figure 7-2 illustrates). For recovery, the data recovery agent's private key is used.

The encrypted FEK is stored, along with every encrypted file or folder, in the NTFS \$EFS attribute. The \$EFS attribute is stored in a file's Logged Tool Stream attribute (\$Logged_UTILITY_Stream). Changes to this attribute are logged in the NTFS change log. EFS encrypts an NTFS file's content, contained in the unnamed "\$data" stream, and also every additional NTFS file stream.

EFS stores multiple encrypted versions of the FEK: one version for the account that encrypted the information, one version for each account that needs access to the encrypted file (using the EFS file sharing feature—available only in Windows XP and Windows Server 2003), and one version for every recovery agent. The first two versions are stored in the Data Decryption Field (DDF) and the last one is stored in the Data Recovery Field (DRF).

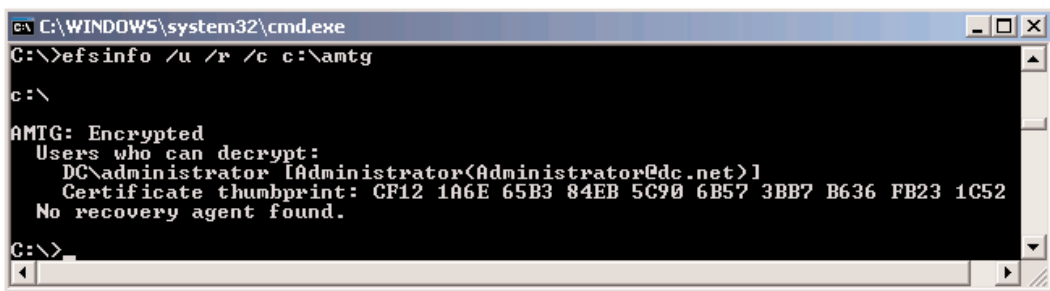
In addition to the encrypted bulk encryption key, DDFs and DRFs contain information that facilitates locating the account's private key that is needed for the decryption of the FEK:

- the account's distinguished name
- the Cryptographic Service Provider (CSP) used for encryption/ decryption
- the CSP's storage location for the certificates (certificate store container)
- the certificate thumbprint

To retrieve a private key from a user's private key store, the Local Security Authority (LSA) will pass the certificate's thumbprint and certificate store to the CSP. The \$EFS file attribute (which contains a file's DDFs and DRFs) also contains a checksum that the EFS system uses to detect integrity changes on the DDF and DRF levels. This design protects against EFS data tampering on the file-system level.

To look at the content of the DDF and DRF fields, you can use the EFSinfo tool or the EFSdump tool. The EFSinfo tool comes with the Windows 2000 and Windows Server 2003 Resource Kits. The EFSdump tool is available from the Sysinternals Web site at <http://www.sysinternals.com>. EFSdump displays the contents of the DDF and DRF fields. For each entry, the tool shows the account name and the Subject's Distinguished Name. With EFSinfo, you can do even more. Two interesting EFSinfo switches are /C and /Y. The /C switch displays the certificate thumbprints of the EFS encryption and recovery certificates; the /Y switch displays the thumbprint of a user's local EFS certificate. To look at both the DDF and DRF fields and the associated certificate thumbprints, use EFSinfo /u /r /c <filename> (as Figure 7-3 illustrates).

Figure 7-3
Using EFSinfo



```

C:\WINDOWS\system32\cmd.exe
C:\>efsinfo /u /r /c c:\amtg

c:\
AMTG: Encrypted
Users who can decrypt:
DC\administrator [Administrator@Administrator@dc.net]
Certificate thumbprint: CF12 1A6E 65B3 84EB 5C90 6B57 3BB7 B636 FB23 1C52
No recovery agent found.

C:\>

```

All EFS operations (encryption, decryption, and recovery) are fault tolerant and are logged. During an EFS operation, a hidden log file (named efs0.log) is created in the hidden system volume information folder. In addition to a log file, the EFS system creates a temporary backup (named efs0.tmp) of the file being encrypted in the file's directory. If your system crashes during an EFS encryption operation, the EFS operation will be rolled back, and the original, possibly corrupted, file will be replaced with the backup file. To look at what is happening behind the scene on the file system level when you are using EFS, you can use Sysinternals' Filemon tool (available from the company's Web site).

For security reasons, the FEK is never paged to disk. Doing so could make the EFS decryption key available to malicious code that is accessing and scanning a user system's disks. Some applica-

tions that deal with an encrypted file, however might copy some of the cleartext to the paging file. Because the paging file is a system file and thus cannot be encrypted, Microsoft advises users to clear the paging file at system shutdown. You can automate this process by setting the GPO setting “Clear virtual memory page file when system shuts down.”

The generation of a private-public key pair and a special EFS certificate happens transparently. When users select the encrypted property, choose the encrypt option in the context menu, or encrypt a file or folder using the cipher command-prompt tool, the system can automatically generate a private-public key pair and send a public-key certification request to a Windows 2000 or Windows Server 2003 enterprise Certification Authority (CA). If no CA is available, a self-signed certificate will be generated by a CSP on the local machine. This arrangement enables EFS to function even in the absence of a CA. If a valid EFS certificate already exists, EFS uses the public-private key pair associated with this certificate.

Once the EFS certificate is created or downloaded to the local certificate store, a reference to it (the certificate hash) is put in HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\EFS\CurrentKeys\CertificateHash. You can retrieve this value from the registry using the EFSinfo tool with the /Y switch. This registry entry is used to remind EFS which EFS certificate and associated public-private key pair should be used for EFS operations.

The following certificate templates can be used for EFS operations: *user*, *administrator*, and *basic EFS*. EFS certificates can be issued by a Windows 2000 or Windows Server 2003 standalone or enterprise CA. The Windows 2003 Enterprise Edition and Datacenter Edition enterprise CAs can create EFS certificates that can be archived. EFS certificates can also be generated by third-party, non-Windows CAs. How to do this is explained in the Microsoft Knowledge Base article Q273856, available from <http://support.microsoft.com/default.aspx?scid=kb;en-us;273856>.

Just like any other personal certificate, EFS certificates (and, with them, the EFS public keys) and EFS private keys are stored in the user's profile. You can find more information on the storage of certificates and private keys in Chapter 1 of this eBook. Even though theoretically EFS certificates and private keys can be stored on a smart card, EFS cannot access them when they are stored there. This limitation exists because EFS is hard coded to use only the Microsoft Base, Enhanced, or Strong CSPs (see Chapter 1).

EFS Data Recovery

A key feature of EFS is its ability to recover encrypted files or folders when a user's private key is lost or becomes inaccessible. Loss of private keys can occur because of hardware or software problems on a user's computer. EFS data recovery can also be useful when an employee leaves the company, when his or her account and profile are deleted, or when access is needed to files previously encrypted by that employee.

In addition to EFS data recovery, the Windows Server 2003 CA also can provide EFS key recovery. We explained this feature in Chapter 5.

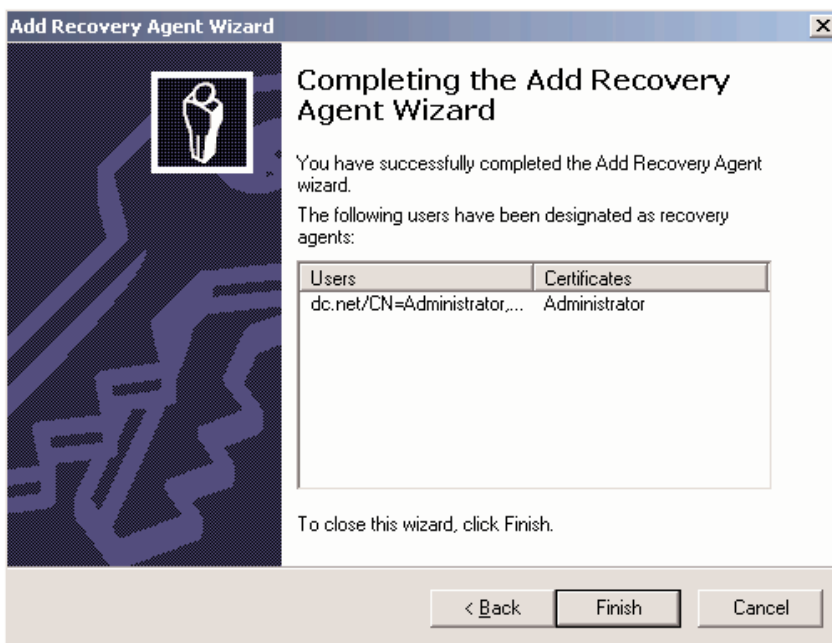
EFS data recovery means that a Windows account known as the *recovery agent*, which is other than the original account that encrypted the data, can decrypt the FEK and recover the user's data. The recovery agent can recover a file if his or her public key was used to encrypt the FEK and the resulting encrypted FEK is part of the encrypted file's NTFS \$EFS attribute—in other words, when the account was defined as a recovery agent at the moment the file was encrypted. A recovery agent

needs to have a special key pair and a special certificate (an EFS recovery certificate) before the EFS encryption process takes place.

EFS recovery agents can be defined on the *domain, site, organizational unit, or local machine level*. You use the Windows 2000 or Windows Server 2003 GPO Encrypting File System entry to define them. This entry is located in the Computer Configuration\Windows Settings\Security Settings\Public Key Policies GPO container. The EFS GPO entry contains the EFS recovery certificates of Windows accounts that are designated as recovery agents by the GPO administrator. To facilitate the life of administrators, Microsoft provides a wizard to set the recovery agents (illustrated in Figure 7-4). With the wizard, the recovery certificates can be downloaded from the Active Directory (AD), or AD can be imported in *.cer format. You must publish the certificates manually to AD to make them available from AD. By default, the Windows CA does not automatically publish EFS certificates in AD (unless you create a new certificate template that is explicitly configured to do so).

Figure 7-4

Setting up an EFS recovery agent using GPOs



EFS recovery certificates have File Recovery (OID 1.3.6.1.4.1.311. 10.3.4.1) in the Enhanced Key Usage field. These certificates can be generated by an enterprise CA or a standalone CA. They can even be generated without the intervention of any CA: The recovery certificates that are generated on a standalone machine are self-signed recovery certificates for the local administrator, generated by the local machine. To manually generate a self-signed EFS recovery-agent certificate and private key on a Windows XP or Windows Server 2003 standalone machine, use the cipher command with the /r switch.

Enabling EFS recovery does not necessarily require a CA. However, within an AD environment, the use of a CA for the generation of EFS recovery certificates offers more flexibility and centralized control over EFS recovery:

- An organization can provide EFS recovery privileges (in the form of an EFS recovery certificate) to specific administrator accounts.
- An organization can control the validity of an administrator's EFS recovery privilege because it has control over the lifetime and revocation status of EFS recovery certificates.

In Windows 2000 EFS, the default recovery agent is the administrator account (an interesting note here: The account's private key is in the administrator profile that is stored on the first domain controller installed in the domain). On a standalone machine, this is the local administrator account. On a machine member of a domain, the default recovery agent is the domain administrator account. A best practice is to change the default domain-recovery account administrator to some other account. Windows Server 2003 EFS contains some important changes regarding EFS recovery, which we explain next.

The EFS recovery GPO settings are downloaded to the machine at startup, at the predefined GPO update interval, or when GPO application is enforced. Changes in the policy's content are not applied immediately to every encrypted file or folder in the policy scope. EFS will enforce and apply the change the first time it is used following the policy change.

Remember that to define a new DRF, EFS needs access to the FEK, which is available only at decryption time. This requirement explains why you must always archive the private keys and certificates of old recovery agents. By doing so, you will always be able to access any encrypted file, even those that have not been opened since an EFS-recovery policy change occurred. To archive a recovery agent's private key and certificate, export them using the Certificate Export Wizard. To find out which archived keying material you should use for a recovery, use the EFSinfo tool with the /C switch to retrieve the recovery agent's certificate thumbprint from the encrypted file.

You must take special care of the place where the archived recovery-agent private keys are stored, to protect them from unauthorized access. Anyone who possesses the recovery private key can read any encrypted file or folder within the EFS recovery-policy scope. A best practice is to put the private key on a floppy disk, a USB drive, or a CD, and then lock it in a safe. Also, when you request a recovery certificate using the CA Web interface, be sure to select "Enable strong private key protection" in the advanced request settings to provide another level of software protection. You can use this feature to prompt the user for a special password every time the private key is accessed.

Besides archiving the recovery agent's private key, you should also delete it from the local system. "Delete the private key if export is successful" is an option that you can set in the Certificate Export Wizard. The reason to do this relates to a fundamental security principle: Encryption keys should never be stored near the files they secure. This is a must-do for standalone machines; it is less critical for machines in a domain, where the recovery agent's private key is almost certainly located on another machine. A less secure alternative is to use a dedicated, highly secure workstation, preferably off the network, that has a local copy of the recovery private key and that is used only for EFS recovery.

The preferred procedure to recover an encrypted file is the following:

- Let the user back up the file using NTBackup or any third-party backup solution that uses the NTBackup APIs (this is the only way that an encrypted file can be exported from a Windows system without decrypting it).
- Let the user send the file to a recovery agent, using a secure channel.
- Let the recovery agent do the recovery.
- Let the recovery agent send the decrypted file back to the user, once more using a secure channel. A secure channel can be created by delivering the file out-of-band using a floppy disc, by copying the file across a VPN tunnel to a shared folder, or by sending the file using an S/MIME mail message.

Another way, less secure and thus less preferred, is to use a dedicated machine to perform the recovery of encrypted files.

New EFS Features in Windows XP and Windows Server 2003

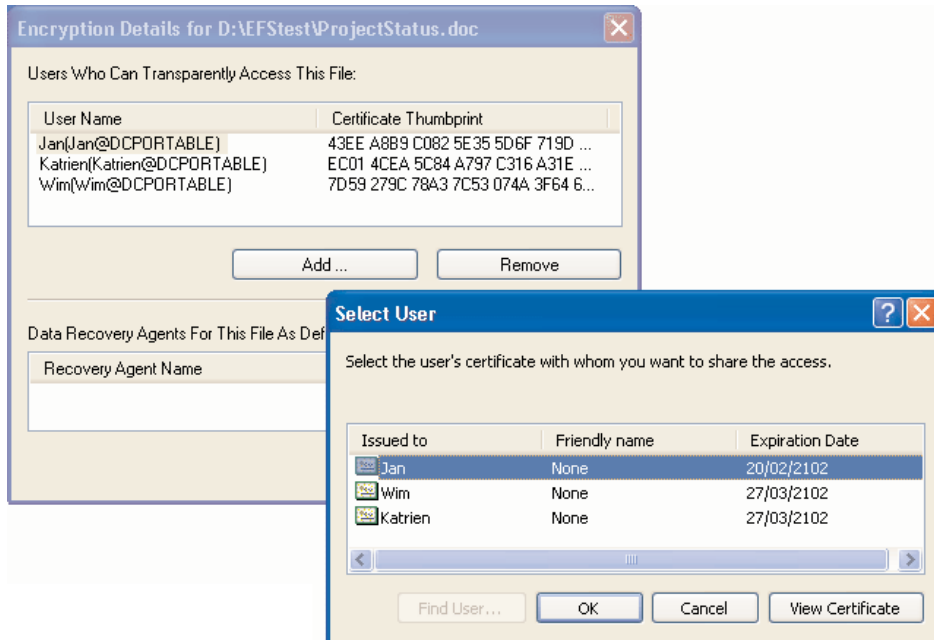
The following sections provide an overview of the new EFS features in Windows XP and Windows Server 2003 EFS.

EFS File Sharing

Perhaps the most visible change to Windows XP and Windows Server 2003 EFS is the support for EFS file sharing, which lets users share encrypted files with other users. From an administration point of view, things would certainly have been much simpler if Microsoft had let users share their encrypted files with Windows groups. But because EFS relies on X.509-based certificates that, by definition, honor the principle of individual accountability, that approach is impossible. EFS file sharing applies only to NTFS files, not to NTFS folders.

EFS file sharing is enabled in a file's advanced properties. If a file is encrypted, you will notice that the Details button in the file's Advanced Attributes dialog box becomes available. Clicking this button brings up the dialog box illustrated in Figure 7-5, which lets you share an encrypted file with other users. In Figure 7-5, you can see that the projectstatus.doc file is encrypted and shared between users Jan, Katrien, and Wim. The sharing of an EFS encrypted file is not an explicit privilege of the user account that encrypted the file and shared it with another user. In the Figure 7-5 example, Jan may have encrypted the file and decided to share it with Katrien. Katrien, in turn, may then have decided to share the file with Wim.

Figure 7-5
Setting up EFS file sharing



Let's look now at how EFS determines whether a user is authorized to change a file's EFS-sharing properties, and with whom he or she can share the file. To change a file's EFS-sharing properties, you need at least write permission to the file. To share a file with another user, you also need access to that user's EFS encryption certificate. From the Select User dialog box (illustrated in Figure 7-5), you can access the user certificates stored in the Other People and Trusted People certificate containers of your certificate store. The Trusted People certificate-store container is used for self-signed certificates that do not chain to a trusted root CA. The Other People certificate-store container is the general cache for certificates that do chain to a trusted root CA. If your machine is a member of a Windows AD-based domain, you will notice that the Find User... pushbutton also lights up. Clicking this button lets you access the EFS user certificates published in AD. If you want to share encrypted files with people whose EFS certificates are not available in one of the above repositories, you can always import their certificates manually into your certificate store.

To better understand how EFS file sharing really works, it is worth looking at the cryptographic nuts and bolts of EFS file sharing. When a user shares an encrypted file with another user, an extra DDF is added to the file's EFS-related NTFS file streams. At no point in the EFS file-sharing process is the file itself decrypted. When Jan decides to share with Katrien a file he previously encrypted, EFS will decrypt Jan's DDF with Jan's private key in order to retrieve the FEK. The EFS system will then encrypt the FEK using Katrien's public key, and add the resulting DDF to the file's EFS-related NTFS file streams.

Web Folder and WebDAV Integration

An important Windows 2000 EFS design limitation surfaces when EFS is used to protect the confidentiality of files stored on file servers. The key problem here is that files are always decrypted locally on the file server, and then transmitted in the clear to the user workstation. This setup also requires the file server to be trusted for delegation, and to have access to a local copy of the user profile. The latter are consequences of the fact that EFS needs access to a user's private key (which is stored in the user profile) to decrypt an encrypted EFS file.

Windows XP and the Windows Server 2003 Web server (IIS 6.0) come with an interesting alternative for the use of EFS on file servers. In Windows XP and Windows Server 2003, Microsoft embedded support for the transport of EFS metadata using the WebDAV protocol. WebDAV stands for Web Distributed Authoring and Versioning. WebDAV is an extension of the HTTP 1.1 protocol that allows for file metadata to be transported together with the file across an HTTP connection. Support for the WebDAV protocol has been available on the client side beginning with Internet Explorer (IE) 5.0 (through the Web Authoring Components) and on the server side from Microsoft Internet Information Services (IIS) 5.0 on.

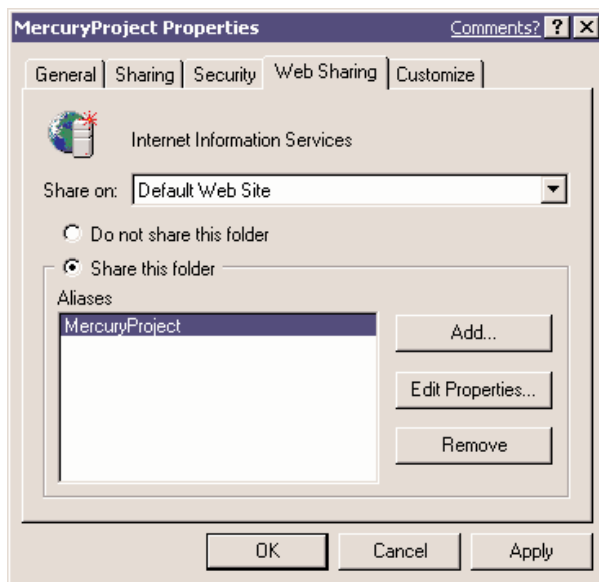
The WebDAV support makes Web folders a worthy alternative to file shares for the use of EFS on file servers. That is also why Microsoft strongly recommends the Web-folder solution instead of file shares for remote storage of encrypted data on file servers. The combination of EFS file sharing, Web folders, and WebDAV makes the Windows Server 2003 platform a very interesting option for the setup of secure file servers. Table 7-2 compares the features of using file shares for remote EFS operations to those of using Web folders.

Table 7-2 Comparison between the features of remote EFS operations on file shares and on Web folders

Remote EFS Operations On...	...File Shares	...Web Folders
Where does EFS encryption/decryption occur?	Files are encrypted and decrypted on the file server.	Files are encrypted and decrypted on the user machine.
Are the files secured during transmission over the network?	Files are sent in the clear over the network connection.	Files remain encrypted while being sent over the network connection.
What technology is or can be used to secure the transmission of files over the network?	Requires IPsec to secure the file transfer between file server and user machine.	Does not require IPsec to secure the file transfer; relies on the WebDAV EFS extensions to securely transmit the file.
Must the file server be "trusted for delegation"?	Requires file server to be "trusted for delegation."	Does not require file server to be "trusted for delegation."
Does the solution require a copy of the user profile on the file server?	Requires availability of user profile on the file server (local or roaming profile).	Does not require availability of user profile on the file server.
Where does the EFS file-sharing authorization process for users take place?	EFS checks for other user certificates on the file server and/or in AD.	EFS checks for other user certificates on the local machine and/or in AD.
What file-transport protocol/method is used?	File sharing uses Server Message Block (SMB). SMB supports streaming and block-by-block writeback, which is much more efficient and performs better than HTTP-based downloading.	Web-folder-based sharing uses HTTP. HTTP requires that the entire file be uploaded or downloaded before an application can have access to it.

Setting up a Web folder is relatively easy. Open up the file's Properties dialog box, select the Web Sharing tab, and select "Share this folder" (as Figure 7-6 illustrates).

Figure 7-6
Setting up a Web folder



Offline Files and Folders EFS Support

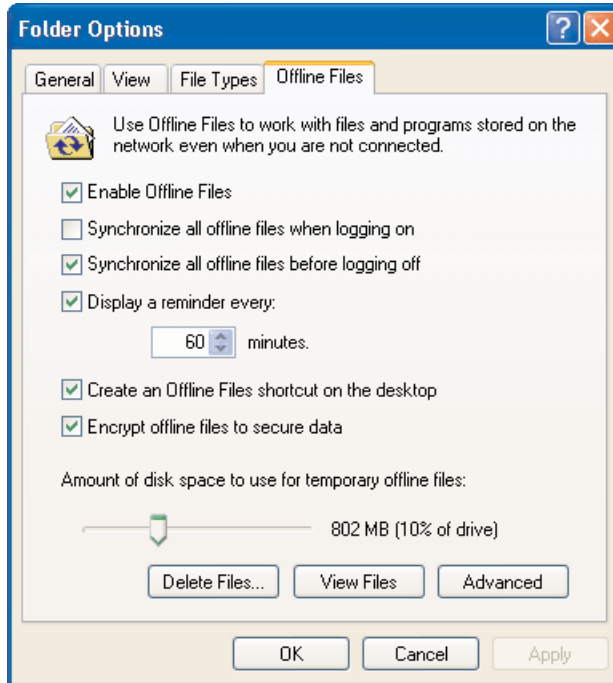
Windows 2000 introduced the concept of offline files and folders. Offline files and folders enable users to access files and folders on network shares even when the user's machine is disconnected from the network. To support this feature, the offline files and folders system keeps a local copy of the files and folders shared on the remote machine and—even more importantly—keeps the remote and local copy of the data synchronized. All offline files and folders are stored in a Client Side Caching (CSC) database, which is located in the hidden %systemroot%/CSC folder. In Windows 2000, offline files and folders do not preserve their EFS encryption attributes when they are copied to the local machine. Also, it was not possible to encrypt the CSC database in Windows 2000.

From Windows XP and Windows Server 2003 onward, the EFS encryption attributes are preserved when a remote file or folder is made available offline. Also, the local client-side cache database can be encrypted.

To enable the encryption of the client-side cache database:

1. Open up My Computer or Internet Explorer, and select Folder Options from the Tools menu.
2. On the Offline Files tab, first enable offline files by selecting the "Enable offline files" check box, and then select the "Encrypt offline files to secure data" check box (as Figure 7-7 illustrates).

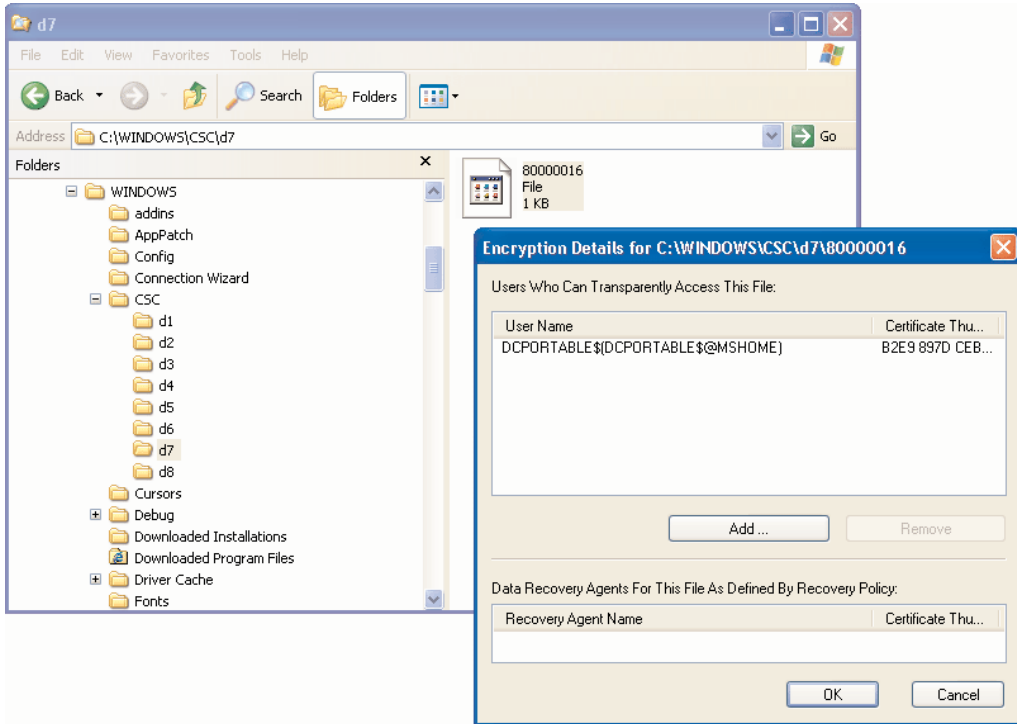
Figure 7-7
Enabling EFS encryption for offline files and folders



When you set this option, the offline files in the CSC database will be encrypted using a self-signed EFS certificate issued to <local_machinename>\$ (as Figure 7-8 illustrates).

Figure 7-8

Viewing the Encryption Details dialog box for the offline files and folders CSC database



Preservation of the EFS encryption settings happens automatically when you make a file or folder available offline. To make a file or folder available offline, right-click the original file on the remote share, and then select “Make Available Offline.” An important detail to be aware of is that offline files and folders functionality is not available when the XP “Fast User Switching” feature is enabled.

Cryptographic Changes

Windows XP and Windows Server 2003 both can be configured to use the 3DES algorithm for EFS encryption and decryption operations. 3DES is compliant with the U.S. Federal Information Processing Security Standards (FIPS) 140-1 Level 1; it uses a 128-bit symmetric encryption key. The default encryption algorithm used by EFS in Windows 2000 and XP is 56-bit DESX. If you have the High Encryption Pack installed, EFS defaults to 128-bit 3DES. The default encryption algorithm used in Windows XP SP1 and Windows Server 2003 is AES with a 256-bit key. Even after you set up Windows 2000, Windows Server 2003, and Windows XP to use 3DES, DESX or AES support (depending on the platform) is not lost. The platforms can still process files previously encrypted with 3DES, DESX, or AES.

There are two ways to configure EFS to support 3DES instead of DESX. The first way is to use a new GPO setting called “System Cryptography: Use FIPS compliant algorithms for encryption.” This setting affects not only the encryption-decryption operations of EFS, but also those of other Windows

security solutions such as IPSec. The GPO setting is located in the Computer configuration\Windows Settings\Security Settings\Local Policies\Security Options GPO container. The second way is based on a registry change and makes the use of 3DES available only to EFS. To make this change, create the AlgorithmID (REG_DWORD) registry value in the HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\EFS registry key, and set the value to 0x6603. As Table 7-3 indicates, you also can use this registry key to force the use of DESX or AES for EFS operations.

Table 7-3 Registry values for EFS symmetric encryption algorithm

Supported Algorithm	Registry	Value Comments
DESX (56-bit key)	0x6604	This value can be used on all versions of Windows 2000 and Windows XP.
3DES (128-bit key)	0x6603	This value can be used on Windows XP, Windows 2000, and later OS versions.
AES_256 (256-bit key)	0x6610	This is the default value. It is compatible with only Windows XP SP1 and later.

Windows Server 2003 also supports the specification of larger default RSA key sizes for keys that are generated for EFS. The default key size used in Windows XP and Windows Server 2003 is 1024 bits. In Windows Server 2003, you can change the default size by setting the following registry key (REG_DWORD): HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\EFS\RSAKeyLength. The value in this key must be a power of 2 (e.g., 1024, 16384...).

Disabling EFS

In Windows Server 2003, you can disable the use of EFS by using a GPO setting that is available from the Properties dialog box of the Encrypting File System GPO container (Computer Configuration\Windows Settings\Security Settings\Public Key Policies\Encrypting File System). To disable the use of EFS for the users in a domain, site, or organizational unit (OU), or on a local machine, clear the “Allow users to encrypt files using Encrypting File System (EFS)” setting. This setting works only for Windows XP and later machines.

As in Windows 2000, if you want to disable EFS on the file or folder level, you can still do the following in Windows XP and Windows Server 2003:

- To disable EFS on the file level, set the system attribute.
- To disable EFS on the folder level, set the system attribute, or create a special entry in the folder’s desktop.ini file: Add `disable=1` in the (Encryption) section.

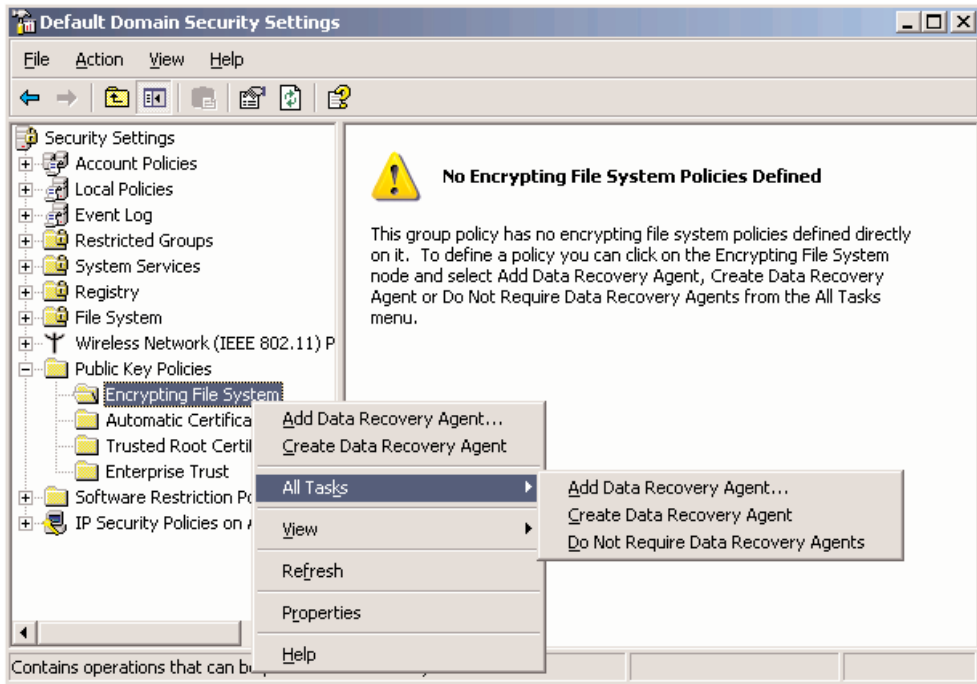
EFS Recovery Changes

As we explained earlier, a unique feature of EFS is its support for data recovery and the facility for designated administrator accounts to decrypt user data. Some organizations, however, do not want an encryption solution that has recovery capabilities. This preference might be because of advanced confidentiality requirements or because organizations simply do not trust the EFS recovery feature.

This requirement for encryption without recovery capabilities is why Microsoft has implemented EFS recovery slightly differently in Windows XP and Windows Server 2003. In both operating systems, EFS can function without the need to have an EFS recovery agent defined (as was the case in Windows 2000). Also, by default, Windows XP and Windows Server 2003 standalone machines, and Windows Server 2003 domains do not have any recovery agents defined. In Windows 2000, the local administrator account was the default local recovery agent, and the domain administrator was the default domain recovery agent. You can enable this new EFS recovery option from the MMC Group

Policy Object snap-in by right-clicking the Encrypting File System container and selecting “Do Not Require Data Recovery Agents.” This sequence is illustrated in Figure 7-9 for Windows Server 2003; most of these options are also available for Windows XP. An option that is not available in XP is the Create Data Recovery Agent option.

Figure 7-9
Defining an EFS data recovery policy



Unlike the Add Data Recovery Agent option, the Create Data Recovery Agent option does not use the Recovery Agent Wizard. Instead, it automatically requests and installs an EFS recovery certificate for the currently logged-on user.

To enable all of these EFS recovery changes, Microsoft has changed the key hierarchy that it uses to protect the EFS private encryption keys in Windows XP and Windows Server 2003. EFS private keys are stored in a user's profile and are cryptographically protected using a key called the master key. The master key is securely stored using a key derived from a user's credentials. Like private keys, master keys are stored in a user's profile. By default, Windows XP and Windows Server 2003 also provide another level of cryptographic protection for security information such as private keys: the syskey option. To learn more about this option, read the following Microsoft Knowledge Base article: <http://support.microsoft.com/default.aspx?scid=kb;en-us;310105>.

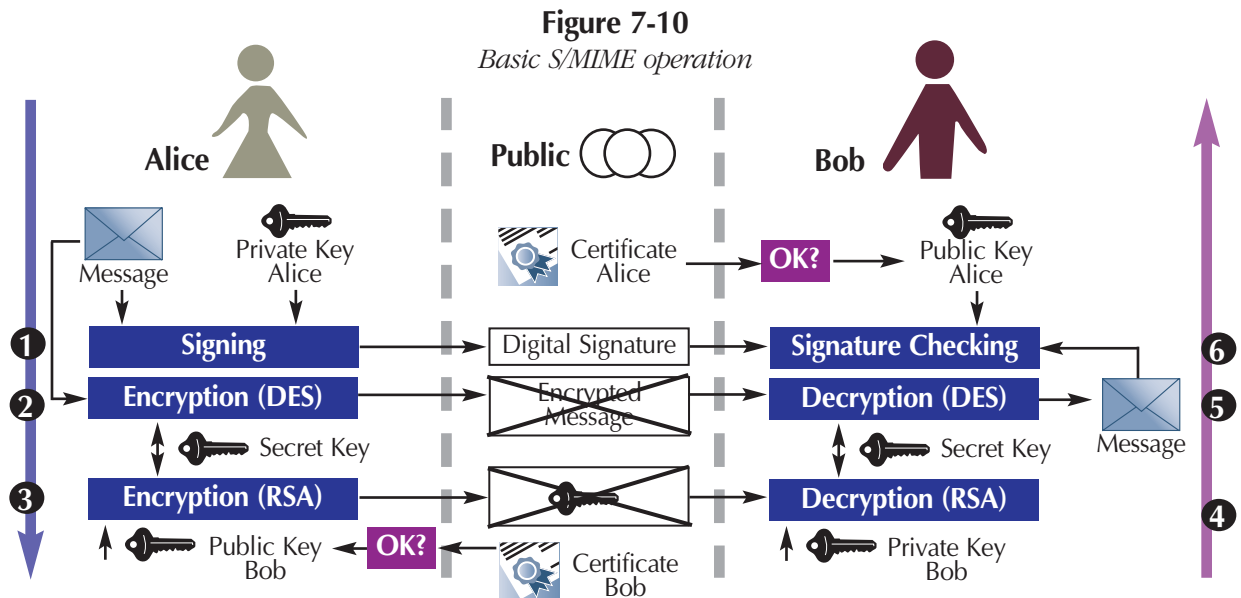
Secure Mail Using S/MIME

In this section, we look at how Microsoft has built S/MIME support into Microsoft Exchange Server 2003, Outlook Web Access, Outlook Express 6.0, and Outlook 2003, and how this support can be integrated with Windows Server 2003 PKI.

S/MIME Basics

S/MIME is an Internet standard for secure messaging. S/MIME can add data authentication, confidentiality, nonrepudiation, and integrity protection to MIME-formatted messages. The Internet Engineering Task Force (IETF) standardized S/MIME version 3.0 in RFCs 2632 through 2634.

S/MIME is an excellent example of a hybrid cryptographic solution. The basic S/MIME operation is illustrated in Figure 7-10. In a typical S/MIME scenario, Alice wants to send a secure message to Bob. “Secure” in this context means guaranteeing confidentiality, integrity, data authentication, and nonrepudiation.



We can split the S/MIME exchange illustrated in Figure 7-10 into six steps, as follows:

1. Using her private key, Alice creates a digital signature for the message.
2. Alice encrypts the message using a bulk encryption key.
3. To create a secure channel that protects the confidentiality of the encryption key, Alice encrypts the encryption key with Bob's public key, which results in a lockbox.
4. Bob decrypts the lockbox using his private key, which results in the bulk encryption key.
5. Using the bulk encryption key, Bob decrypts the message, which gives him the readable message.

6. Bob uses Alice's public key to verify the authenticity and integrity of the message by verifying the digital signature (this verification includes both asymmetric and hashing cryptographic operations).

S/MIME builds on MIME. Thanks to MIME, which the IETF defined in RFCs 2045 through 2049, the body of a mail message can contain data types other than just flat ASCII. An Internet mail message typically consists of a message header, which contains sender and recipient information, and an optional message body. You can use MIME to add nontextual objects, such as images, audio, formatted text, and Microsoft Word documents, to messages. MIME terminology refers to a data type as a content type. A multipart content type lets you embed different content types into a single message body. In a multipart body, boundaries mark the beginning and end of each content type.

S/MIME provides security extensions that let MIME entities encapsulate security objects, such as digital signatures and encrypted message blobs. To do this, S/MIME adds new MIME content types that provide data confidentiality, integrity protection, nonrepudiation, and authentication services. These content types are called `application/pkcs7-MIME`, `multipart/signed`, and `application/pkcs7-signature`. The MIME content type provides information about the type of data a message carries in its MIME attachments. With this information, an application (in this case, the mail client's S/MIME logic) can know how it must process the MIME data before it displays them to a user.

As Table 7-4 shows, the S/MIME attachment's extension differs depending upon the S/MIME service the content type provides. The MIME header specifies the name of the MIME attachment. Some mail clients, such as clients of non-S/MIME-enabled systems, or early versions of S/MIME, need the attachment to recognize a message's S/MIME content. Other mail clients rely completely on the content-type information in the message header to identify MIME entities. S/MIME secures only the message body. Header information must remain unencrypted for messages to relay successfully across gateways on the path between sender and recipient.

Table 7-4 S/MIME content types and services

MIME Content Type	MIME Subtype	S/MIME Type	S/MIME Service	Attachment Extension
Application	pkcs7-MIME	Signed data Enveloped data	Guarantees data integrity, data authentication, and nonrepudiation. Uses opaque signing. Guarantees data confidentiality.	.p7m
Multipart	signed	N/A	Guarantees data integrity, data authentication, and nonrepudiation. Uses clear signing.	N/A
Application	pkcs7-signature	N/A	N/A	.p7s

Exchange Server S/MIME Support

From version 4 onward, Microsoft Exchange Server has included the Advanced Security subsystem to let users secure their mail messages using S/MIME. Microsoft built Advanced Security around the optional Key Management Service (KMS). This is still true for Exchange 2000, but it is no longer true for Exchange 2003. In Exchange 2003, the KMS functionality has been merged into the Windows Server 2003 CA.

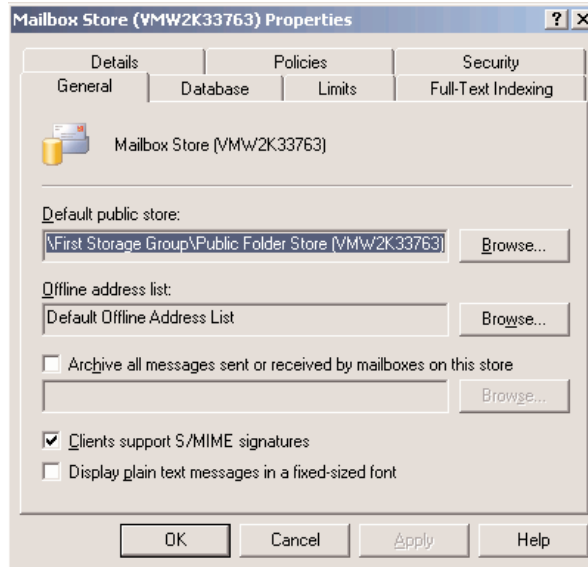
In its early days, the KMS was a CA, a registration authority, and also the entity taking care of key archival and recovery. From Exchange 5.5 SP1 onward, you can outsource the KMS CA functionality to a Windows CA. In Exchange 2003, the CA and archival and recovery functions have been moved to the Windows Server 2003 CA. From Exchange 2000 onward, Exchange's S/MIME functionality is also tightly integrated with AD. AD stores user certificates and certificate revocation lists (CRLs).

Exchange Advanced Security has always included a key-recovery feature that lets you recover copies of users' lost or deleted encryption keys. In Exchange Advanced Security, key recovery is server oriented. The KMS database contains copies of each Advanced Security-enabled user's current and previous private encryption keys. In an Exchange 2003 environment, Windows Server 2003 CA provides the key-recovery function. We explained the key archival and recovery function of the Windows 2003 CA in Chapter 5.

The presence of a key-recovery system explains why Exchange KMS and S/MIME in general use a dual-key-pair system. In a dual-key-pair system, users have one key pair for encryption operations and another pair for signing operations. Exchange Advanced Security and S/MIME could not guarantee trustworthy digital signature services if they used a single key pair and stored the pair's private key in a central database for key-recovery purposes. Digital signatures require that only users can access their private signing keys (otherwise, anyone could impersonate the user). Therefore, Exchange Server stores the signing pair's private key only on the client side. Note that not all S/MIME product vendors support the dual-key-pair system.

Because the KMS does not exist any more in Exchange 2003, you can configure very little when you are setting up S/MIME for your mail clients. In the properties of a mailbox store is a check box that says "Clients support S/MIME signatures." This check box must be selected if you want to enable your clients to use S/MIME (the check box is enabled by default). Figure 7-11 illustrates this configuration option.

Figure 7-11
S/MIME configuration in Exchange 2003



Microsoft Mail Client S/MIME Support

Both the full-blown Outlook mail client and Outlook Express have offered S/MIME support for quite some time. Brand new to Exchange 2003 is the S/MIME support for Outlook Web Access (OWA). Outlook 2003 is the latest version of Microsoft's complete mail client. Outlook Express 6.0 is a lightweight Internet-oriented mail client that Microsoft distributes together with IE 6.0. You can connect Outlook Express 6.0 to Exchange 2003, Exchange 2000, or Exchange Server 5.5 through SMTP and POP3 or IMAP4, and you can connect it to a directory through Lightweight Directory Access Protocol (LDAP). OWA allows for HTTP-based mail access from a browser—the OWA Web pages and logic are included with the Exchange 2003 code. Table 7-5 gives an overview of the Microsoft mail clients' main S/MIME features.

Table 7-5 Outlook client S/MIME features

	Outlook 2003	Outlook Express 6.0	Outlook Web Access
Mail protocol	MAPI/RPC, POP3, IMAP4, SMTP, HTTP	POP3, IMAP4, SMTP	HTTP
Encryption key recovery	Yes (if combined with a Windows Server 2003 CA or Windows 2000 Server with Exchange KMS)	Yes (if combined with a Windows Server 2003 CA or Windows 2000 Server with Exchange KMS)	Yes (if combined with a Windows Server 2003 CA or Windows 2000 Server with Exchange KMS)
CRL Distribution Point (CDP) support	Yes	Yes	Yes
Private key storage	Data Protection API, optional Syskey protection	Data Protection API, optional Syskey protection	Data Protection API, optional Syskey protection
Support for clear and opaque signing	Yes	Yes	No, only clear signing
Support for secure receipts	Yes	No	No
Support for security labels	Yes	No	No
Certificate Provider	KMS, internal CA, commercial CA	Internal CA, commercial CA	Internal CA, commercial CA
Certificate renewal	Can be automated (if using Windows Server 2003 and XP user autoenrollment)	Can be automated (if using Windows Server 2003 and XP user autoenrollment)	Can be automated (if using Windows Server 2003 and XP user autoenrollment)
LDAP support	Yes	Yes	No

Of the three mail clients, Outlook 2003 is certainly the most complete when it comes to S/MIME functionality. Outlook 2003 comes with support for some of the S/MIME-enhanced security services (explained later) and includes easy-to-use digital ID-management features. For example, from Outlook 2003, you can publish your digital ID (certificate) to the Exchange Global Address List (the Global Catalog) and request a new digital ID with an online CA.

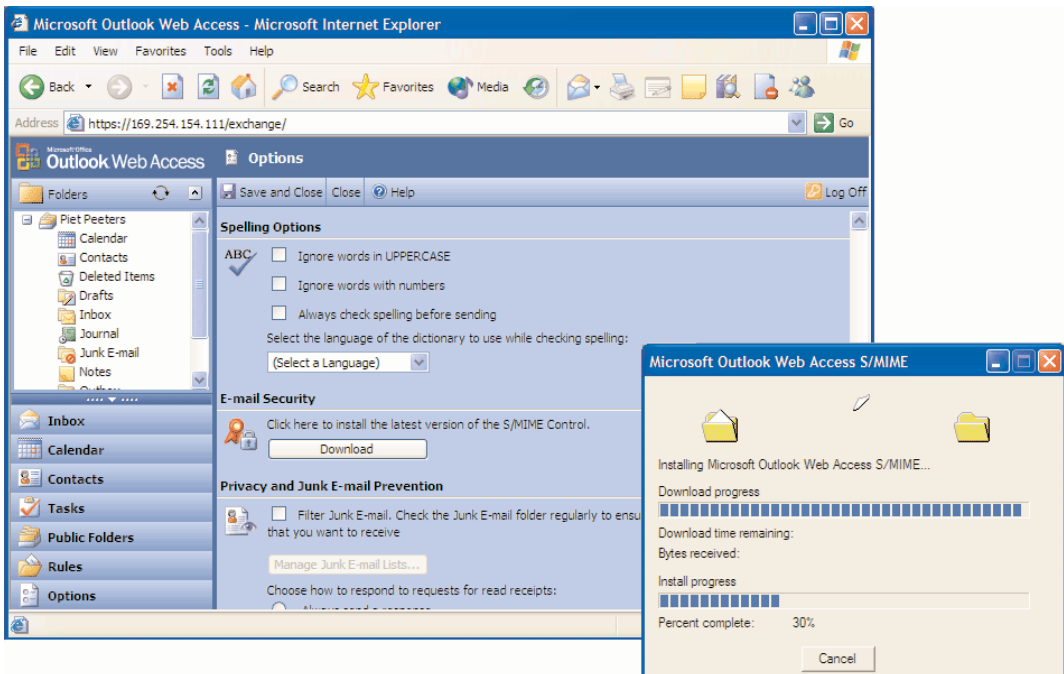
When you click the Get a Digital ID... button in the Outlook 2003 Security Options dialog box, Outlook redirects you to the Office Marketplace Web site. From there, you can, for example, request a digital ID with the United States Postal Service (USPS). To publish your personal certificates to the Global Address List (GAL), go to Tools, Options, Security Options, and select Publish to GAL... To disable this feature, change the Registry setting HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\11.0\Outlook\Security\Pub-lishToGalDisabled (REG_DWORD) to 1. Table 7-6 gives an overview of other Outlook 2003 S/MIME settings that can be controlled through registry settings. All of the settings are located in the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\11.0\Outlook\Security registry folder.

Table 7-6 Outlook 2003 S/MIME-related registry settings

Setting	Type – Value	Meaning
AlwaysEncrypt	REG_DWORD – 0,1	When set to 1, all outgoing messages are encrypted; corresponds to “Encrypt message content and attachments” GUI check box.
AlwaysSign	REG_DWORD – 0,1	When set to 1, all outgoing messages are signed; corresponds to “Add digital signature to this message” GUI check box.
ClearSign	REG_DWORD – 0,1	When set to 1, clear signing is used for all outgoing messages; corresponds to “Send this message as cleartext signed” GUI check box.
RequestSecureReceipt	REG_DWORD – 0,1	When set to 1, secure receipts are requested for all outgoing messages; corresponds to “Request S/MIME receipt for this message” GUI check box.

Exchange 2003 adds S/MIME capabilities to OWA. To use S/MIME in OWA, you must run IE 6 with SP1 or later. You enable OWA support in the configuration options of your OWA client. When you enable this feature, an ActiveX control providing the S/MIME logic will be downloaded from the OWA server to your client. You must be a power user or local administrator to install the control. If you are using OWA from different machines, you will have to enable it (and download the ActiveX control) on every machine (as Figure 7-12 illustrates).

Figure 7-12
Setting up OWA S/MIME support



When one user has downloaded the control to, for example, a kiosk machine, it will be available to all users of that machine. OWA users can obtain S/MIME certificates in exactly the same way as when they are using Outlook 2003 or Outlook Express 6.0: All of the certificate-enrollment methods we described in Chapter 4 also apply to OWA S/MIME. Once users have enabled their OWA client to use S/MIME, the OWA interface and message properties will be extended to add digital signatures and encrypt the message content. A very nice feature of the OWA S/MIME support is that all certificate-related processing (such as revocation checking) is done on the Exchange 2003 server side. By default, all OWA traffic occurs over a secured HTTP-over-SSL communication channel (https).

Enhanced Security Services

RFC 2634 specifies four optional security-service extensions, also known as Enhanced Security Services (ESS), for S/MIME. These services include secure receipts and security labels. Microsoft supports some of these extensions in Outlook 2003 and Outlook Express 6.0. The extensions were first introduced in SR1a for Outlook 2000.

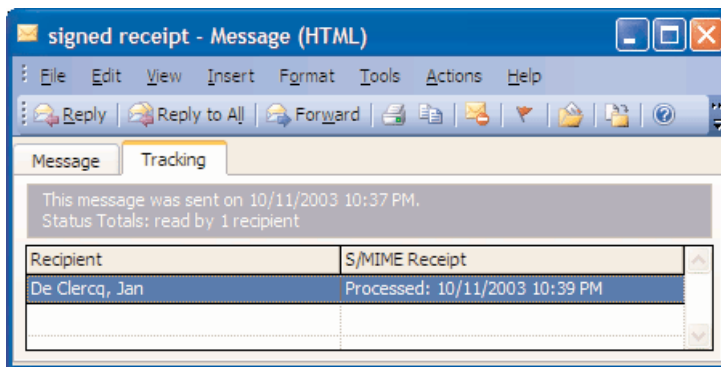
Secure receipts (which you should not confuse with Outlook's delivery receipts or read receipts) provide "non-repudiation of reading," which gives you cryptographic proof that the intended recipient has read and verified a signed message. A secure receipt is signed, meaning that when you receive a message and reply to it using a signed or secure receipt, you sign the receipt using a private key. You cannot deny having done so because only you can access and use the key. A secure receipt takes three steps in its travels. First, you generate and send a message, specifying a secure receipt. Next, the recipient responds to the secure receipt request. Finally, you receive the secure receipt.

You can set the secure receipt request for "signed only" or "signed and encrypted" messages. The message can be clear or opaque signed. You cannot make secure receipts receiver-dependent—if you set "Request S/MIME receipt for this message," the setting applies to all recipients. Also, secure receipts always return to the message's sender; it is not possible to set another return mailbox. You can make secure receipts the default by selecting the "Request S/MIME receipt for all S/MIME signed messages" check box in Outlook's security options.

By default, Outlook 2003 automatically sends a secure receipt when you open the message and when the system can cryptographically verify the message signature. If you add the Registry value HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\11.0\Outlook\Security\RespondToReceipt-Requests (REG_DWORD), and set the value to 1, Outlook prompts you before it sends a secure receipt: "A request has been made to send an S/MIME receipt when the message has been verified. Do you want to send an S/ MIME receipt?"

To verify a secure receipt—to cryptographically verify the receipt's signature and instruct Outlook to check the receipt's content against the original—you must open the receipt. If the original message is not in the Sent Items folder, Outlook prompts you to find it. If you cannot find the original message, the secure-receipt verification fails. If verification succeeds, Outlook 2003 automatically adds tracking information to the original message, as Figure 7-13 shows. Tracking information enables easy and centralized receipt status checking.

Figure 7-13
S/MIME signed receipt-tracking information



A security label, a kind of tagging system for Email messages, defines a message content's sensitivity. As with secure receipts, you can set security labels on signed messages. The power of the security-label feature lies in its ability to restrict a user's access to a mail message, which is a standard requirement for messaging systems in military environments. The military, which does not want a sergeant—even if that sergeant is a system administrator—to have access to a general's mail, conceived of many of the concepts that RFC 2634 defines and SR1a implements.

In Outlook 2003, Microsoft provides a UI that lets you attach security labels to the messages you send. In addition to the UI, you need two things to implement security labels: first, security policy modules, which define the classification levels, and second, client-side logic, which enforces the security labels based on a user's classification level. Because security policies differ for each organization, Microsoft does not provide the logic out of the box for security labels. Instead, Microsoft provides a sample policy and a policy-module design document in the MS Office software development kit (SDK).

Leveraging Smart Cards and USB Tokens for PKI-Enabled Applications

Windows 2000, Windows XP, and Windows Server 2003 can support both smart cards and USB tokens. Although these two device types have a different form factor and typically use a different computer-connection interface, they both offer the same service: secure, hardware-based PKI credential (private keys and certificates) storage.

Unfortunately, not every Windows PKI-enabled application supports smart card or USB token credential storage. As of this writing, the following Windows PKI-enabled applications are natively capable of dealing with smart card-enabled or USB token-based credentials:

- Secure mail using S/MIME from the Outlook mail client
- Smart card logon to a Windows Server 2003 domain (interactive logon)
- Smart card logon to Windows Server 2003 from the Windows Server 2003 Terminal Services client

- Smart card logon to a Windows Server 2003 domain using remote access
- Client authentication from IE in SSL/TLS-based secure Web scenarios

Windows Server 2003 includes important enhancements to the way Windows administrators can use smart cards and USB tokens. Administrators can now access their credentials stored on a smart card or a USB token when they are running the `dcpromo.exe` command (to install AD), the `net.exe` commands, and the `runas.exe` command (to switch between different security identities), and when they are using terminal services.

Next, we focus on the built-in Windows 2003 and Windows XP smart card and USB token support, Windows Server 2003 smart card logon, and third-party software that you can use to extend the Windows smart card or USB token-management capabilities on the Windows Server 2003 platform. Unless mentioned otherwise, in the following sections, “smart card” refers to both smart cards and USB tokens.

For a broader introduction to smart cards and USB tokens, refer to the following books:

- *Smart Card Security and Applications* by Mike Hendry, Artech House, 2001.
- *Authentication: From Passwords to Public Keys* by Richard E. Smith, Addison-Wesley, 2002.
- *Smart Cards: The Developer's Toolkit* by Tim Jurgensen and Scott Guthery, Prentice Hall, 2002.

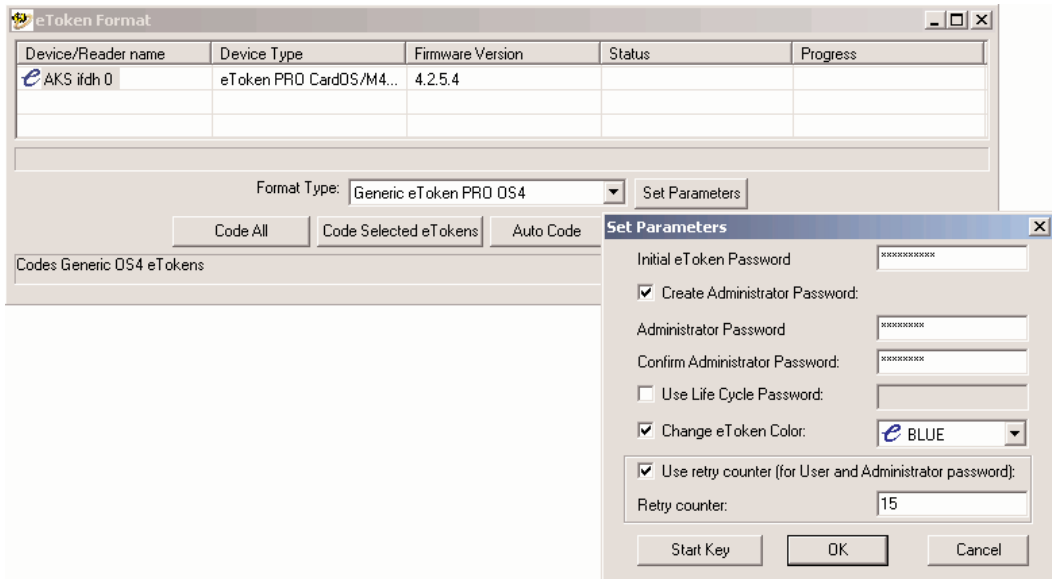
Windows Server 2003 and Windows XP Smart Card Support

For a complete list of all the smart card readers and USB tokens that are compatible with Windows 2003 and Windows XP, refer to the Windows Hardware Compatibility List (HCL), available from <http://www.microsoft.com/windows/catalog/> for Windows XP and from <http://www.microsoft.com/windows/catalog/server/> for Windows Server 2003. All Windows Server 2003- and Windows XP-compatible smart card readers support the PC/SC smart card interface. More information on the PC/SC smart card interface standard is available from <http://www.pcscworkgroup.com/>.

Out-of-the-box Windows Server 2003 supports smart cards from Gemplus, Infineon, and Schlumberger. The support for smart cards depends on the availability of a driver and a smart card CSP. Both software components enable the operating system to communicate with the card for credential storage, and for cryptographic and configuration operations. Smart cards and USB tokens from other vendors (ActivCard, Datakey, eAlladin, and Spyrus) are supported if you add the appropriate drivers and CSPs.

Most smart cards and tokens come with special management software that lets you configure different card or token properties, such as the PIN code. When you associate a PIN code with a smart card, you bind the card to the entities that know the PIN code—a process that is known as *smart card personalization*. Most management software also lets you set the number of bad PIN entry attempts, after which the card is locked. Figure 7-14 shows the token-management utility eToken Format, which, together with its eToken USB tokens, is made available by eAlladin.

Figure 7-14
eAladin eToken format utility



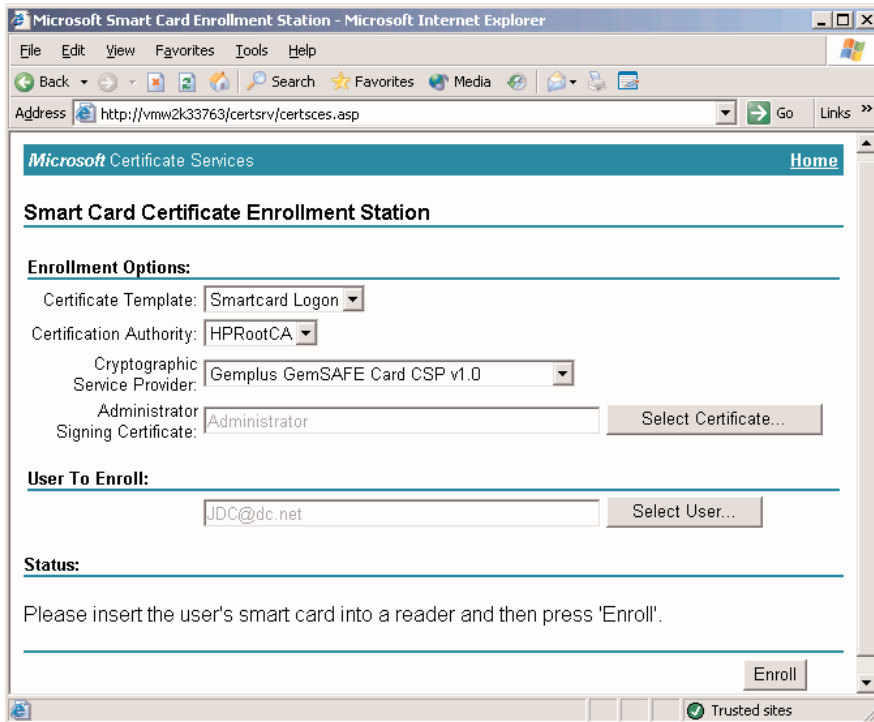
Enrolling for Smart Card-Based Credentials

In a Windows Server 2003 environment, a user can enroll for smart card credentials (certificates and private keys), provided he or she has the appropriate permissions on the smart card certificate templates, and provided the templates are available on the CA. To enroll for smart card credentials, a user can use any of the enrollment methods mentioned in Chapter 4.

Windows Server 2003 comes with two default smart card-related certificate templates: smart card user and smart card logon. In addition to SSL/TLS client authentication and smart card logon, the smart card user template offers support for secure Email. Smart card certificates are visible in a user's certificate store. When a smart card is plugged in, CryptoAPI will automatically copy the smart card certificates to the user's certificate store. CryptoAPI will propagate only the first certificate on the smart card to the user's My certificate store container. It's up to the CSPs to propagate other certificates stored on the smart card.

An alternative to this method is to let the user certificates and private keys be loaded centrally by an administrator who has a special enrollment-agent certificate. This approach is what Microsoft calls "enrollment using a smart card enrollment station." Given the importance of smart card enrollment from a security point of view, and given the difficulty of the user certificate-enrollment procedure in general, the use of a smart card enrollment station is the preferred way to enroll users for smart card-based credentials in a corporate environment. The smart card enrollment-station interface (illustrated in Figure 7-15) is Web based.

Figure 7-15
Smart Card Certificate Enrollment Station interface



You can access this interface from the default certificate-server Web interface (provided that you are using an enterprise CA) by selecting the “Request a certificate for a smart card on behalf of another user” and by using the “smart card certificate enrollment” option (which is available from the advanced certificate request menu option). Only administrators who have an enrollment-agent certificate in their personal certificate store can request smart card certificates on behalf of another user. Smart card certificate requests are signed using the enrollment agent’s private key. The CA validates the requests using the enrollment agent’s corresponding public key. If you check the access-control settings on the enrollment-agent certificate template, you will notice that, by default, only domain and enterprise administrators can enroll for an enrollment-agent certificate.

By now, it should be clear that you should handle enrollment-agent certificates and smart card enrollment stations with extreme caution. An administrator with an enrollment-agent certificate can impersonate anyone in the corporate network. He or she is the one generating smart card credentials, so he or she can also use those credentials on behalf of a user to log on. If someone succeeds in logging on to an enrollment station using an administrator’s credentials, he or she can do even more harm. He or she can request smart card certificates on behalf of anyone in the organization and impersonate anyone on machines where a smart card reader is available. That is why it is advisable to do the following:

- Install a smart card enrollment station on a dedicated, highly secured machine;
- Limit the number of enrollment agent administrators;
- Use a special CA to issue smart card certificates;
- Implement strict access-control settings on the enrollment-agent certificate templates; and
- Write special security policies that regulate the use of enrollment-agent certificates and smart card enrollment stations.

Smart Card Logon

Smart card logon in Windows 2000 and Windows Server 2003 is based on an extension to the Kerberos protocol, called PKINIT, which stands for use of Public Key Cryptography for Initial authentication in Kerberos. In PKINIT, all occurrences of the hashed password in the initial Kerberos authentication sequences are replaced by a user's public and private keys.

The use of smart cards for identification offers the following advantages:

- Smart cards offer a user-identification alternative that is much stronger than plain password identification. Smart card logon is based on two-factor authentication: It combines knowledge (of an alphanumeric PIN code) and possession (of the smart card).
- Smart card logon is more difficult to break. The smart card logon sequence relies on asymmetric keying material instead of simple passwords.
- Smart cards offer secure and tamper-resistant credential storage. The user's credentials (private keys and certificates) never leave the card. Theoretically, all critical calculations that involve the private key also occur only on the card itself.
- Smart cards can provide roaming of credentials. A user can log on and have access to his or her credentials from every system that has a smart card reader installed.

Installing a smart card reader on a Windows 2000, Windows XP, or Windows Server 2003 machine will change the GINA (the Graphical Identification and Authentication component, or the screen that pops up when you press <CTRL>-<ALT>-), as illustrated in Figure 7-16 for Windows Server 2003.

Figure 7-16
Smart Card Logon interface



The easiest way to set up smart card logon is to use a Windows enterprise CA. An enterprise CA automates most of the enrollment-related tasks, such as publishing the certificate to AD and linking it to the user's Windows account. Third-party CAs (e.g., Entrust or Baltimore CAs) also can issue smart card logon certificates. How to do this is explained in the Microsoft Knowledge Base article Q281245 available from <http://support.microsoft.com/default.aspx?scid=kb;en-us;281245>. For a smart card logon to work, both the domain controller, which validates the smart card authentication request, and the user, who logs on using a smart card, must have valid certificates. "Valid" means that both certificates must chain up to a trusted CA, and that none of the certificates in the domain controller's or the user's certificate chains should be revoked. To check for revocation, the CRLs must be available and valid. For more information about certificate validation, please refer to Chapter 5 of this eBook.

A nice feature of smart card logon in Windows 2000, Windows XP, and Windows Server 2003 is that OS behavior for smart card removal can be set. You can configure the smart card removal behavior in the machine Security Options of the Domain, Site, OU, and Local Computer GPO objects at the following location: Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options. The values that can be set are No Action, Lock Workstation, and Force Logoff.

Windows 2000 and Windows Server 2003 also allow an administrator to force the use of a smart card for interactive logon. By default, a user who has a smart card can still log on using his or her password. To do so, select the "Smart card is required for interactive logon" check box in the user object account properties dialog box, or set the "interactive logon: Require smart card" security option in the computer portion of the GPO security settings.

Smart Card Management Systems

For many large enterprises, the built-in Windows 2000 and Windows Server 2003 smart card management features are not enough. These organizations are looking for advanced smart card and token-management capabilities, such as the following:

- Assigning smart cards or tokens to users. In a Windows environment, this means linking the smart card or token (and not just the PKI credentials) to an AD user account.
- Keeping track of the smart card or token content and usage.
- Defining which PKI-enabled applications can be used with a user's smart card or token.
- Handling lost smart cards or tokens.
- Formatting or reformatting smart cards or tokens.

Table 7-7 provides an overview of software vendors who sell smart card management-system software.

Table 7-7 Smart card management software

Vendor	Product	URL
ActivCard	ActivCard Identity Management System (AIMS)	http://www.activcard.com
Alacris	idNexus Smart Card Management Module	http://www.alacris.com
BellID	ANDiS	http://www.bellid.com
CardBase	Mascot	http://www.cardbase.com
Datakey	Card Management System (CMS)	http://www.datakey.com
eAlladin	Token Management System (TMS)	http://www.ealladin.com
Intercede	Edefice	http://www.intercede.co.uk

Conclusion

In this chapter, we have shown how applications can use the security services a PKI offers. Although we did not offer an exhaustive overview of all applications that can take advantage of a PKI and its services, the details included illustrate once more the critical security role a PKI can have in today's enterprises.

All the information provided in this and the previous chapters should give you sufficient Windows PKI background to build a secure and scalable PKI solution for your organization. The most important thing you should remember is that building a PKI is much more complex than running the Windows CA installation wizard. Building a PKI requires detailed planning and design that examines all PKI components and procedures, and involves all the stakeholders in your organization.

I hope this eBook was helpful! Good luck!