505.3
Windows PKI and Smart Cards

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

SEC505.3 | Securing Windows and PowerShell Automation

SANS

Deploying Windows PKI and Smart Cards

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up-to-date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

The SANS Institute does not provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document or associated files, tools or scripts. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

Microsoft, MS-DOS, MS, Windows, Active Directory, Internet Information Server, IIS, and Group Policy are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Apache is a product and trademark of the Apache Software Foundation. Mitochondria are cellular organelles that possess their own genes separate from the human genome and are believed to originally have been symbiotic bacteria. Or are they still? Google Chrome is a product and trademark of Google, Inc. FireFox is a product and trademark of the Mozilla Foundation.

RSA BSAFE Crypto-C, RSA BSAFE Crypto-J, Keon Desktop, MD2, MD4, MD5, RC2, RC4, RC5, RC6, RSA, and SecureID are trademarks or registered trademarks of RSA Security Inc.

Other products and names are trademarks or registered trademarks of their respective owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance upon the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission which would clearly add value to the document, please e-mail the contact listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

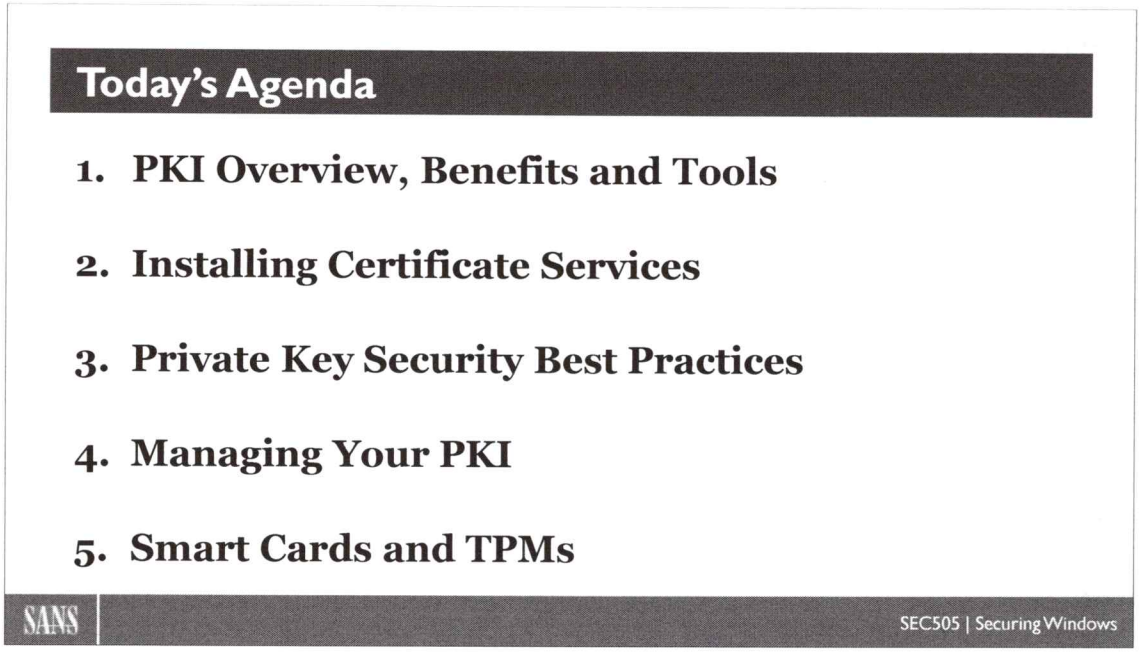
Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
Document Version: 33.1 (B02_01)
Last Modified: 31.Oct.2016

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Michael Howard (Microsoft): helping me to find "the PKI guys" at Microsoft.
Will Vaughn (www.annuitynet.com): additional recommendation for securing private keys.
Joe Bonaiuto (ONI): the cryptosong.
Rakesh Bharania (Cisco Systems): secret, undocumentable assistance...
Neil Todd (JP Morgan chase): useful additions.
Rob Younce (MBNA): dssvault.com, dimedealer.com, etc. for hacking DTV smart cards.
Andrew Nielsen (Raytheon): more secret, undocumentable assistance-- thanks Andrew!
Roberto Quinones (Intel): URLs about the "NSA key".
Gordon Taylor (Royal Bank): Syskey.exe /L for hands-free enabling on NT.
Franklin Witter (Branch Banking & Trust): IE 5.5, HEP and SP2 issues.
Edward Sargent (Microsoft): URL corrections for PKI info.
Ron Bartnikas (BMO.COM): \Winnt\System32\Microsoft\Protect\
Crypto Tech Group at NAI Labs (www.nai.com): writing the DPAPI paper for MS!
Alex Lopyrev (Bank of Montreal): great URL on MS protected storage system.
Ken Hoover (Yale): Enterprise Admin membership needed to install Enterprise CA.
Steven Stark (DNS Computing): iisexport.exe
Dan Dennison (thedennisons.org): AES in XP
Richard Guida (Johnson & Johnson): advice from PKI roll-out at J&J.
Scott Cleven-Mulcahy (usa.net): CAPOLICY.INF variables info.
Rimvydas Zilinskas (Bank of Lithuania): fix to an obvious EFS error.
Alex Lopyrev (Bank of Montreal): commercial EFS recovery tools.
Mike Lonergan (Microsoft): great EFS and SC logon details!
Claes Nyberg (human): great crypto book recommendation.
David Bushta (human): corrections to broken URLs and FireFox factoids.
Bill Hampton (Accredo): CryptoAPI potential vulnerabilities.
Steve Moon (Davis Vision): updates to crypto hardware vendors list.
Nick Lewis (ACM): PKI steering committee URL updates.
David Rice (Tantric Security): recovery agent cert requires on-line CA.
Charles Bombard (CCV): BitLocker attack link: http://citp.princeton.edu/memory/
Greg Farnham (human): embarrassing grammatical and spilling errors.
Christian Gigandet (human): good BitLocker tip for non-TPM machines.
Michael Bruno (human): Vista-EFS to XP-EFS issues.
Kyle Voss (human): caught really stupid errors in slide and book.
Ken Gallo (human): only Vista and later currently support SHA-256.

Anita Metcalf (human): support for wildcard certificates.
Cal Frye (Human): updated screenshot for Server 2008-R2 Enterprise Edition.
Arden Meyer (Human): correction to allowed protocols.
Terry Lenn (Indiana Univ): good tip about SMBv2 and EFS encryption.
David Busby (Schlumberger): numerous typos and corrections.
Aaron Beuhring (Williams & Connolly): GPO control of PnP devices.
Smita Carneiro (Ross Enterprise): fix to a GPO path for roaming profiles.
Rick Moffat (Human): fix to the root CA audit script.
Ginny Munroe (DeadlineDriven.com): lots of grammarickies -- like this line!

Table of Contents



Today's Agenda

This course will present the essentials of Windows Public Key Infrastructure (PKI) and Smart Card authentication. The focus is on the Critical Security Control for Data Protection, but PKI indirectly supports other Controls as well; for example, the certificates from the PKI can be used to for smart card authentication of administrators and for securing access to wireless access points and VPN gateways.

Why PKI?

All the world is connecting to the Internet and PKI technologies have become the chosen method of making it secure. Many corporations, government agencies and university campuses are now beginning to deploy wide-scale PKIs, just as they deployed PBX phone systems in the 1970s, e-mail systems in the 80s, and websites in 90s. Smart cards are even built into DirecTV receivers, cable modem boxes and Sony PlayStations can even use SSL encryption of transmitted data.

Smart cards are already widely used in Europe and East Asia. In America, many Fortune 500 corporations and federal agencies are making their Employee ID Cards "smart" by simply printing them on cards with microprocessors and memory. In Windows 8 and later, a TPM chip in the motherboard can be used as a virtual smart card.

Digital signature legislation is making high-dollar e-commerce possible by providing a framework for legally enforcing electronic contracts. PKI makes this possible and the profits generated will finance further PKI deployments: the cycle is self-reinforcing. In short, PKIs are the next wave in the development of the information economy.

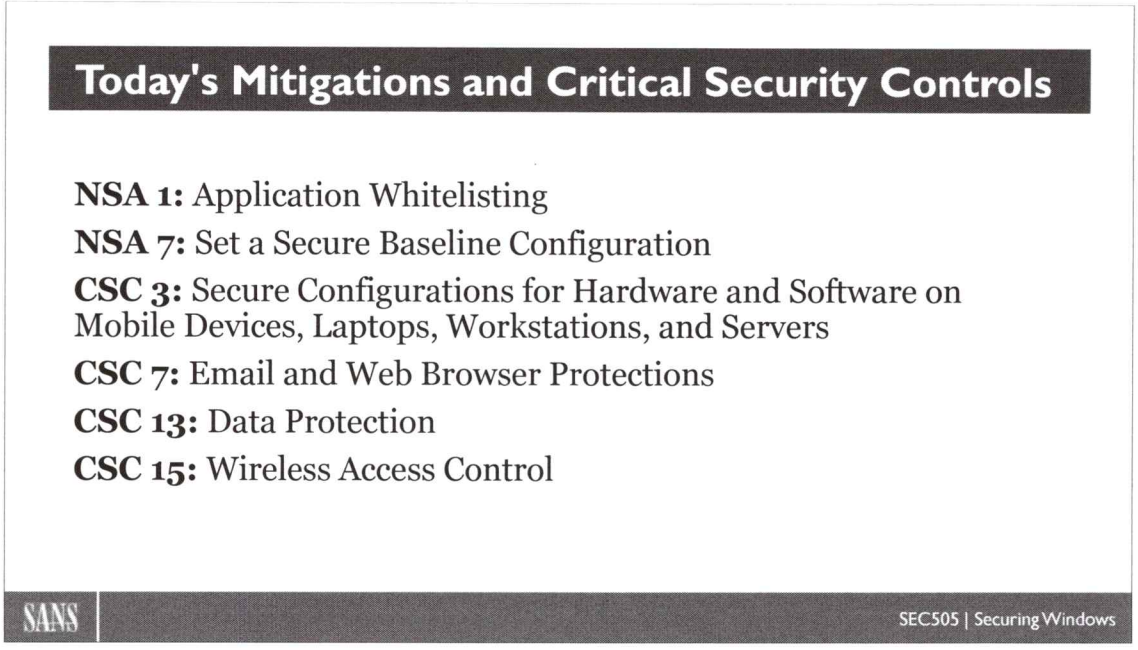
As a network engineer, it is not a matter of if you need to learn about cryptography and PKI, but *how much*? This course is intended to provide you with everything you need to know to *use* a PKI, but without drowning you in the details.

The major sections of the course are:

- Cryptography and PKI Concepts
- Overview of Windows PKI
- Installing Windows Certificate Services
- Private Key Storage and Protection
- Managing Certificates
- Smart Card Authentication

By the end of this course, you will be able to:

- Deploy a Windows Public Key Infrastructure.
- Issue, renew and revoke digital certificates.
- Secure the private keys of clients and Certification Authorities.
- Issue smart card certificates.



Today's Mitigations and Critical Security Controls

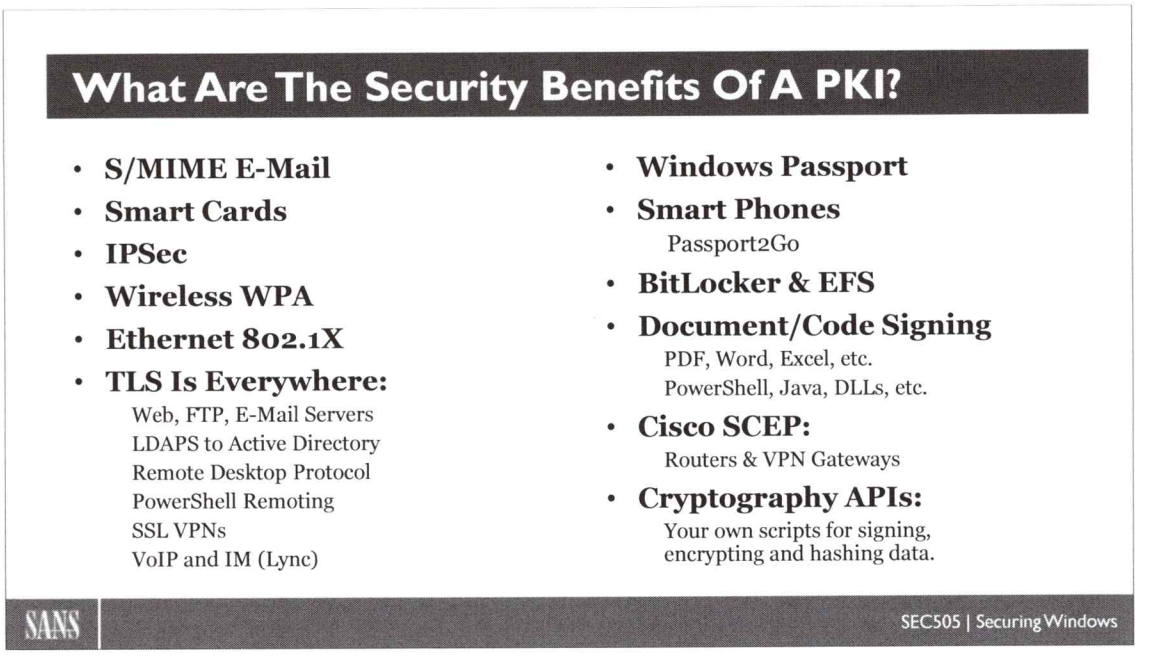
One mission of the National Security Agency (NSA) is to offer network security guidance through its Information Assurance Directorate (IAD). The NSA/IAD list of Top 10 Information Assurance Mitigation Strategies can be downloaded from the IAD web site (www.iad.gov).

The Critical Security Controls (CSC) project aims to describe the 20 most important tasks and activities for network security. You can download the latest version of the CSC from the web site of the Center for Internet Security (www.cisecurity.org).

Today's material is especially relevant for implementing the following NSA Top 10 Mitigations and CIS Critical Security Controls:

- NSA 1: Application Whitelisting
- NSA 7: Set a Secure Baseline Configuration
- CSC 3: Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
- CSC 7: E-mail and Web Browser Protections
- CSC 13: Data Protection
- CSC 15: Wireless Access Control

But PKI indirectly applies to many of the NSA Top 10 Mitigations and CIS Critical Security Controls, not just those listed above. PKI is an enabler for many other security technologies that use certificates for strong authentication and encryption. For the sake of this course, PKI is also covered because it can be difficult to implement correctly.



What Are The Security Benefits Of A PKI?

A "Public Key Infrastructure" (PKI) is a collection of databases, services, applications, protocols and standards surrounding the use of public key certificates for secure communications and data storage. (Digital certificates will be discussed soon.)

An example of an "infrastructure" is our national system of roads and highways. Our highway system and a PKI have the following four characteristics in virtue of being infrastructures:

1. Can be used for a *variety of purposes*,
2. Is *standardized* in its construction, rules of use and interfaces,
3. Is relatively *easy to use and manage*, and
4. Is *cost-effective* in that it costs less to build and maintain the infrastructure than to live without it.

1) ***Variety of Purposes***. A PKI is a general-purpose security application enabler, hence, it can be employed for an unlimited number of purposes with almost any type of communications or data storage capability.

2) ***Standardization***. PKI standards are published in Internet Engineering Task Force (IETF) recommendations, National Institute of Standards (NIST) guidelines, Internet Request for Comments (RFC), and other documents. The protocols and standards involved are vigorously debated in IETF working groups, and numerous vendors are competing to make their products *de facto* standards, e.g., PKCS from RSA Laboratories. The purpose of this standardization is to provide ***interoperability*** between different

vendors' products. Windows-supported PKI standards include: X.509v3, PKIX, CRLv2, S/MIME, TLS, SSLv3, SGC, IPSec, PKINIT, PKCS, and PC/SC.

3) ***Easy To Use***. A PKI also makes the use of cryptographic services easy to use for developers and end users. The key to ease-of-use is ***transparency***. A well-implemented PKI should be almost invisible to end users and require no special training; for example, users should be able to encrypt and sign e-mail by simply clicking a single button in their e-mail applications. PKI services should also be (relatively) transparent to application developers through the availability of cryptographic Application Programming Interfaces (APIs). On Windows, CryptoAPI/CryptoNG and the Security Support Provider Interface (SSPI) enable developers to easily write applications which plug into the PKI services of Windows.

4) ***Cost-Effective***. A PKI is a cost-effective investment for medium to large organizations that require high security, do business over the Internet, digitally sign electronic contracts, use the Internet to securely transmit sensitive information, etc. Each of these services can be implemented *ad hoc* or piecemeal, but this is too expensive when a large number of users or servers require these security services. A PKI permits the use of secure communications and data storage to ***scale*** without a cost explosion. If only a small number of users or servers need to use digital certificates, then it would better to outsource one's PKI or use alternative security mechanisms. For Windows, if an organization has already invested in an Active Directory infrastructure, then adding PKI services is relatively inexpensive.

Specific Benefits

Yes, but what can a Windows PKI do for me *today*?

The following are examples of the uses and benefits of a Windows PKI:

- **Secure E-Mail.** Users can encrypt and digitally sign their e-mail messages to maintain confidentiality, integrity and non-repudiation of origin. Many e-mail programs support S/MIME to simplify the process of using secure e-mail with digital certificates, such as Microsoft Outlook and Netscape Communicator.

- **Smart Card support.** Smart Card logon uses the keys and microprocessor on a credit card-sized card to authenticate to one's desktop system and remote servers. This is integrated with the Kerberos protocol for a very high level of authentication security (following the PKINIT extension to Kerberos 5). User accounts can be configured to require smart card logon, and computers can automatically "lock" when the user's smart card is removed. The smart card itself is tamper-resistant and will render itself unusable if a thief attempts to guess its PIN number. All cryptographically sensitive operations occur on the smart card itself so that user's private keys are not exposed; hence, the smart card is also used for decryption and digital signing.

- **Windows Passport.** Windows 10 and later supports Fast IDentity Online (FIDO) authentication to web applications using public/private key pairs instead of passwords. The FIDO Alliance includes Microsoft, Google, PayPal, Visa, MasterCard and other organizations. Windows Passport is Microsoft's implementation of FIDO. The public/private key pairs used with Passport can be generated locally on each device without interaction with a PKI, but it's best for security if the public key is signed by a CA and returned to the device in the form of a certificate. Either with or without a PKI, if the device has a TPM chip, the TPM will help to secure the private key and other authentication tokens related to Passport. Passport works with either on-premises Active Directory, Azure Active Directory, or a hybrid of the two.

- **IPSec.** Internet Protocol Security (IPSec) is an industry-standard set of protocols for authenticating, encrypting and encapsulating TCP/IP packets. IPSec does not necessarily require digital certificates in order to work, but the most secure form of IPSec authentication is with certificates.

- **Certificate-Based Authentication to IIS.** With or without a smart card, a user can be authenticated to an IIS webserver with a personal certificate installed in their browser. At the same time, the IIS server is authenticated to the user with a certificate installed on the IIS server. When users authenticate this way, their personal certificates are mapped to regular local/domain user accounts, and they are transparently logged onto the IIS server with these user accounts. This authentication plus TLS/SSL encryption provides a very secure basis for web applications on the Internet, such as on-line banking and stock trading.

- **Certificate-Based VPN Authentication.** The Unified Access Gateway (UAG), Threat Management Gateway (TMG), DirectAccess, and the Routing and Remote Access Service (RRAS) permits users to log on over Virtual Private Networking (VPN) connections. Both the connection and the subsequent logon to the domain are authenticated using Extensible Authentication Protocol (EAP) and Transport Layer Security (TLS) with the user's personal digital certificate.

- **Wireless 802.1X EAP-TLS and PEAP Authentication.** 802.11 wireless clients and Access Points (APs) can mutually authenticate to each other and securely exchange encryption keys using 802.1X Extensible Authentication Protocol (EAP) methods. Windows systems support both EAP-TLS and Protected EAP (PEAP) for use with 802.1X. EAP-TLS requires a digital certificate on both the client and server sides, while PEAP requires a certificate on just the authentication server, i.e., the RADIUS server.

- **Encrypting File System.** The NTFS file system on Windows supports native, transparent encryption of data, just as it supports native, transparent compression. Each file is encrypted with its own random key, then this key is encrypted with

the user's public key and stored with the file. In Windows 10 and later, EFS is used for Enterprise Data Protection to secure company files for DLP.

- **Microsoft Skype for Business.** Skype provides instant messaging, presence information, VoIP, and video-teleconferencing within the corporate LAN, over the Internet, between federated organizations, and to popular instant messaging providers like Yahoo and MSN. Certificates on the Skype for Business servers (formerly known as Lync) are used for server authentication and data encryption.

- **TLS/SSL Encryption to IIS (HTTP and FTP).** Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are general-purpose authentication, integrity-checking and encryption protocols that use digital certificates. TLS and SSL are used by a variety of services and applications, especially for secure communications between browsers and webservers using HTTPS and FTPS.

- **TLS/SSL Encryption to Domain Controllers with LDAP.** Domain controllers use LDAP to provide access to Active Directory and the Global Catalog. The data is sent in cleartext, however, unless TLS/SSL is used to secure the LDAP channel. TLS/SSL requires a digital certificate on the controller for LDAPS on TCP/636 and TCP/3269 though. A certificate is also required on the controller in order to support smart card authentication.

- **SMTP Active Directory Replication.** To replicate Active Directory data using the SMTP connector, a Windows CA is required because the data is encrypted and digitally signed by the domain controllers using S/MIME.

- **TLS/SSL Encryption to SQL Server.** Microsoft SQL Server can SSL-encrypt all transactions with it, but only if the server has a computer certificate installed. Encryption is important to protect client application traffic, replication traffic, and the DBA's management of the server.

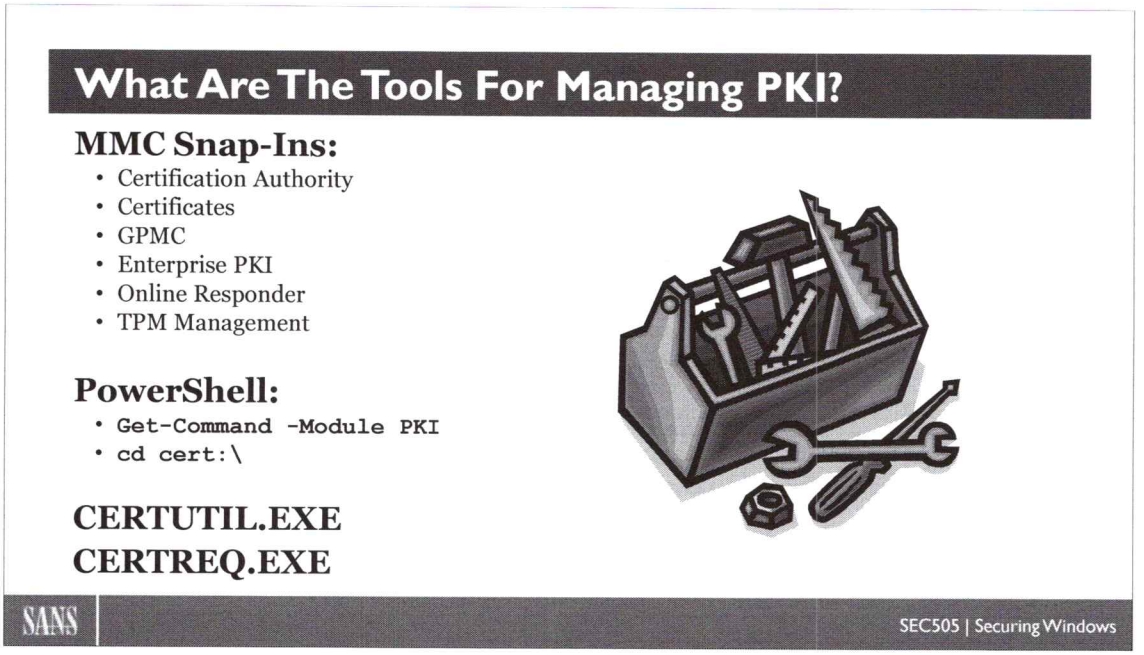
- **Smart Phones.** Some smart phones can use certificates for S/MIME e-mail, WPA wireless, IPSec VPN connections, as well as HTTPS and SMTPS. "Passport To Go" (Passport2Go) is the use of Windows Passport on a portable device, such as a smart phone, for multi-factor authentication to a second device, such as a PC, or to a web application.

- **Code Signatures.** Microsoft's Windows Script Host (WSH) and PowerShell can be configured to only execute scripts if they have been digitally signed with a certificate from a trusted issuer. That issuer will be you. (The PowerShell seminar will show how to set this policy and how to digitally sign scripts.) Similarly, macros in Word, Excel, Outlook, Access, etc. can be digitally signed, then these Office applications can be configured through Group Policy to only run macros that have been signed by you.

- **Document Signatures.** PowerPoint, Word, PDF and other document formats support digital signatures to verify integrity and authenticate the source of the document. In Word 2010, for example, you can sign a document (File tab > Protect Document > Add Digital Signature) and a red ribbon icon in the bottom status bar will appear that, if double-clicked, will show the current (valid) signatures.

- **Cisco Router SCEP Support.** The Windows *Resource Kit* CD-ROM includes a utility (CEPSETUP.EXE) which will install support for Cisco's Simple Certificate Enrollment Protocol. Cisco routers use SCEP to enroll for IPSec certificates. Windows 8.1 and Windows Phone 8.1 and later support SCEP too.

- **Custom Applications: CryptoAPI, CryptoNG and CAPICOM.** Through the use of Microsoft's CryptoNG interface developers can write their own applications which utilize the Windows PKI. CryptoNG is the low-level interface (C and C++), while CAPICOM is the easy-to-use Automation COM wrapper for CryptoAPI (VB, VBScript, JScript, Perl). CryptoAPI is the older Windows XP/2003 interface, but CryptoNG fully implements CryptoAPI for backwards compatibility.



What Are The Tools For Managing My PKI?

There are a number of MMC and command-line tools available for managing your PKI.

Certification Authority MMC snap-in

After you install Certificate Services the Administrative Tools folder will have a new snap-in named Certification Authority (certsrv.msc). This is the primary tool for managing your CA and is used to perform the following tasks:

- Start and stop the Certificate Services executable (certsrv.exe).
- Backup and restore the CA's key pair, certificate database and configuration.
- Renew CA certificates.
- Set CRL publication schedule.
- Manually force a fresh CRL to be published.
- Modify the Policy and Exit modules.
- Manually accept pending certificate requests.
- Control which certificate templates are available for enrollment.
- Set management permissions on Certificate Services in Active Directory.
- View issued, pending and revoked certificates.

To open the Certification Authority MMC snap-in so that it will show extra information about revocation lists (CRLs), open the console with the "/e" switch:

```
certsrv.msc /e
```

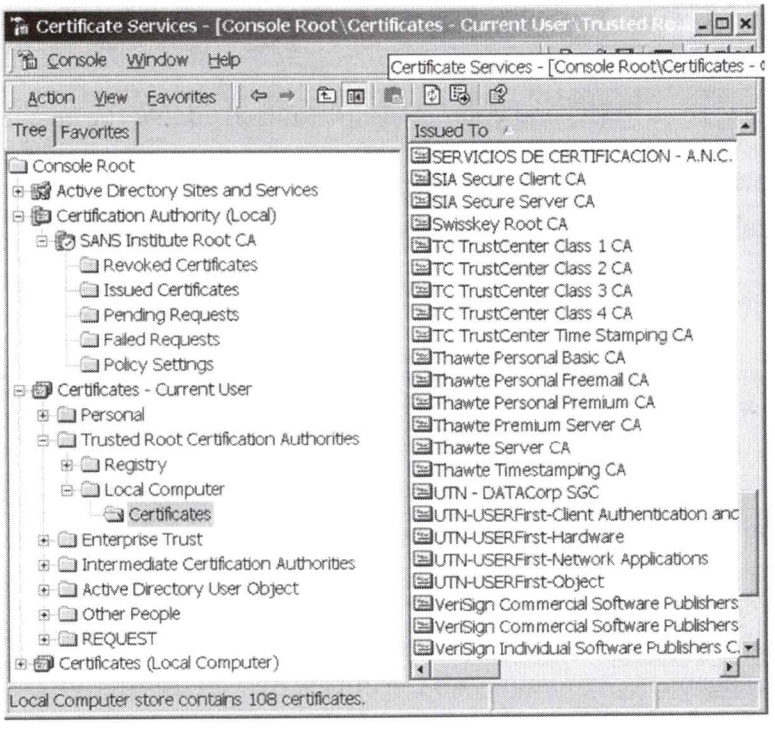
Certificates MMC snap-in

The Certificates snap-in is the primary tool for managing issued certificates, CTLs and Trusted Root Authorities on particular computers. When the snap-in is added to a MMC, the user will be prompted to select the Certificate Store for one of the following:

- My User Account -- the currently logged-on user only.
- Service Account -- any local or remote service which uses a certificate.
- Computer Account -- any local or remote Windows computer.

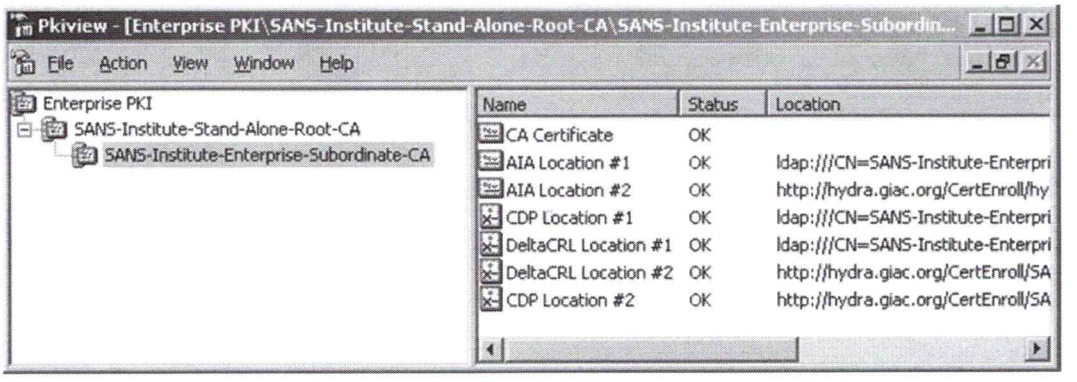
The Certificates snap-in can be used to perform the following tasks by either administrators or regular users:

- Request, export, import, delete, view and renew certificates.
- Export CRLs.
- Change the allowed purposes on certificates, including CA certificates.
- Set a "friendly name" and description on a certificate.
- Search various Certificate Store locations.



Enterprise PKI MMC snap-in

The Enterprise PKI tool (pkiview.msc) is a MMC snap-in that comes with the Windows Server 2003 Resource Kit Tools (a free download from Microsoft) and in built into Server 2008 and later. The tool shows one or more PKI hierarchies within your organization along with health status information. It mainly verifies that CA certificates and CRLs are accessible on the LAN.



PowerShell Certificates Provider

If PowerShell is installed, many certificate-related tasks can be performed through it because the local certificate store is exposed in PowerShell as the CERT:\ drive by the Certificates provider. This means the standard cmdlets can operate on these certificates just as they do with registry keys or NTFS files. For example, to list your personal certificates, run the following command at the PowerShell prompt:

```
dir cert:\currentuser\my\* | format-list
```

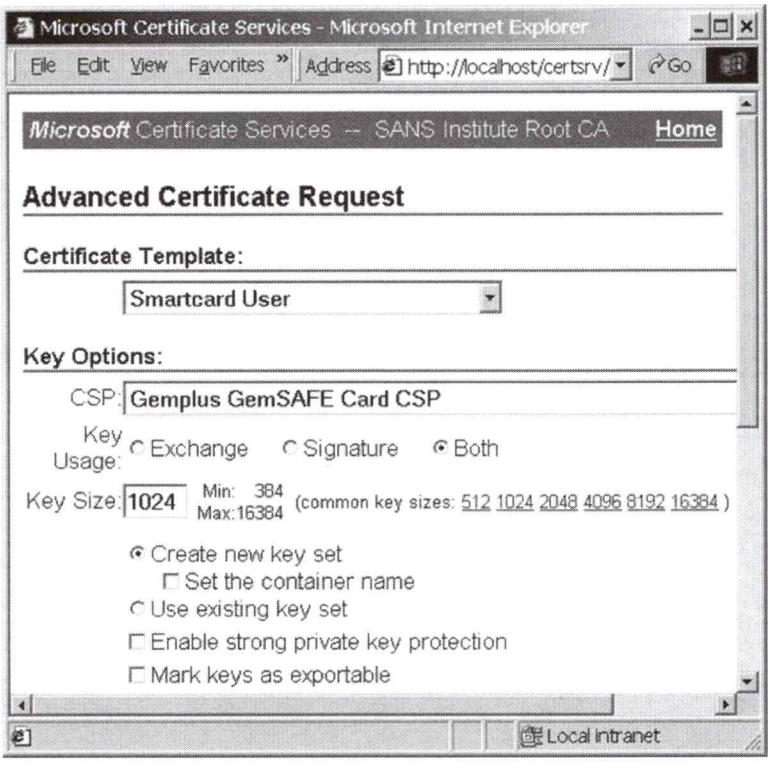
In PowerShell 3.0 and later, there are cmdlets specifically for PKI:

```
get-help *cert*
get-command -module pki
```

Certificate Services Website (http://localhost/certsrv/)

When Certificate Services is installed, there is an option to install the Certificate Services Web Enrollment Support web site. The URL for this site is http://*servername*/certsrv/. This website can be used by local or remote users to do the following:

- Download the CA's certificate.
- Download the latest Certificate Revocation List (CRL).
- Request a user certificate by filling in a form to specify all Advanced options.
- Create a Base64-encoded PKCS#10 certificate request.
- Submit a Base64-encoded PKCS#10 certificate request file.
- Submit a Base64-encoded PKCS#7 certificate renewal request file.
- Enroll another user for a smart card certificate on behalf of another user.
- Check on a pending certificate request.



IIS is built into the operating system of Windows Server. The web site can be installed on the CA itself or on another IIS server. However, if the website is installed on an IIS server other than the enterprise CA itself, that IIS server must be marked "Trusted for Delegation" in Active Directory. This means the IIS server can temporarily take on the identity of the certificate requestor (called "impersonation") in order to access the enterprise CA on behalf of the requestor.

> ***Try It Now!***
>
> To mark a computer as "Trusted for Delegation" in Active Directory, open the Active Directory Users and Computers snap-in > double-click the computer > Delegation tab > check the rabiobutton labeled "Trust this computer for delegation to any service" (or specific services only).

If the Certificate Services Website is installed on the CA itself, or if the CA is a stand-alone CA, the IIS server does not have to be marked "Trusted for Delegation".

Key Recovery Tool (Server 2003)

The Key Recovery Tool (KRT.EXE) is a graphical program included with the Windows Server 2003 Resource Kit Tools (a free download from Microsoft). The Key Recovery Tool is a graphical wrapper for the CERTUTIL.EXE program and is used to recover archived private keys from a Windows Enterprise CA.

Certificate Templates MMC snap-in

This is the primary tool for copying and editing the templates used to create certificates on Enterprise CAs.

Group Policy Settings for PKI

There are a number of Group Policy Object settings that relate to PKI management. Please see the Group Policy documentation that comes with the *Windows Resource Kit* for a full description of these settings, but the following is a list of most of them:

- Disable changing certificate settings in Internet Explorer.
- Disable the entire Contents tab in Internet Explorer.
- Prevent users from using the Certificates snap-in.
- Prevent users from using the Certification Authority snap-in.
- Smart Card removal behavior (lock workstation or force logoff when removed).
- Allow Public Key Policies snap-in extension on Certification Authority snap-in.
- Do not automatically encrypt files moved to encrypted folders.
- NTFS permissions and auditing of key materials.
- Secure channel: digitally sign secure channel data (when possible).
- Secure channel: digitally encrypt or sign secure channel data (always).
- Secure channel: digitally encrypt secure channel data (when possible).
- Secure channel: require strong session key (Windows 2000 or later).

DSSTORE.EXE

This is a command-line PKI troubleshooting utility that comes with the *Windows Resource Kit*. DSSTORE.EXE can do the following:

- Initiate download of Enterprise Root certificates from Active Directory (-pulse).
- Trigger auto-enrollment for computer certificates (-pulse).
- Display the Enterprise Root certificates stored on a computer (-entmon).
- Display the auto-enrollment objects and certificates on a computer (-entmon).
- Display machine objects, like Service Principal Names and DNS host names, on a computer (-entmon). See SETSPN.EXE in the *Resource Kit* for a related utility.
- Display information about a CA such as its CA name, DNS name and the certificate template types that it can use (-tcainfo).
- Display, add or delete the certificates, CRLs and AIAs of non-Windows or off-line Windows CAs in Active Directory, even though these CAs are not configured to use Active Directory. This is in lieu of using Group Policy.
- Check the presence and validity of domain controller computer certificates, which can affect how the controller operates as a Kerberos Key Distribution Center. This includes the ability to verify the entire certificate chain or "path of trust" to the computer certificate (-dcmon).
- Check the presence and validity of certificates on smart cards. This includes the ability to verify the entire certificate chain to the card's certificate (-checksc).

CERTUTIL.EXE

CERTUTIL.EXE is a command-line tool that can do almost all the tasks of the MMC snap-ins, as well as some things the snap-ins cannot do. This is a utility you should become familiar with if you administer a large or complex PKI. CERTUTIL.EXE can perform the following (run "certutil.exe -v -?" to see all switches):

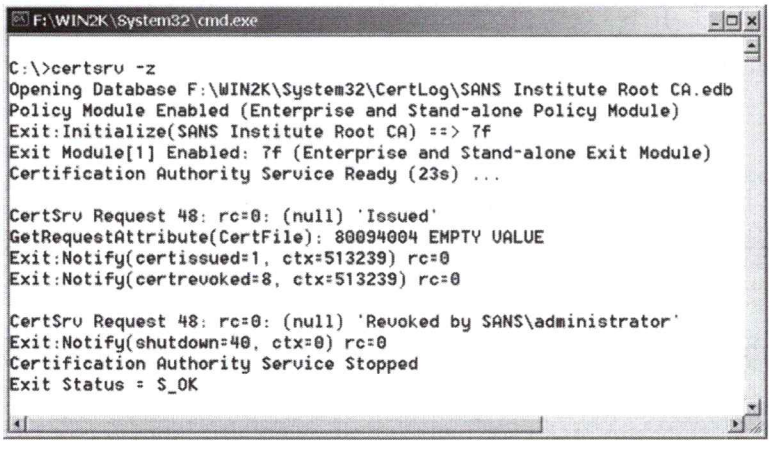
- Extract and recover an archived private key from a CA database (2003).
- Backup and restore the CA keys and database.
- Convert a Certificate Server 1.0 database to a Certificate Services 2.0 database.
- Create or remove Certificate Services Web virtual roots and file shares.
- Encode and decode Base64 files.
- Determine if a certificate is valid.
- Display certificates in a certificate store.
- Display error message text for a specified error code.
- Display the database schema.
- Get the certification authority (CA) configuration string.
- Import issued certificates that are missing from the database.
- Publish or retrieve a certificate revocation list (CRL).
- Resubmit or deny pending requests.
- Retrieve the CA signing certificate.
- Revoke certificates.
- Set and display certification authority registry settings.
- Set attributes or an integer or string value extension for a pending request.
- Shut down Certificate Services (certsrv.exe).
- Verify a public/private key set.
- Verify a certification chain or "trust path".

WINHTTPCERTCFG.EXE

WINHTTPCERTCFG.EXE comes with the Windows Server 2003 Resource Kit Tools (a free download). This command-line tool can import certificates and private keys from PFX files. It can also display or alter the permissions on a private key to regulate who can access it. Its long name reflects a likely use for it: to install an SSL certificate and private key on many IIS servers in a farm; but it can be used on any type of machine, not just IIS servers.

CERTSRV.EXE

CERTSRV.EXE is actually the Certificate Services executable itself. This is what is launched as a background process when Certificate Services starts automatically. After stopping CERTSRV.EXE as a background process, the program can be run within a command-prompt window with the -Z switch (certsrv.exe -z) and it will display status messages as it starts up, processes requests, encounters errors, etc. This is for troubleshooting purposes. It can be shut down by executing "certutil.exe -shutdown" from another command-prompt window.



CERTREQ.EXE

CERTREQ.EXE is a command-line utility for submitting PKCS#10 certificate requests and PKCS#7 renewal requests. Optionally, it can use straight RPC instead of DCOM for submitting these requests. (Run "certreq.exe -v -?" to see all switches.)

Quest Software's PowerShell CmdLets for Windows PKI and AD

Quest Software gives away for free an excellent set of PowerShell cmdlets for managing Active Directory and the Windows PKI. This makes scripted bulk management of CRLs and certificates much easier (http://www.quest.com/powershell/activeroles-server.aspx).

ADSI, CAPICOM.DLL and the COM+ Certificate Enrollment Control

Certificates stored in Active Directory are accessible through the ADSI interface. Hence, it is relatively easy to write VBScript code to access them. For example, the following two lines of VBScript could acquire a user's certificate.

```
# This is VBScript:
Set user = GetObject("LDAP://CN=Jason,CN=Users,DC=sans,DC=org")
cert = user.userCertificate
```

The Certificate Enrollment Control (CEnroll) used by the Certificate Services Website for managing certificates is a COM+ component. Hence, it is accessible from almost any programming language, such as C++, Visual Basic, VBScript and JScript. This permits the development of custom applications for the management of certificates and private keys. This is important because, for example, the IIS Enrollment Station webpage for the issuance of smart cards with certificates should be customized for the sake of security (discussed later in the section on smart cards).

CAPICOM.DLL provides a scriptable interface to the CryptoAPI interface. This allows you to write your own scripts to tap into the cryptographic services provided by the operating system and on-line CAs. For example, your own scripts could request and install new certificates, delete unwanted trusted root CA certificates, hash data, encrypt data, etc. You can obtain sample enrollment scripts from the CD-ROM which accompanies *Microsoft Windows Server PKI and Certificate Security* by Brian Komar

(Microsoft Press). It will likely be added to the scripts repository on Microsoft's website too, though the current author could not find it there at the time of this writing.

MSCEP.DLL and Simple Certificate Enrollment Protocol (SCEP)

MSCEP.DLL comes with the Windows Server Resource Kits. When installed, it provides support for the Simple Certificate Enrollment Protocol (SCEP) on a Windows CA. SCEP is used, for example, by Cisco routers for installing IPSec certificates to support router-to-router VPN tunnels.

Connection Manager Certificate Deployment Tool (CMGETCER.DLL)

The Connection Manager Certificate Deployment Tool (CMGetCer) is used with the Connection Manager service to automate the installation of VPN-related certificates on the systems of remote users. CMGetCer comes with the Windows Server 2003 Resource Kit Tools (a free download from Microsoft).

Miscellaneous MSDN CryptoAPI Tools

The MSDN Library at http://msdn.microsoft.com/library/ contains a listing of CryptoAPI Tools available in the Platform SDK on the MSDN CD-ROM. In the library, search the name of the tool below that you want to use. SetReg.exe and CertMgr.exe are nice for batch scripts in particular. The MSDN Library gives full command-line options.

- Cert2SPC.exe -- Creates a Software Publishers Certificate for testing purposes.
- CertMgr.exe -- Manages certificates, CTLs and CRLs.
- ChkTrust.exe -- Checks the validity of a signed file.
- MakeCert.exe -- Creates X.509 certificates for testing purposes.
- MakeCTL.exe -- Creates a Certificate Trust List (CTL) file.
- SetReg.exe -- Easily sets many registry values related to PKI.
- SignCode.exe -- Signs and timestamps a file.
- MakeCat.exe -- Creates an unsigned catalog with the hashes of other files that have been digitally signed. Catalogs are used during program installation and verification.

Background Information: How Does Windows Implement Its PKI?

The core components of the Windows PKI are:

- CryptoNG/CryptoAPI
- Cryptographic Service Providers
- Certificate Services
- Certificate Stores
- Protected Stores

CryptoNG/CryptoAPI

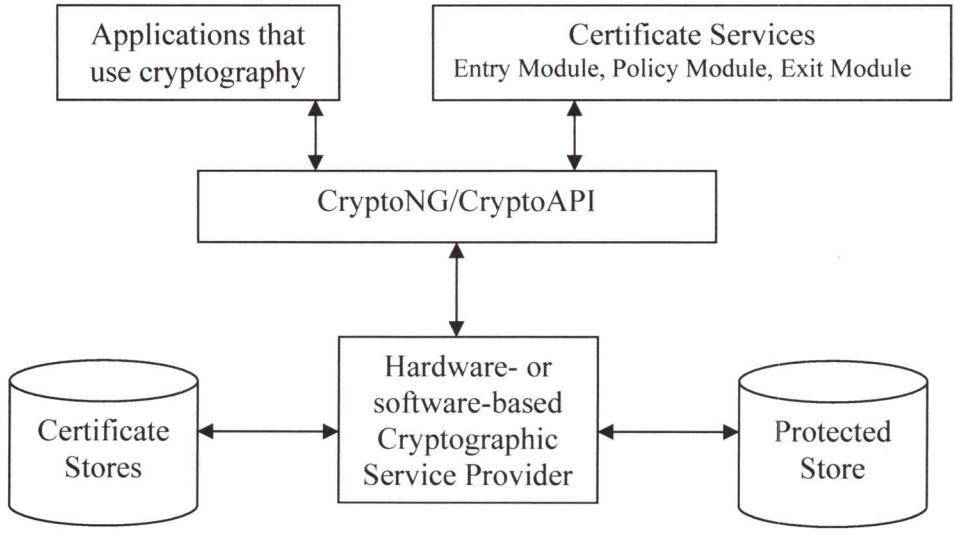
Cryptography Next Generation (CryptoNG) is the Application Programming Interface (API) through which most cryptographic services are requested. CryptoNG provides a

menu of key management, encryption, hashing, signature, trust verification and other cryptographic services to applications and the operating system so that developers can use cryptography without having to understand its details or write code to implement it directly. The older version of CryptoNG was CryptoAPI, and CryptoNG fully implements CryptoAPI for backwards compatibility. The CryptoNG/CryptoAPI documentation is available at http://msdn.microsoft.com/library/.

Importantly, there will be a single API interface no matter what type of hardware, software, ciphers or key lengths are used. CryptoNG is the most important part of implementing a public key *infrastructure*. It is also the controlled gateway through which most access to Cryptographic Service Providers must pass.

Cryptographic Service Providers

Cryptographic Service Providers (CSPs) actually do the work which the CryptoAPI menu of services offers. CSPs can be hardware or software, developed by Microsoft or not. A CSP might be a DLL, TPM chip, smart card or a cryptographic SSL accelerator device, just so long as the card or device supports CryptoAPI. Different CSPs use different algorithms, key lengths and storage mechanisms. CSPs must be digitally signed by Microsoft to work in Windows. Microsoft's software-based CSPs are compliant with at least FIPS140-1 Level 1.



Depending on which CSP is used, different ciphers and key lengths will be available. In some sectors, such as government and military, there are restrictions on which ciphers and hashing algorithms can be used. The available CSPs is mainly determined by the operating system version and service pack level; for example, support for 256-bit AES and AES-based hashing comes only with Windows Vista and later.

Certificate Services
Certificate Services (certsrv.exe) controls the generation of certificates, issues certificates and private keys to requestors, publishes certificates to Active Directory, publishes Certificate Revocation Lists (CRLs), and keeps a record of all certificate-related transactions in a separate database from Active Directory. This is the core service run by CAs. Certificate Services makes requests through CryptoAPI to various CSPs to perform its crypto-work, and through Active Directory Services Interface (ADSI) to publish to Active Directory.

Certificate Services uses DLLs called "modules" to handle certificate issuance. The "entry module" accepts PKCS#10 certificate requests via RPC or HTTP. The "policy module" determines whether a certificate request should be fulfilled, denied or left pending for an administrator's decision. The "exit module" determines how a fulfilled certificate request is returned to the requestor and how the certificate should be published, e.g., made available in Active Directory. The modules route their data and requests through Certificate Services which relays them through CryptoAPI to the necessary CSPs. Modules are DLLs that can be customized or replaced with third-party modules.

Certificate Stores
Certificates, Certificate Revocation Lists (CRLs) and Certificate Trust Lists (CTLs) are stored in "Certificate Stores". Private keys are kept elsewhere (discussed later). A Certificate Store is not a single, physical location, but a logical construct built by CryptoAPI to make it easier for CSPs to find certificates, CRLs and CTLs. Just as the DNS database has a single naming scheme but is physically distributed around the world with thousands of servers that work together, so a "Certificate Store" can be physically distributed across multiple registries, hard drive folders, databases and memory cache locations. To the CSP there is just one Certificate Store, but when the CSP searches the store, CryptoAPI will do the work of searching the various physical locations that make up the Store.

There are different types of Certificate Stores, each of which might be mapped to one or more physical locations. The name of the store in parentheses is what you will see in the registry, Active Directory and in some Microsoft documents:

- **Personal (MY) Certificate Store**: contains certificates for users, computers and services. Each will have their own separate personal store.

- **Trusted Root Certification Authorities (Root) Certificate Store**: contains certificates for CAs that your computer and applications will trust to issue valid certificates. This store can be thought of as a built-in CTL for root CAs.

- **Enterprise Trust (Trust) Certificate Store**: contains Certificate Trust Lists (CTLs) of other CA certificates not included in your Trusted Root CA Store. A CTL can specify for exactly which *purposes* a CA will be trusted to issue certificates and for how long.

- **Intermediate Certification Authority (CA) and CRL Certificate Store**: this is a performance-enhancing cache of CA certificates that need to be available when CryptoAPI checks the validity of a certificate. The CAs in this Store are *not* implicitly trusted by simply being in this Store. Importantly, this Store is where all Certificate Revocation Lists (CRLs) are also cached for efficient access. Whenever a subordinate CA certificate is obtained for path validation, a copy will be cached here indefinitely.

- **Active Directory User Objects (AD or UserDS) Certificate Store**: this is a pointer to the certificates in Active Directory associated with user accounts.

- **Request (Request) Certificate Store**: this contains the certificate requests that have been submitted to a CA and are awaiting approval or rejection.

- **Software Publisher Certificates (SPC) Certificate Store**: contains certificates for software publishers that are trusted by the local computer. This starts out as empty until the user manually approves SPCs with Internet Explorer or other code-checking applications.

- **Other People**. Similar to the Intermediate CA Store, this container simply caches the certificates of other people. Being in this container does not imply trust of the other person's certificate or their CA. When using certificates from this container, the pedigree of the certificate must originate from a trusted CA.

- **Trusted People**. Certificates stored in this container are trusted, even if the certificates were issued by a non-trusted CA or are self-signed.

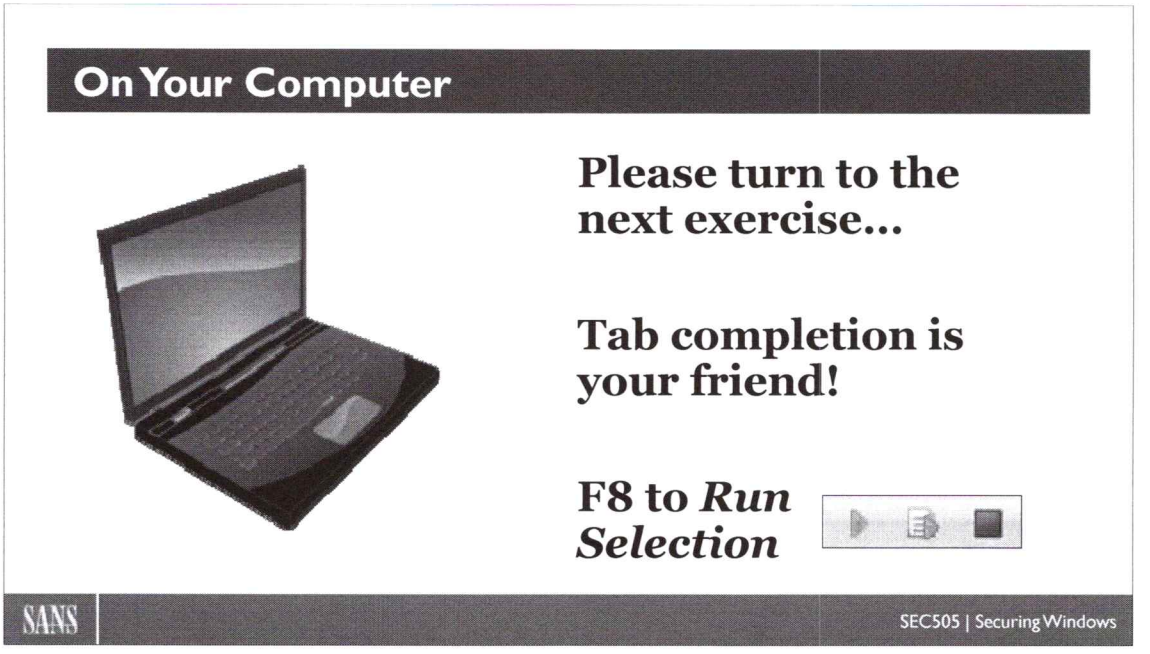
- **Untrusted Certificates**. Certificates found in this store are always untrusted, even if they were issued by a trusted CA and do not appear on the CA's CRL. (This is found on Windows XP.)

Protected Stores

Private keys are encrypted automatically by the Protected Storage service and kept in "Protected Stores". Data protected by this service are said to be put "into Protected Storage" and these data are collectively called "*the* Protected Store", but there isn't a single location which makes up the Protected Store-- it's just a catch-all term to mean anything which is secured by the Protected Storage service. The Protected Store is a *logical* store constructed by CryptoAPI that is physically composed of many different locations (just like the Certificate Stores) and each location might be protected in a different way.

The Protected Store was first introduced as a part of Internet Explorer 4. It is used by both end users and CAs to store private keys. Strictly speaking, the software-based Microsoft CSPs use Protected Storage to protect private keys; a hardware-based CSP would store private keys in the crypto hardware.

Because of the crucial importance of how private keys are distributed, stored and encrypted, the details of Protected Stores will be discussed in a later section. The short of it, however, is that your private keys are encrypted with your logon passphrase and kept in your user profile folder as individual files.



On Your Computer

This lab has multiple parts.

The Certificates Drive

In PowerShell, switch to the certificates drive and list its contents:

```
cd cert:\

dir
```

Note: Don't forget that you can use tab completion for names.

Switch to your computer's container for trusted root CA certificates and list its contents:

```
cd .\\localmachine\ca

dir
```

Get the first certificate in the list, capture it to a variable, and display its property data:

```
$cert = dir | select-object -first 1

$cert | format-list *
```

Get the public key from that certificate and display its properties and methods:

```
$pubkey = $cert.publickey.key

$pubkey | get-member
```

Public Key Encryption

To encrypt some data with a public key, such as a text string, that data must be converted into an array of System.Byte objects first, which are raw 8-bit bytes. A text string can be converted to an array of bytes, encrypted, decrypted later, then that byte array converted back to a text string again.

In the .NET Framework, there is a class named "System.Text.Encoding" which can be used for manipulating text encoded in various formats, such as ASCII and UTF-8. After converting a text string to a byte array, these bytes can be converted into other text encoding formats. PowerShell uses Unicode encoding (UTF-16) by default.

To use the properties and methods of a class from the .NET Framework, put the class name in square brackets followed by two colons ("::") and then the property or method name of that class; for example, to calculate 3^2, run "[System.Math]::Pow(3, 2)".

Convert a text string into an array of bytes encoded as UTF-8:

```
$plaintext = "my secret message"

$plainbytes = [System.Text.Encoding]::UTF8.getbytes( $plaintext )
```

When you display an array of bytes, they are shown as integer numbers in the shell:

```
$plainbytes
```

To make it easier to read, we can join the integers with commas into a string:

```
$plainbytes -join ","
```

Encrypt an array of bytes with the public key, then display the encrypted bytes:

```
$encryptedbytes = $pubkey.encrypt( $plainbytes, $false )

$encryptedbytes -join ","
```

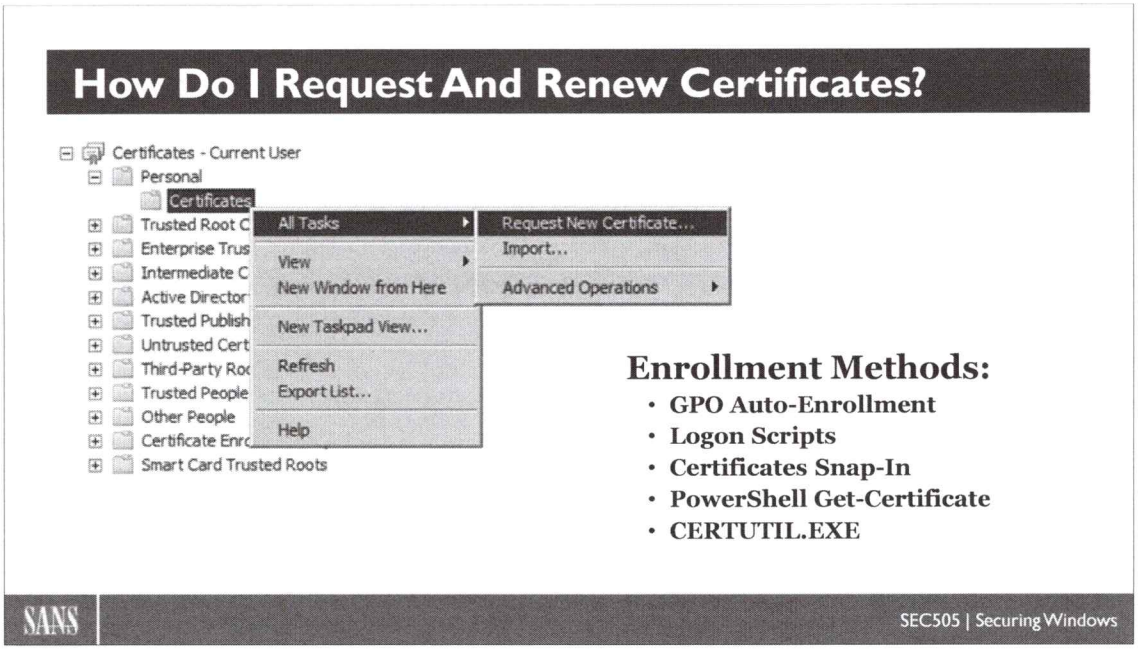
> **Note:** In the above, the $false argument means that the data should be padded, if necessary, following RFC 2437. The padding is not important to the encryption.

Save the encrypted bytes to a new file:

```
$encryptedbytes | set-content c:\temp\file.bin -encoding byte
```

We have to explicitly tell the Set-Content cmdlet that the data is composed of raw bytes or else the cmdlet will assume the data is text and it will try to encode it as Unicode.

But what about *decrypting* the data? We need the *private* key, which we don't have…yet.



How Do I Request, Delete, Renew, Import, Export, Open and Find Certificates?

The following assumes you already have an enterprise CA on your network. We will install a CA later and discuss terms like "Certificate Store".

Group Policy

Keep in mind that the primary tool for enrolling large numbers of users and computers for certificates is Group Policy. This will be discussed and demoed later.

Certificates MMC Snap-In

One tool for manually requesting, renewing, deleting, importing, exporting, opening and finding certificates is the Certificates MMC snap-in. A console for it must be created by hand. Keep in mind that regular users do not use this tool, only administrators do. Group Policy can be used to prevent users from using the Certificates snap-in if desired.

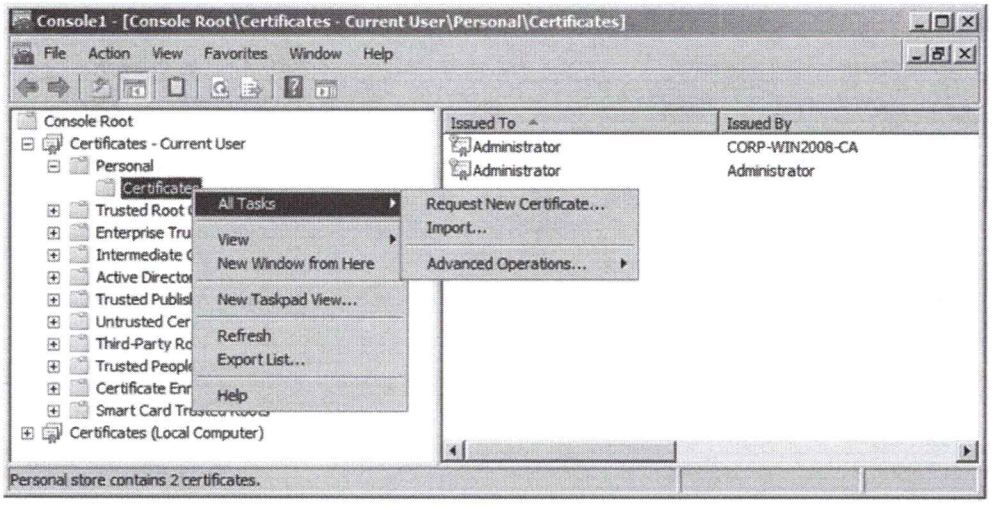
> ***Try It Now!***
>
> To create an MMC console icon to launch the Certificates snap-in, run mmc.exe from the Run line > File menu > Add/Remove Snap-In > Add > Certificates. You can choose to manage certificates for yourself, a local or remote computer, or a local or remote service account. You can add multiple Certificates snap-ins to a single console if desired. To save, pull down the File menu > Save As.

The Certificates snap-in displays the Certificates Store for the selected user, computer or service. This store includes both personal and CA certificates. Computer and service certificates are found in the Personal folder, just like user certificates.

The snap-in can optionally display the physical location of certificates and archived certificates. Recall that the Certificates Store is a logical store, composed of different physical storage locations. Old certificates are archived for recovery purposes.

Try It Now!

To display physical storage locations and archived certificates, right-click the Certificates snap-in > View > Options > check the desired boxes next to "Physical Certificate Stores" and "Archived Certificates". (Notice the option to organize certificates based on purpose as well. When this is selected, physical stores are not shown.)



Most of the tasks are completed using Wizards that will walk the user through the necessary steps. The following lists where the Wizards are located:

- **Request**: right-click Personal > All Tasks > Request New Certificate.
- **Delete:** Personal > Certificates > right-click certificate > Delete.
- **Renew:** Personal > Certificates > right-click certificate > Renew…
- **Import:** Personal > Certificates > right-click certificate > Import.
- **Export:** Personal > Certificates > right-click certificate > Export.
- **Find:** right-click any container > Find Certificates.
- **Open:** double-click any certificate.

PowerShell

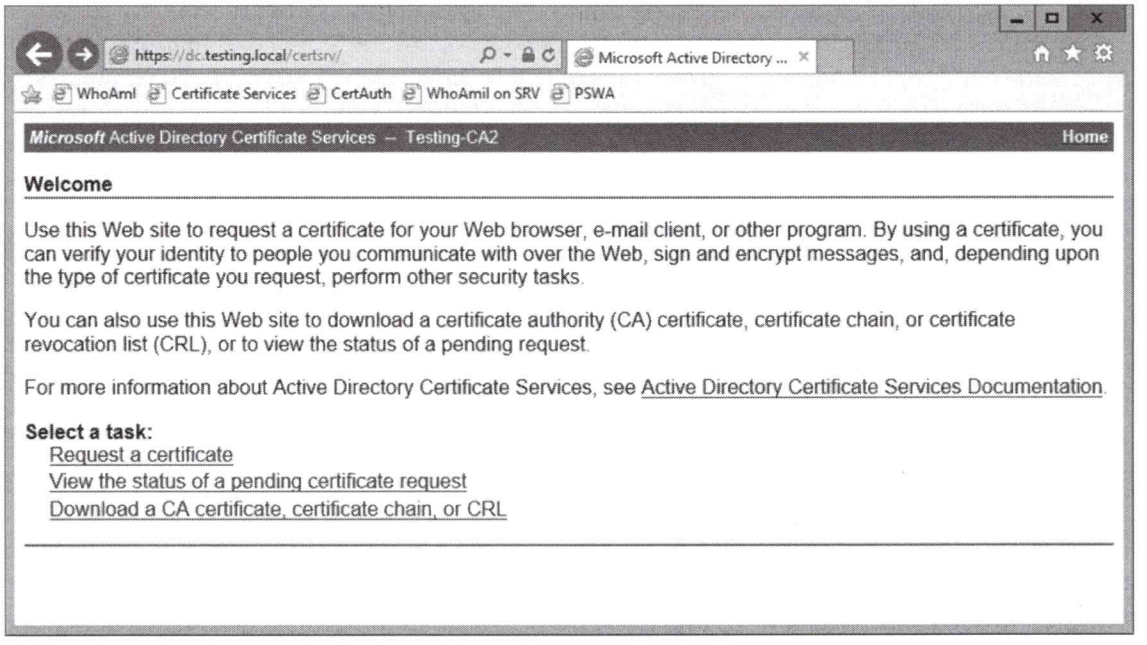
The older command-line tool for requesting new certificates is CERTUTIL.EXE. At the time of this writing, this tool still works and is still installed by default, but it has been deprecated in favor of PowerShell. When possible, avoid standardizing on this tool.

To request a personal certificate with the User template by doing an LDAP query of Active Directory for an available certificate server:

```
Get-Certificate -Template User -CertStoreLocation
 Cert:\CurrentUser\My -Url ldap
```

Certificate Services Website

An alternative to using the Certificates snap-in is the Certificate Services Website (http://*servername*/certsrv/). For 100% compatibility, the website requires Microsoft Internet Explorer. The website can be installed on the CA itself or on another IIS server.

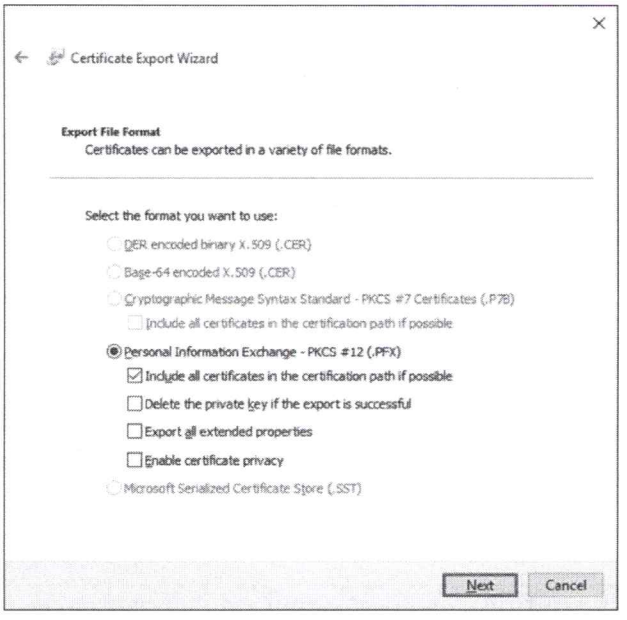


When Exporting Certificates

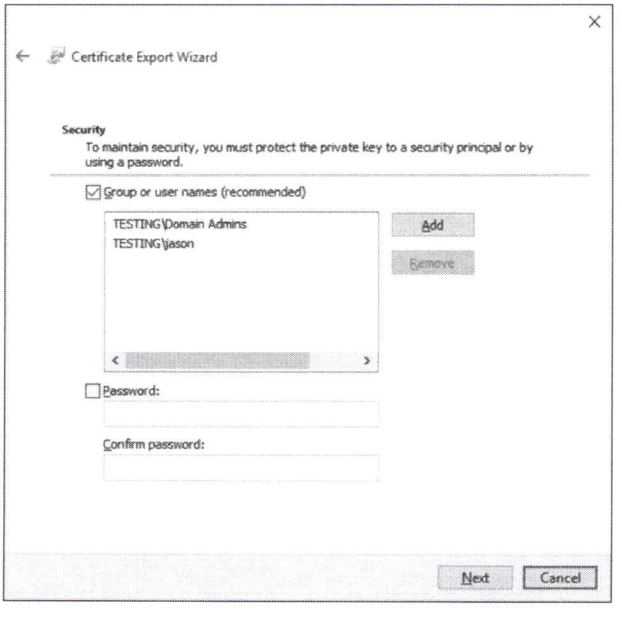
When you export a certificate you will have the option to also export the private key at the same time only if the private key has been enabled for export. This is configured when the public/private key pair is requested.

But note that public/private key pairs that can be used for digital signatures cannot be marked exportable. Look in the Key Usage field of a certificate to see if its corresponding private key can be used for digital signatures.

A password will be required to export the private key because it will be encrypted using this password, hence, use a strong password, but do not lose it or else the private key will be irrecoverable. Do not rely on password protection alone to safeguard your private key, however, since it would be easy to mount a brute-force attack against the password should the export file be stolen.



The option to "Include All Certificates In The Certification Path If Possible" is useful when transporting your public/private key pair to another computer, especially when the other computer does not already trust the issuer of your certificate. When you import on the other computer, the certificates in your CA trust path will also be imported, and your certificate will be trusted because its issuer will now be trusted by the computer.



If you are using Windows 8, Server 2012 or later on a domain member computer, and you are logged on with a global account, then you may export the private key to a .PFX file without specifying a password. Instead, you must specify the name of at least one user or group and Windows will encrypt the private key using the Data Protection API (DPAPI) such that only the specified user(s) may import the key. This requires the

import to occur on another computer with the same OS version and domain membership requirements, plus at least one domain controller running Server 2012 or later must be accessible as well. The private key is encrypted with a random passphrase (actually, the Base64 encoding of a 32-byte random number), but DPAPI will unlock the passphrase only for the intended user(s). All this is invisible to the user. If you wish to specify your own passphrase instead, this is possible in the export GUI, and this has the advantage of allowing an import with the passphrase on older operating systems too.

The Export-PfxCertificate cmdlet supports the -ProtectTo parameter for this DPAPI export functionality also.

When Renewing Certificates

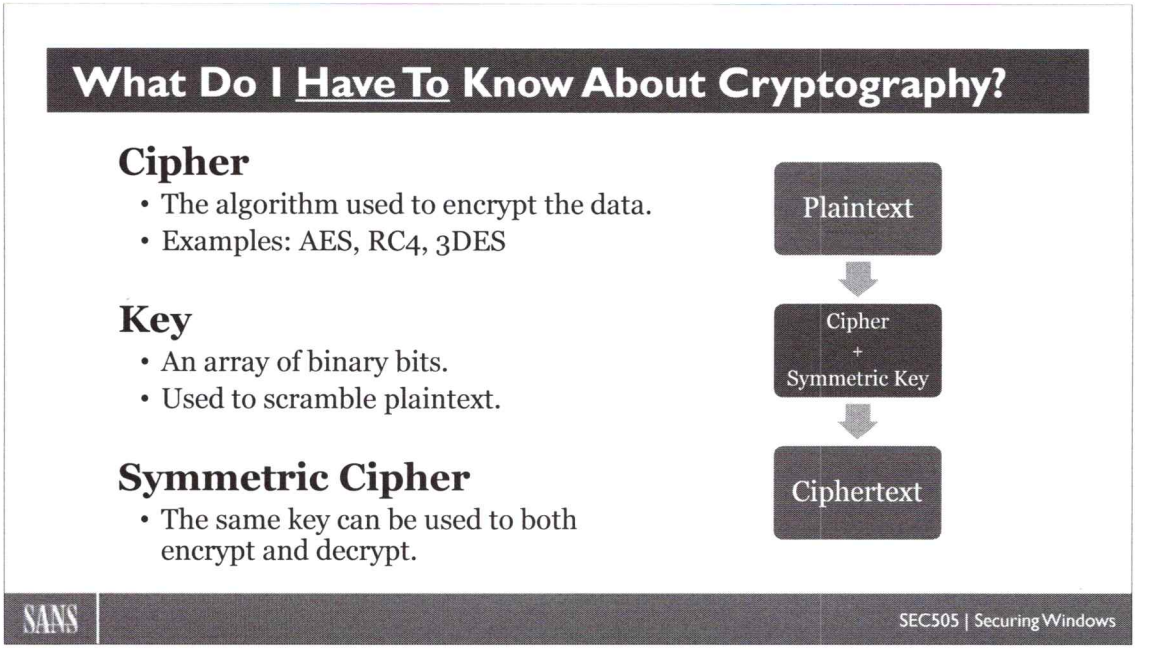
When you renew a certificate, you have the option to do so with either a new key pair or the same key pair of the certificate being renewed. To do either, right-click the certificate > All Tasks > Renew….

A CA will never issue a certificate whose ending validity date exceeds its own (issued validity dates are "nested" within the CAs time to live). Hence, give root and intermediate CA certificates relatively long life-spans, and issuing CAs shorter ones. There is a trade-off between security and convenience: the longer the life-span of a CA certificate, the less often it needs to be renewed and the less often the certificates it issues need to be renewed; on the other hand, if a CA or end-user certificate is compromised, a shorter life-span may result in less damage. Such policy issues will be decided by the particular needs of your environment.

While an issued certificate's validity period can never exceed that of its CA, it may be shorter. The maximum life-span of a certificate is determined by the following (editable) registry values:

Hive: HKEY_LOCAL_MACHINE
Key: \SYSTEM\CurrentControlSet\Services\CertSvc\Configuration*CA-name*
Value: ValidityPeriod ("Days", "Weeks", "Months", or "Years")
Value: ValidityPeriodUnits ("1", "2", "3", and so on.)

The default validity period is one year for certificates issued by stand-alone CAs, and two years for enterprise CAs. The templates used by enterprise CAs, moreover, might specify a shorter validity period than the maximum, but a template can never successfully set a validity period longer than the maximum defined in the registry. You are free to edit the registry values to suit your environment's needs.



What Do I Have To Know About Cryptography In Order To Use It?

Almost all the background information you need to know about cryptography in order to use and manage it can be summarized in just a few pages.

Benefits of Cryptography: Basic Terms

Cryptographic techniques can be used to gain the following security benefits:

- Authentication
- Data Integrity
- Non-Repudiation
- Confidentiality

All of these benefits can be provided without a PKI, but the *quality* of these benefits is often extremely high when implemented with a PKI.

> **Authentication:** "Authentication" is the act of verifying that something (person, computer, service, certificate, etc.) really is who/what it appears or claims to be.

Persons, computers and services that make use of a PKI are often called "security principals" or just "**principals**". A principal's identity is displayed as a set of credentials. A principal's "**credentials**" are its identity in a form which can be used by a security system, such as the security system in the operating system of one's computer.

An entity will have its credentials authenticated by providing one or more "**factors**" to the service doing the authentication. A factor is something an entity possesses, knows, is, or can do, which proves that its credentials are correct. For example, a "**four-factor authentication**" might require a person to insert a unique smart card into a reader (possesses), enter a password (knows), allow his or her retina to be scanned (is), and speak a few words into a voice-print analyzer (can do) in order to be successfully authenticated.

> **Data Integrity:** "Data integrity" has two senses: data has integrity if the data hasn't changed after having been copied, transmitted, or stored for a period of time; and data has integrity only if it has (or has not) changed in known ways in accordance with the expectations of the people which own or control that data.

"Integrity-checking" in this second sense usually means verifying that some data (like an e-mail to one's bank) has not been intentionally or unintentionally altered in ways which harm those doing the checking.

> **Non-Repudiation:** "Non-repudiation" is the inability to deny that a certain action was performed when there is information which shows that the action was performed, and the correctness and integrity of that information can be *proven*, perhaps in a court of law. The context here is legal and commercial transactions.

Consider a situation where you are e-mailing changes to a contract to your lawyer. You desire "**non-repudiation of delivery**" because you don't want your lawyer to be able to claim she never received it. Your lawyer, on the other hand, desires "**non-repudiation of origin**" because she doesn't want you to be able to deny that you were the one who sent it.

Strictly speaking, non-repudiation is an aspect of data integrity and key management with a focus upon legal or financial purposes. It's where PKI meets e-commerce in front of a judge and jury. Non-repudiation includes all of the hardware, people and practices involved in the generation, transmission and storage of that information which prevents deniability. Just think of all the shenanigans politicians and corporate executives engage in to maintain "plausible deniability", or think of all the arguments and tricks a lawyer would think up to say that a defendant didn't really do something— *that* is what non-repudiation is intended to prevent.

To read more about non-repudiation, see http://www.itu.int and search for the X.813 standards regarding non-repudiation.

> **Confidentiality:** Data is "confidential" if it is encrypted such that only the intended parties are able to recover the original plaintext data.

Data can also be kept confidential by using private transmission and storage media, such as physically secure fiber optic lines and hard drives locked in a safe. Often, physical

protection and encryption are used together, e.g., encrypting a USB flash drive which is normally kept in a locked drawer when not in use.

Encryption Keys, Ciphertext and Cleartext

A "**key**" is data used by a mathematical algorithm to control the conversion of understandable information ("cleartext" or "plaintext") into random-looking data ("ciphertext") that cannot be understood. This conversion process is called "**encryption**", and reversing the process to go from ciphertext back to cleartext is called "decryption".

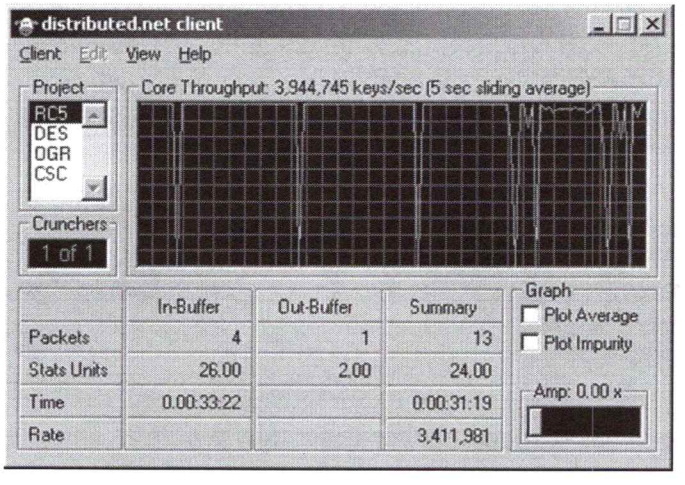
The algorithm used to encrypt data is called a "**cipher**" and encryption keys are known by the ciphers that use them. Hence, someone might ask you, "What kind of key is that?" and the answer would be to name the cipher that uses it, e.g., RC4, RSA, AES, etc.

cipher(cleartext, key) = ciphertext

Key Size/Strength, True Random Numbers, and Quantum Computers

A key is just a set of binary bits. The more bits in a key, other things being equal, the stronger the encryption it can potentially provide. For every additional bit in a key, its strength doubles, so a 101-bit key is twice as strong as a 100-bit key (with certain assumptions, bear with me here).

There is an organization (www.distributed.net) which conducts key-breaking challenges where anyone can volunteer to contribute their spare CPU cycles. You download a client from their site which will use the idle CPU time on your computer to brute-force guess encryption keys to try to crack a piece of ciphertext made for the test.



On any given day there can be more or less machines working on a distributed.net challenge, but usually there are thousands. In June of 2012, for example, the average RC5 key-guessing rate was about *one trillion* keys per *second.*

But even at this "fast" rate, it would still take just over 10,000,000,000,000,000,000 years to do a brute-force search through every possible 128-bit RC5 key (by comparison, the

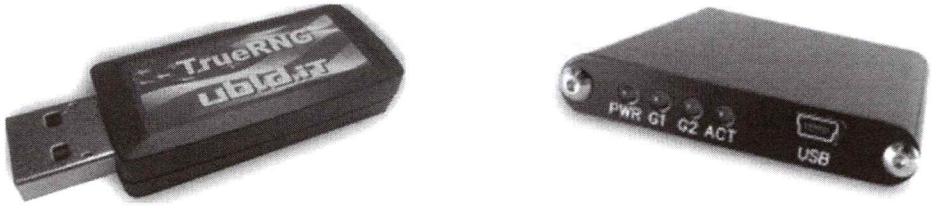
universe has only been around for about 13,800,000,000 years, a tiny fraction of that time). And a brute-force search at this rate through all possible 256-bit AES keys would require more zeros in the number than can be fit in a single line of text in this manual. (Of course, a brute-force search is the slowest method of revealing an encryption key; social engineering and installing a rootkit onto a target's computer are far more effective methods of "cracking" strong keys.)

So what do numbers like the above actually mean to us? Is there a practical point to all this? As a rough rule of thumb for non-classified data, never use symmetric keys smaller than 128 bits, and 256-bit keys are strongly preferred when available. For RSA asymmetric keys, the rule of thumb is to require 1024 bits at a bare minimum, 2048-bit keys are strongly preferred, and 4096-bit keys or better are necessary when data must be protected beyond 10 years. Be careful when using RSA keys larger than 2048 bits, though, because of software compatibility issues with older software and appliances.

> **Tip:** For more precise recommendations for different types of environments and adversaries, see http://www.keylength.com. The web site is very intuitive to use.

Vendors will often brag about what large keys they have, but a decryption vulnerability in one of their products is typically the result of a coding error or bad design, not the choice of a small key. Worrying about key sizes is usually misspent anxiety; it is far more likely for a key to be exposed through implementation flaws, malware infections or human error than for an adversary to invest the time and money to do real cracking.

One example of an implementation flaw is the use of a poorly-designed random number generator. If the bits in a key are not random, then they are predictable to some degree, which makes the key easier to crack. A special hardware device must be used to generate "true" random bits. These devices typically leverage the quantum decay of particles or the chaotic nature of certain electronic or thermal processes, such as the TrueRNG generator, which looks just like an external USB drive ($50 to $100, http://www.ubld.it).



Without a hardware-based random number generator, Windows and Linux will sample fast-changing data available through the operating system to feed into a "pseudo" random number generator algorithm. The seed data is considered to be unpredictable enough that an adversary could derive no advantage from the fact that the bits do not come from a true random source, but "good enough" is always relative to one's adversaries. True random bits are best, pseudo-random bits are what we're normally stuck with. And if a key is derived from the hash of a password chosen by a human, can humans *really* choose

random passwords? Do quantum or chaotic fluctuations in the brain really percolate all the way up to the conscious choice of a new "random" password?

Sometime in the next 50 years, however, there will be a "quantum apocalypse" for cryptography when quantum computing becomes widely available. The threat is real enough that in August of 2015 the National Security Agency (NSA) in the United States started to provide public guidance concerning quantum-resistant countermeasures:

https://www.iad.gov/iad/library/ia-guidance/

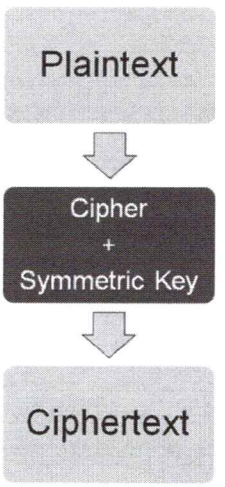
The most important points from this guidance are 1) new quantum-resistant algorithms must be developed and deployed in the next several years, 2) increase the size of keys used today, and 3) pre-shared keys can provide quantum resistance if they are very large and truly random, such as for IPSec. How large is "large"? Here are the minimum key sizes and algorithms while we wait for new quantum-resistant ciphers to be developed:

AES: 256 bits
RSA: 3072 bits
Hashing: SHA-384
Diffie-Hellman (DH) key exchange: 3072 bits
Elliptic Curve (ECDH) key exchange: curve P-384
Elliptic Curve (ECDSA) signatures: curve P-384

What in the world do these things mean? Well, let's talk about it!

Two-Way, Symmetric, Secret, Bulk, Session Keys

Some keys are "**two-way**" or "**symmetric**" in that anything encrypted with a two-way key can be decrypted with that exact same key. RC4, AES and 3DES are examples. Because the same key can be used to both encrypt and decrypt the same data, these keys have to be carefully kept secret. So, these keys are often called "**secret keys**" and should not be confused with private keys.



symmetric-cipher(cleartext, key) = ciphertext
symmetric-cipher(ciphertext, key) = cleartext

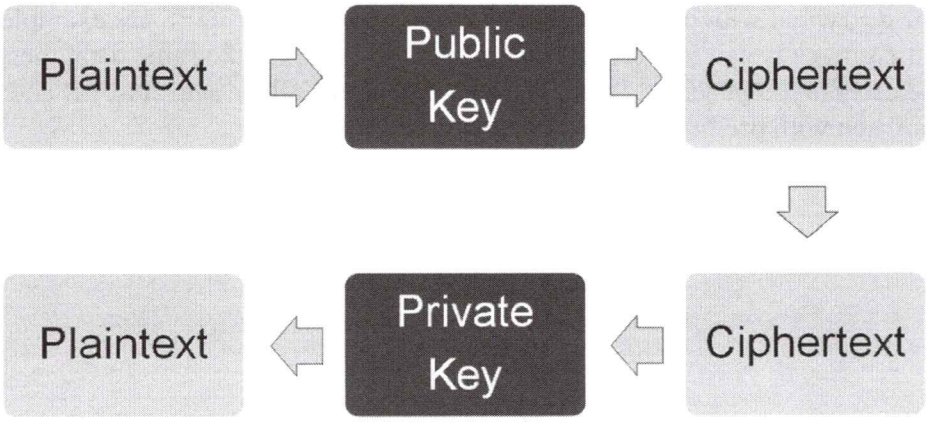
Two-way keys are fast, so they are often used for "bulk encryption" of large amounts of data. They are often called "session keys" because they encrypt almost all the data during a session with a server, except for the initial key exchange.

One-Way, Asymmetric, Public/Private, Key-Exchange Keys

Other keys are "**one-way**" or "**asymmetric**" in that they cannot be used to decrypt data which they originally encrypted. For example, RSA keys are one-way, asymmetric keys.

One-way keys are always created in pairs, and the pair has a very special property: anything encrypted with one key can only be decrypted with the other, and vice versa. For all practical purposes, for each possible one-way RSA key, there is one and only one corresponding key which can decrypt the ciphertext it produces. The one-way keys in a key pair are mathematically inter-twined together when they are generated, so they are always generated as a set.

Once generated, one of the asymmetric keys is dubbed the "**public key**" and the other, the "**private key**". They differ only in purpose, not functionality. The public key is distributed freely around the world. The private key is kept hidden and secured so that only the owner can use it. You want others to have your public key so they can encrypt data and send it to you. Only you have the corresponding private key, so only you can decrypt the data others send to you. Even if attackers have a copy of your public key, they cannot decrypt any data encrypted with that key.



cipher(cleartext, public-key) = ciphertext that only private-key can decrypt
cipher(cleartext, private-key) = ciphertext that only public-key can decrypt

One-way keys are roughly 1000 times slower than two-way keys. Hence, they are not generally used to encrypt data in bulk. Instead, one-way keys are often used to encrypt

two-way keys in order to securely share them with others, and therefore are often called "**key-exchange keys**".

Using a Key to Encrypt a Key to Encrypt a Key to Encrypt a Key...

A key is just binary data, hence, that data (that key) can be encrypted with another key. So, a 40-bit RC4 key could be encrypted with a 56-bit DES key, then *that* DES key could be encrypted with a 128-bit RC5 key, and so on. In this case, the original 40-bit key is encrypted twice with two different keys and two different ciphers. We've *layered* encryption on top of encryption.

128-bit RC5 encryption
56-bit DES encryption
40-bit RC4 key

A slightly different procedure would be to encrypt the 40-bit RC4 key using the 56-bit DES key, and then encrypt the DES key itself with the 128-bit RC5 key. Notice that, in this case, the 40-bit RC4 key has only been encrypted once.

Now we have two chunks of ciphertext: the 40-bit RC4 key encrypted with the 56-bit DES key, and the 56-bit DES key encrypted with the 128-bit RC5 key. We have not layered the encryption, we have *chained* it.

To get to the original 40-bit key, we first have to decrypt the 56-bit key with the 128-bit key, then we could decrypt the 40-bit key with the 56-bit key.

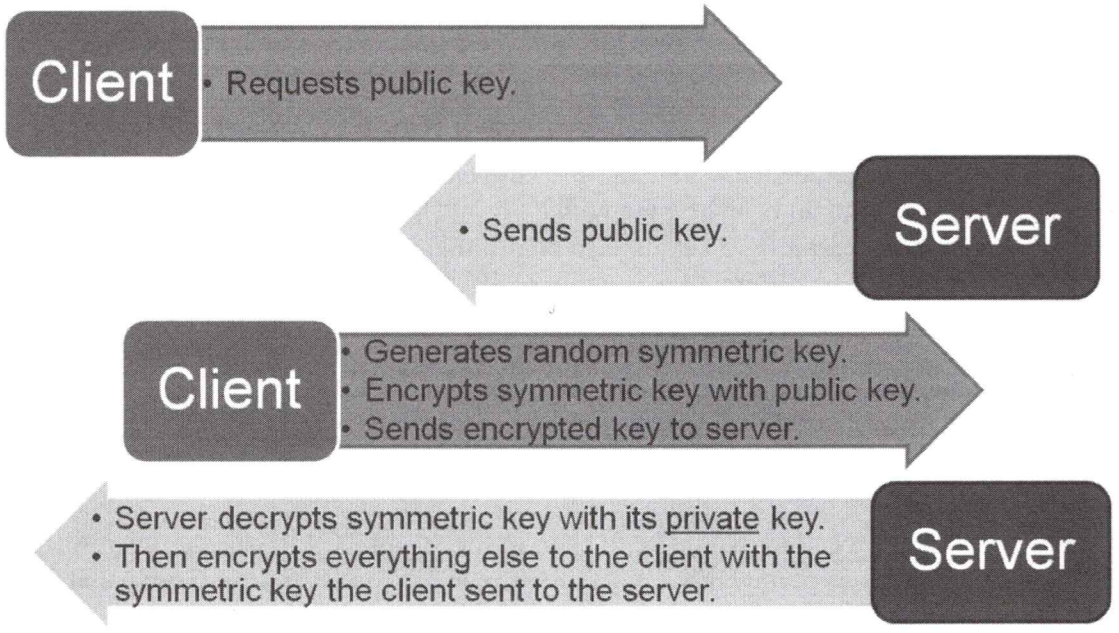
		128-bit RC5 encryption
56-bit DES encryption	←	56-bit DES key
40-bit RC4 key		

This kind of layering and chaining of encryption keys is common, especially when one of the keys is derived from the hash of a password (discussed below).

Secure Key Exchange

An important problem is how to securely share a bulk, session, two-way key with a remote party. If you merely send the two-way key to the remote party, attackers could eavesdrop on the transmission and capture the key. Because it is two-way, it could be used to decrypt any other data encrypted with it. A two-way key is desired, though, because it is fast.

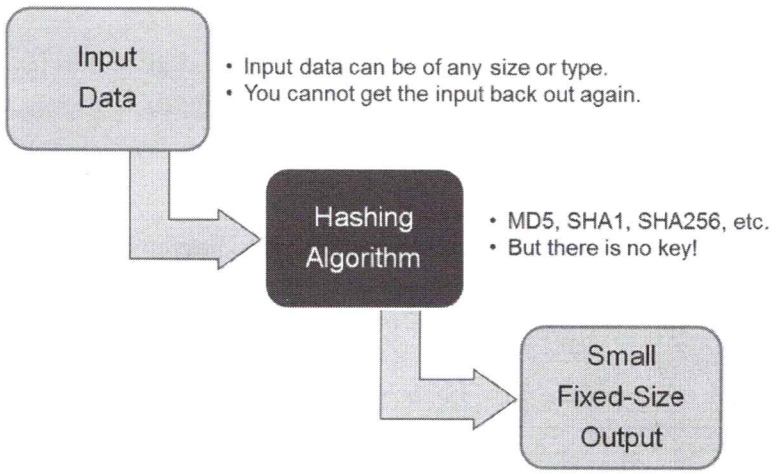
A very common solution is to 1) acquire the remote party's public one-way key, 2) randomly generate a two-way session key, 3) encrypt a copy of the session key with the other party's public key, 4) send the encrypted session key to the other party, 5) the other party decrypts the session key with their private key, and 6) encrypt all subsequent communications with the session key only. Now the two parties have securely exchanged an identical bulk encryption key.



Another secure key exchange method is called the "Diffie-Hellman technique". Two parties, each with their own public/private key pairs, can use the DH technique to mutually agree upon a shared symmetric key without ever exposing that shared key to the network. DH key negotiations are used extensively with IPSec.

Hashes, Message Digests and Fingerprints

Switching gears a bit, there is another type of algorithm called a "**hash function**". It is not used to encrypt data, but to check its integrity. A hash function takes data of any size as input, and produces a small fixed-length string of bits as output. The output is called "the hash of" or "the **digest** of" or "the **fingerprint** of" the original input. The output is sometimes called a "hash value". Examples of hash functions include MD5, SHA-1 and SHA-2. To repeat, though, because it is very commonly misunderstood: a hashing algorithm is *not* an encryption algorithm.



Importantly, a hash function is very sensitive to any changes in the original input. So, if some data is hashed (say, the Library of Congress), then that data is modified in the slightest way (say, a period changed to a comma in any document in the Library), then the modified data is hashed again, the two hash numbers produced will be different.

At an earlier time:	Hash(Data) = X
At a later time:	Hash(Data) = Y
Compare:	$X \neq Y$
Therefore:	Data has changed!

Hence, if some data is hashed, then hashed again at a later time, but the two hash numbers produced are different, this proves that the original data has somehow changed. Hence, we can use hash functions for **integrity** verification, like super-enhanced checksums.

Hashes Used as Encryption Keys

The output of a hash function is a small string of bits. There is no reason why this string of bits could not be used as an encryption key for a cipher. (Hash values can be padded or truncated to make them fit the key-requirements of a cipher.)

For example, if your password is hashed with MD5 to produce a 128-bit string, that string could be used as an encryption key with AES to encrypt other data. To decrypt the data, another person would either have to have the AES key or your password, since they could hash your password themselves to produce the exact same AES key.

Hash(Password) = string of bits = key for cipher to encrypt other data

This brings up the random number problem again. Humans are not very good random number generators, hence, keys derived from password hashes are often not very good. The best we can do is to provide training to users on how to choose less-bad passphrases and to change those passphrases on a regular basis.

Digital Signatures

To prove that some data has not been altered since it was created, and to prove that it came from a certain key-holder, the data can be **digitally signed**. One way to produce a digital signature of some data is by:

1. hashing the data to produce a hash value,
2. encrypting this hash value with the signer's private key, and
3. appending the encrypted hash to the data. The private-key encrypted hash number is the signature of the data originally hashed.

Someone can "check the signature" by 1) hashing the data independently with the same algorithm as the originator, 2) decrypt the originator's encrypted hash value with the originator's public key, and 3) comparing the originator's hash value with one's own independently calculated hash. If the two hash values are the same, this proves the data has not been altered. If the encrypted hash can be decrypted with a person's public key, this proves that their hash was encrypted with that person's corresponding private key.

The problem is proving that a certain public key really belongs to a certain person or entity. How can a set of credentials be bound to a public key to prove that *this* public key belongs to *this* entity? The problem can be solved with a trusted Certification Authority. That's what's coming up in the next section.

Want To Learn More? Cryptography Recommended Reading

This seminar is only a brief introduction to a vast literature on cryptography, PKI and smart cards. To help you get started, the following is a list of recommended readings.

Microsoft Documentation of the Windows PKI

The best book to get is the latest edition of *Microsoft Windows Server PKI and Certificate Security* by Brian Komar (MS Press). Start here and then work your way out through the more recent KB articles and white papers as needed.

See http://www.microsoft.com/technet/security/ for new whitepapers and associated papers on PKI, smart cards, CryptoAPI, Authenticode, etc.

A large number of TechNet and Knowledge Base articles exist pertaining to Certificate Services (too many to list here). Search "certificate" at http://search.microsoft.com to begin your search, then do a Search Within Results to narrow.

The Cryptography FAQ and Vendor Resources

The Cryptography FAQ is an excellent *and free* on-line reference. Start here when you have general cryptography and PK questions independent of any vendor's products.

- http://www.faqs.org/faqs/cryptography-faq/

A famous book in cryptography, *Handbook of Applied Cryptography*, by Menezes, Oorschot and Vanstone, is available for free in PDF format from here:

- http://www.cacr.math.uwaterloo.ca/hac/

The link below will take you to an article entitled "Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure" by C. Ellison and B. Schneier. This is a seven-page summary of the pitfalls of relying on PKI for security. The page also has other interesting material concerning practical cryptography and security:

- http://www.schneier.com/paper-pki.html

Central Cryptography Texts
Schneier's *Cryptography Engineering* is the book to buy if you want to purchase only one. Schneier also publishes a free monthly e-mail bulletin on security and cryptography issues that is good (http://www.schneier.com/crypto-gram.html), and the blog at this site is very good too.

For a book that focuses on PKI, *Understanding Public-Key Infrastructure*, by Adams and Lloyd, canvases all the essentials in RFC-like fashion. *Digital Certificates*, by Feghhi and Williams, has a more practical, but less comprehensive, approach. Some of the books are old, but most cryptography fundamentals have stayed the same.

- *Cryptography Engineering*, Schneier, et al. (John Wiley & Sons: 2010).
- *Cryptography and Network Security 2nd Edition*, Stallings (Prentice Hall: 1999).
- *Cryptography: Theory and Practice*, Stinson (CRC Press: 1995).
- *Digital Certificates*, Feghhi and Williams (Addison-Wesley: 1999)
- *Handbook of Applied Cryptography*, Menezes, et al. (CRC Press: 2001)
- *Understanding Public-Key Infrastructure*, Adams and Lloyd (Macmillan: 1999).

Microsoft CryptoAPI, CAPICOM and CryptoNG
Microsoft's CryptoAPI and CAPICOM documentation can be found in the Platform SDK of the MSDN Online Library at http://msdn.microsoft.com/library/. Look in the Platform SDK > Security > Cryptography section. If you want to write scripts in PowerShell or Python that use PKI, then read the CryptoNG (CNG) documentation.

Acronyms and Abbreviations

3DES = Triple-DES: 168-bit DES using three 56-bit keys or 112-bits with two keys.
AD = Active Directory (ntds.dit)
AIA = Authority Information Access (URLs where a CA's certificates are found)
AKI = Authority Key Identifier (certificate serial and ID number a CA uses)
BDC = Backup Domain Controller (NT 4.0)
CA = Certification Authority (an issuer of digital certificates, e.g., VeriSign)
CDP = CRL Distribution Point (where a CA's CRLs can be found)
CNG = Microsoft Cryptography Next Generation (succeeds CryptoAPI)
CRL = Certificate Revocation List (a CA's list of revoked certificates)

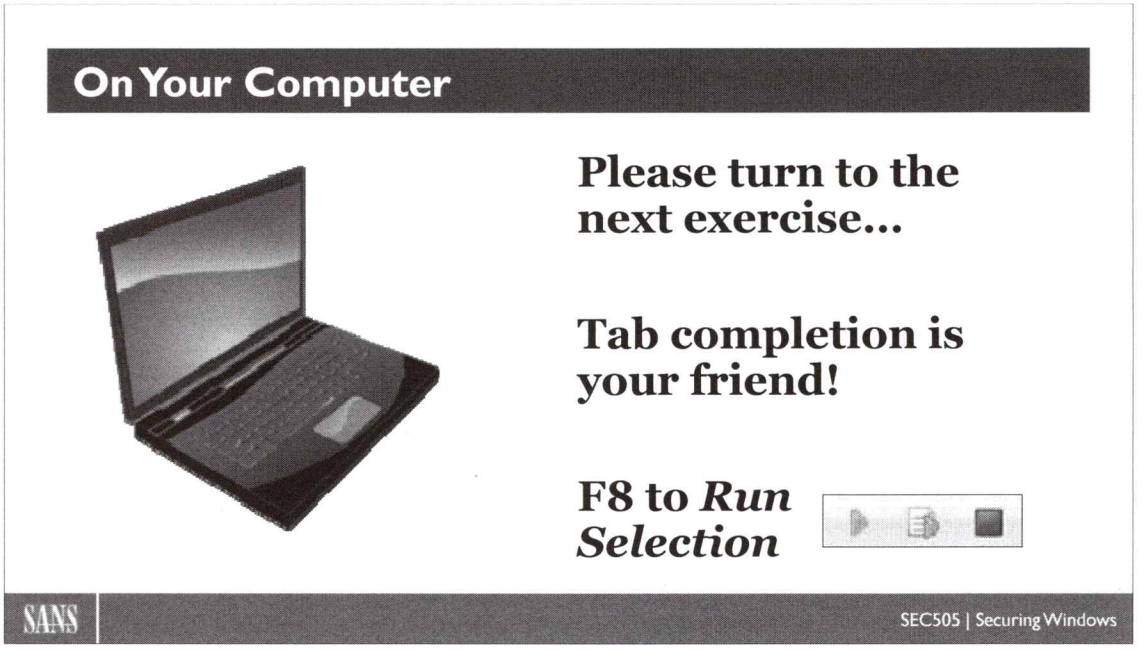
CSP = Cryptographic Service Provider (performs cryptographic operations)
CTL = Certificate Trust List (list of trusted CAs)
CryptoAPI = Menu of cryptographic services provided by Windows.
DACL = Discretionary Access Control List (permissions on an object)
DC = Domain Controller (dcpromo.exe)
DDF = Data Decryption Field (EFS file attribute with encrypted FEK)
DDNS = Dynamic Domain Name System (DNS automatic update)
DES = A 56-bit symmetric block encryption cipher developed by NSA.
DFS = Distributed File System (virtualized shared folders)
DS = Directory Service (AD, Novell NDS, and Unix NIS are examples)
DRF = Data Recovery Field (EFS file attribute with encrypted FEK)
EFS = Encrypting File System (NTFS transparent file encryption)
FEK = File Encryption Key (EFS key used to encrypt a file)
GC = Global Catalog (index of AD)
GPO = Group Policy Object (a set of configuration settings)
HSM = Hardware Security Module (CA private key protection)
IETF = Internet Engineering Task Force (publishes RFCs)
IIS = Internet Information Server (MS's HTTP and FTP server)
IPSec = Internet Protocol Security (encryption and signing of IP data)
ISA = Internet Security and Acceleration Server (MS's firewall, NAT, proxy)
ITU = International Telecommunications Union (sets standards)
KDC = Key Distribution Center (DC serving Kerberos tickets)
Member Server = non-DC which has a computer account in a domain
MMC = Microsoft Management Console (mmc.exe)
MS = Microsoft Corporation (www.microsoft.com)
NC = Naming Context (a partition of the AD database)
Native = Native Mode operation on pure Windows networks
NETLOGON = A folder shared as NETLOGON (part of SYSVOL)
NT = Microsoft Windows NT 4.0 (www.microsoft.com)
NTFS = NT File System (permissions, auditing, compression, encryption)
OID = Object Identifier (a number to uniquely identify an object type in a DS)
OU = Organizational Unit (subdivision of a domain)
PDC = Primary Domain Controller (NT 4.0)
PGP = Pretty Good Privacy (web of trust PK software: www.pgp.com)
PK = Public key (a one-way asymmetric encryption key)
PKCS = Set of *de facto* PK standards from RSA Laboratories (PKCS#7)
PKI = Public Key Infrastructure (standardized services surrounding use of PKs)
PKIX = IETF working group to define an interoperable PKI
RFC = Request for Comments (a document defining/proposing standards)
RRAS = Routing and Remote Access Server (MS's dial-up/VPN router)
RSA = Rivest-Shamir-Adleman (popular algorithm for PKs)
S/MIME = PK-based encryption and signing of e-mail (Outlook)
SAM = Security Accounts Manager (NT 4.0 SAM database of accounts)
SACL = System Access Control List (list of audit settings on an object)
SGC = Server Gated Cryptography (SSL for banks; non-issue now)

Snap-In = MMC snap-in tool (e.g., AD Users and Computers)
SSL = Secure Sockets Layer (PK-based encryption, signature and authentication)
Stand-Alone Computer = non-DC which is not a member of any domain
Stand-Alone CA = Windows CA which does not use AD or templates
SYSVOL = A folder shared as SYSVOL with scripts and GPOs
TLS = Transport Layer Security (next version of SSL)
VPN = Virtual Private Networking (encryption and tunneling of packets)
X.500 = ITU standard for a directory service (like Active Directory)
X.509v3 = ITU standard for a digital certificate format, version 3 (RFC 2459)

PKCS Message Types

RSA Laboratories has developed a number of Public Key Cryptography Standard (PKCS) message types which have become widely accepted in the industry and are used in Windows. Some of the more important PKCS standards are the following:

- PKCS#5 is a method of deriving a private key from a password.
- PKCS#7 is a format for storing certificates, but not private keys.
- PKCS#10 is a format for certificate requests to be sent to CA's.
- PKCS#12 is a format for securely storing certificates and private keys (.pfx files).



On Your Computer

Please switch to the C:\Temp directory:

```
cd C:\Temp
```

Hash all the text files in C:\Temp with SHA-256:

```
Get-FileHash -Algorithm sha256 -Path C:\Temp\*.txt
```

Note: Get-FileHash is available only on Server 2012 R2 and later.

Save these hashes to a comma-delimited CSV file for reference (before.csv):

```
Get-FileHash -Algorithm sha256 -Path C:\Temp\*.txt |
  Export-Csv -Path before.csv
```

Append some text to the end of the Low-Integrity.txt file, which will change its hash:

```
"AAA" | Out-File -Append -FilePath .\Low-Integrity.txt
```

Rename the Medium-Integrity.txt file ("ren" is an alias for Rename-Item):

```
ren .\Medium-Integrity.txt .\Medium-Integrity-File.txt
```

Create a new text file:

```
"AAA" | Out-File -FilePath .\NewFile.txt
```

Rehash all the text files again, saving the output to a new CSV file (after.csv):

```
Get-FileHash -Algorithm sha256 -Path C:\Temp\*.txt |
  Export-Csv -Path after.csv
```

Compare two CSV files with hashes to find out what has changed:

```
.\Compare-FileHashesList.ps1 -ReferenceFile .\before.csv
  -DifferenceFile .\after.csv -IncludeUnchanged
```

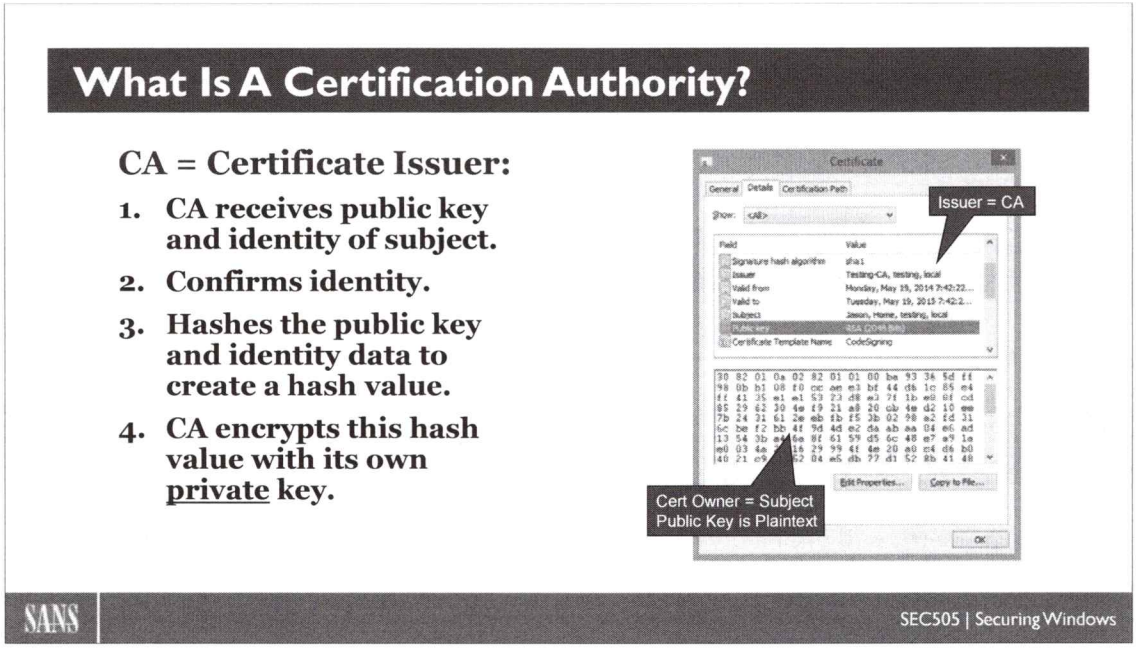
The Status property will either say "New", "Missing", "Changed" or "Same" as the two CSV files are compared. The following table summarizes the meaning of the Status property. (Without the -IncludeUnchanged switch, the "Same" files are suppressed.)

Status	Meaning
Missing	File is listed in the reference or baseline CSV (on the left) but not in the difference CSV (on the right); it may have been deleted, renamed or moved.
New	File was not in the original baseline CSV (on the left), but does exist in the difference CSV (on the right); it may be a new or renamed file.
Changed	File is listed in both of the CSV's, but with different hash values recorded.
Same	File is listed in both CSV's and has identical hashes in both CSV's too.

Compare two CSV files with hashes, but only show a summary:

```
.\Compare-FileHashesList.ps1 -ReferenceFile .\before.csv
  -DifferenceFile .\after.csv -SummaryOnly
```

If you are more familiar with MD5DEEP, then the Compare-FileHashesList.ps1 script can compare the output files of that tool too (see https://github.com/jessek/hashdeep/).



What Is A Certification Authority?

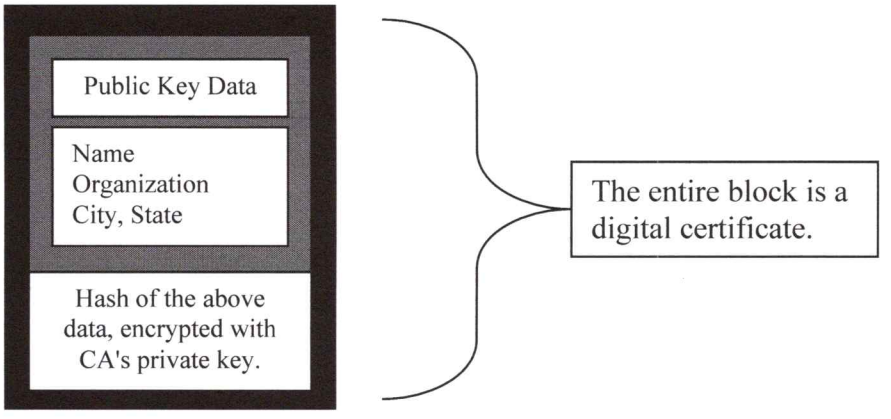
A Certification Authority (CA) binds credentials to public keys. "Binding" a public key to a set of credentials is a way of linking that key to a person or entity in a way which can be verified by others. An example of a CA is VeriSign (www.verisign.com). The short of it is this: if I trust your CA, then I can verify that the public key you have given me really does belong to you and that it has not been altered since it was created.

Again, "credentials" are simply a description (or set of descriptions) of a person, computer, organization, service or other security principal. A typical set of credentials for a person might look like this:

Common Name (CN) = Jason Fossen
Organization (O) = SANS Institute
Organizational Unit (OU) = Instructors
Locality (L) = San Diego
State (S) = CA
Country (C) = US
E-mail (E) = jason@sans.org

Credentials are "bound" to a public key by 1) including both the credentials and public key in single file or document, 2) hashing the document, 3) encrypting the hash value with the *private* key of the CA, and 4) appending the encrypted hash value to the end of the document. In short, CA's simply digitally sign documents which include both a public key and a set of credentials. In doing so, the CA is asserting that "*This* public key

in this document belongs to the person, server, organization, service or other principal specified in the credentials also in this document".



How Do I Know the Certificate Was Not Forged Or Altered?

The "binding" between the credentials and the public key can be checked by 1) taking the *public* key of the CA and decrypting the hash appended to the document, 2) hashing the credentials and public key with the same hash function that the CA used, and 3) comparing your hash value with the hash value decrypted from the document. If the hash values perfectly match, this proves the document has not been altered in any way from the time when it was created.

Importantly, the bare fact that you were able to decrypt the CA's hash number proves that it was encrypted with the private key of the CA. Since you have the public key of the CA, and there is only one corresponding private key, and that private key is in the protected possession of the CA, and anything encrypted with that private key can only be decrypted with the CA's public key, then being able to decrypt the hash with the CA's public key proves that it was encrypted by the CA. If you trust the CA to always truthfully bind credentials, then you can trust that *this* public key goes with *this* principal whose credentials are in the document.

How Do I Know I Really Have the CA's Public Key?

The public key of the CA is included in the CA's certificate. A root CA (see below) signs its own certificate, i.e., the subject and the issuer are identical. The public key contained in the certificate is used to check the certificate's own signature and integrity. This will at least confirm that the public key has not been modified in the original certificate, even though the signature was left untouched.

An attacker could replace VeriSign's entire certificate with his own on your computer, and you would trust certificates issued by the attacker as being from VeriSign, but every other certificate from VeriSign that you check (e-mail messages, webserver certificates, etc.) would fail. This is an indication that something is wrong.

You can also download and import certificates from CAs directly to replace any certificate on your system that you suspect is bogus. Or you could do this as a preventative measure every night by scheduling the download. True, an attacker could intercept your download request from the CA's website, but the difficulties for the attacker are rising steeply....

You can compare your copy of the CA's certificate with the copies installed on the computers of people you trust, especially people in other organizations on other networks. True, an attacker could replace them all in anticipation....

You can also call the CA on the phone and ask for the thumbprint of the CA's certificate. A "thumbprint" is a hash value of a certificate which is short enough to easily check on the phone (this is its intended purpose actually). This thumbprint hash is calculated when you open/view the certificate. True, an attacker could intercept your phone call....

You can also physically visit the offices of the CA and request their certificate in person. True, an attacker could trick you into visiting a bogus office with a false front....

But Why Should I Trust The CA?

Excellent question, but beyond the scope of this course. This issue is hotly debated in cryptography circles...and in three-letter government agencies. The books in the Recommended Reading section of this course are a good place to start.

If you only trust yourself, then obtain a free copy of some of the best encryption software available: Pretty Good Privacy (PGP) at http://www.pgp.com/products/freeware/. PGP is the *de facto* standard for file and e-mail encryption, with numerous plug-ins to make it easy to use with Windows Explorer and most of the popular e-mail applications like Eudora and Outlook. However, PGP does not participate in the hierarchical trust model used by Windows PKIs.

What Is An X.509 Digital Certificate?

When a CA binds a public key to a set of credentials by signing a document that includes both, this document becomes a "digital certificate". Windows uses industry-standard X.509 version 3 certificates.

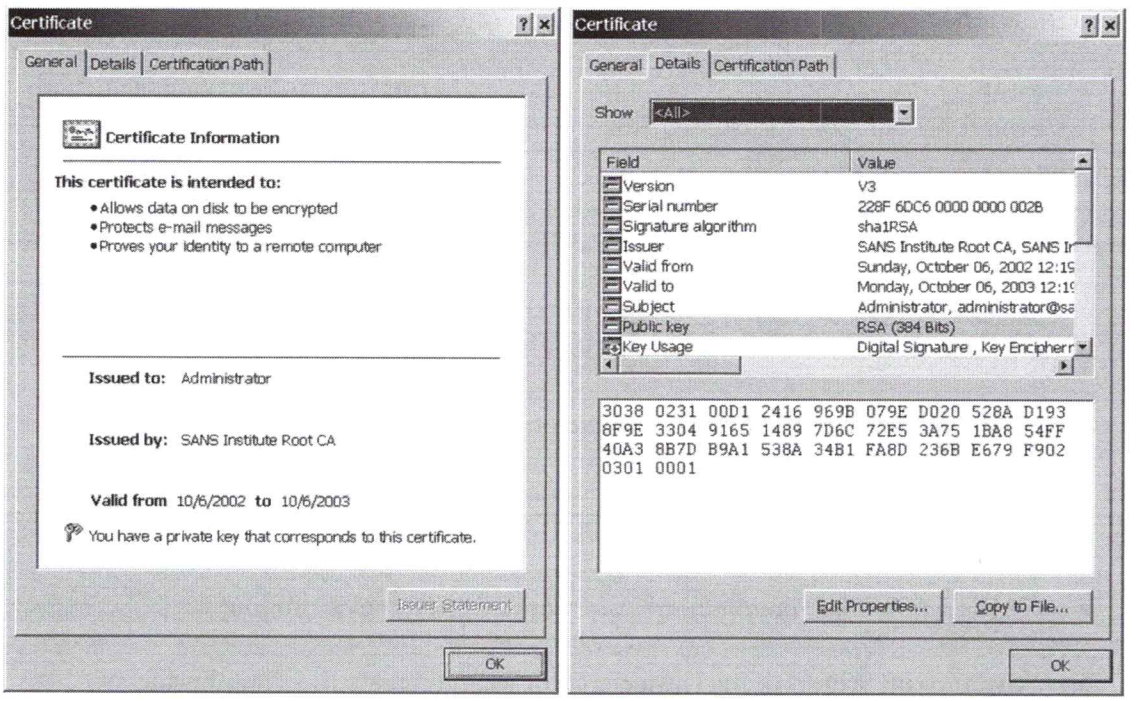
> **Note:** X.509 is an International Telecommunications Union (ITU) standard (http://www.itu.int) that has been further developed by the Public Key Infrastructure working group (PKIX) of the Internet Engineering Task Force (IETF) as defined in RFC 2459.

X.509v3 Structure

An X.509v3 certificate must contain, at a minimum, the following data fields:

- **Version Number**: the version type of the certificate.
- **Serial Number**: a unique number assigned by the CA.

- **Signature Algorithm**: identifies the CA's signing hash function.
- **Issuer Name**: the name of the CA which issued the certificate.
- **Valid From**: the starting date and time when the certificate can be used.
- **Valid To**: the ending date and time when the certificate can be used.
- **Subject Name**: the name of the owner of the certificate.
- **Public Key**: the public key itself plus information on how it was created.
- **Signed Hash**: a hash of the above data, encrypted with the CA's private key.



An X.509v3 certificate may also optionally include the following information as extensions to the standard format. These "certificate extensions" are also bound by the CA's encrypted hash.

- **CRL Distribution Point (CDP):** This identifies where one or more Certificate Revocation Lists can be found which would include the present certificate if it had been revoked. The location can be defined with LDAP://, HTTP:// or FILE:// URL paths.

- **Certificate Template:** Windows uses templates to define the structure of different types of certificates. This item identifies which template the enterprise CA used to create this certificate. Stand-alone CAs do not use templates.

- **Basic Constraints:** Indicates whether or not the certificate is for a CA and how many levels deep below it in a CA trust hierarchy certificates can be issued by subordinate CAs.

- **Authority Information Access (AIA):** Defines where the certificate of the issuing CA can be obtained. The location can be defined with LDAP://, HTTP:// or FILE:// URL paths.

- **Friendly Name:** Simply a user-friendly display name in MMC snap-ins.

- **Key Usage:** Lists the basic purposes for which the certificate can be used, such as Digital Signature, Key Encipherment, Non-Repudiation, Certificate Signing, Off-line CRL Signing, CRL Signing, etc.

- **Enhanced Key Usage:** Lists additional special-purpose key uses, such as Secure E-mail, Client Authentication, Smart Card Logon, etc.

- **Subject Key Identifier:** A hash of the public key used to identify the public key when it is referenced in other documents and in the registry.

- **Authority Key Identifier (AKI):** Identifies the public key of the issuing CA by the CA's Subject Key Identifier, Issuer Name, and Serial Number.

- **Subject Alternative Name:** Lists one or more alternative valid names for the subject (owner) of the certificate. For example, a Windows User Principal Name (UPN) looks like an e-mail address and may be found here. The UPN is used to map certificates to user accounts in Active Directory.

- **CA Version:** The version number of Certificate Services running.

- **Thumbprint Algorithm:** Hash function used to create the Thumbprint.

- **Thumbprint:** An easy-to-read hash of the certificate that can be, for example, read over the phone to help validate a certificate as being the desired one.

Can My Users Be Issued Multiple Certificates?

Yes. In fact, this is common practice. And these certificates do not all have to be issued by your organization.

A user might have a certificate installed on a smart card just for Kerberos logon, another certificate just for digitally signing contracts, a third certificate for official e-mail, a fourth certificate for personal e-mail, a fifth certificate for the Encrypting File System, a sixth certificate used with PGP, a seventh personal certificate for remote authentication to an on-line bank or stock-trading account, and ten more certificates (and private keys) that are no longer used but need to be archived.

Simplifying the management and use of so many certificates is part of what a PKI is supposed to provide.

What Are The Benefits Of Having Multiple Certificates?

Not all certificates have the same purpose. The private keys of extremely sensitive certificates may require protection measures that make them inconvenient to use. These private keys may be on slow smart cards or require a long password every time they are invoked. Hence, it may be more efficient to have a separate certificate for non-critical purposes that is easier to use.

Not everyone will trust your organization's CA. A vendor, supplier, bank, government agency, etc. may prefer to issue you a certificate themselves because they don't want to trust your organization's CA wholesale. Just as you might have multiple ID cards from many other groups/agencies/organizations, so you might have multiple certificates.

Multiple certificates and private keys provide fault tolerance. If all contracts are signed with a single private key, then the compromise of that key could be devastating legally. If all of one's data is encrypted with a single public key, then the compromise of its private key could reveal everything.

Due to federal laws or export restrictions, some certificates might be permitted to have very long key-lengths (such as signing-only certificates) while others must be shorter (such as certificates for data encryption).

How Will Anyone Know If A Certificate Is Valid Or Not?

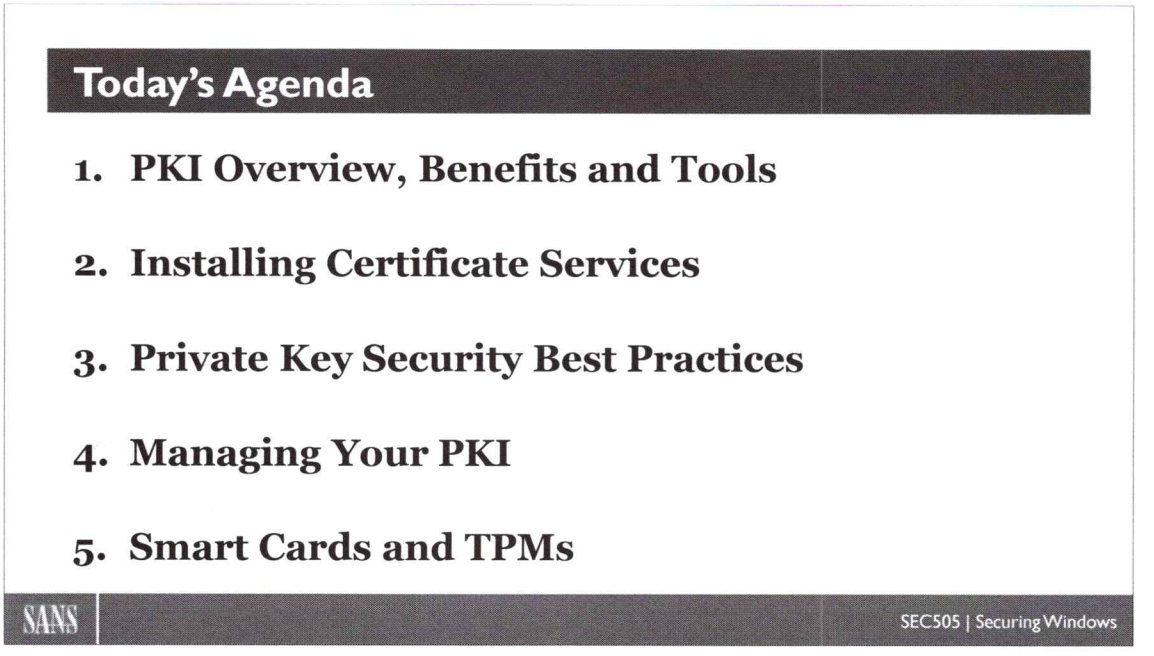
A certificate's "path" is its digital signature pedigree from the issuing CA that created it up to a root CA. The next section discusses CA hierarchy paths.

"Path validation" is a procedure in which a certificate's path is verified to be correct and trusted. Path validation involves obtaining every CA certificate in the path, checking their signatures, validity dates, integrity, and ensuring that the root CA is trusted.

CryptoAPI handles path validation automatically for applications that request it for a certificate. The following summarizes CryptoAPI's certification path validation procedure:

1. Check certificate type and purpose.
2. Check certificate start and end valid dates.
3. Check certificate integrity and signature.
4. If the issuing CA is not a root CA, then obtain all CA certificates in the path up to the root. CryptoAPI will use the AKI, AIA and issuer name fields in each CA certificate to find the necessary parent certificates in Certificate Stores. If any certificate fails checks for type, purpose, validity dates, or integrity, fail validation check. If the entire path is good, and the root CA is trusted, then the test stops, returning a success.
5. If the root CA is not in the Trusted Root CAs container, then check to see if it is listed on a Certificate Trust List (CTL) in the Enterprise Trust container. If not, fail validation. If it is, then check whether desired purpose is supported; if it is not, fail; if it is, return success.

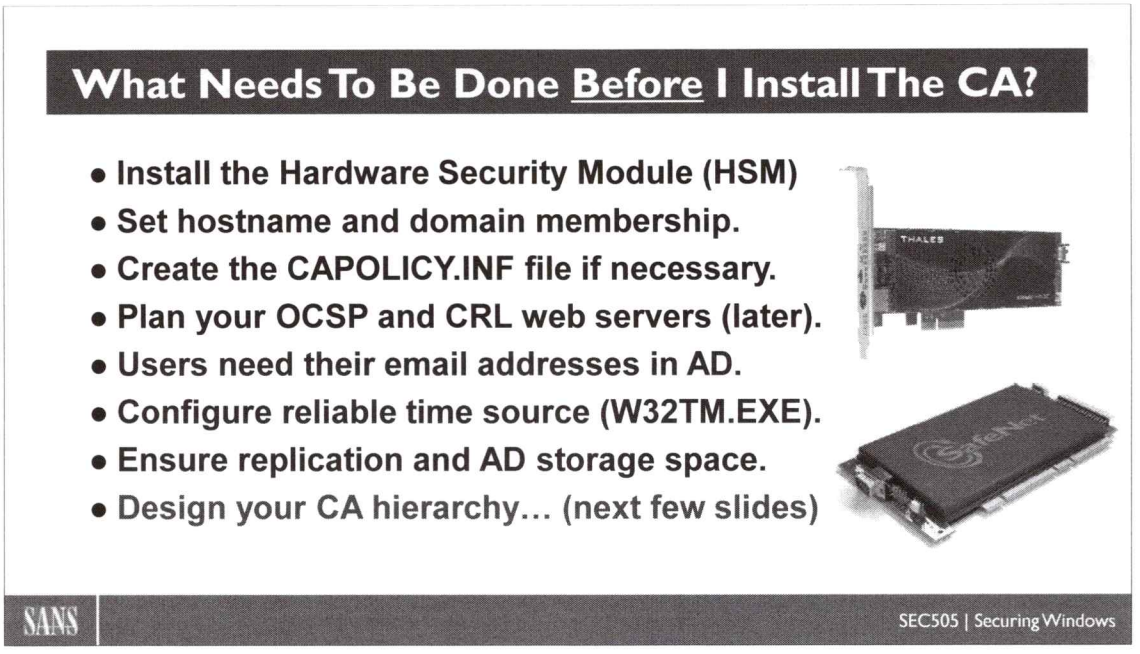
Note that if a CA certificate has expired, but the certificate being checked has not, then this does not cause path validation to fail. As long as the certificate being checked was issued when the CA's certificate was still valid, then the certificate being checked may still be valid (assuming it passes the other regular tests).



Today's Agenda

Now that we understand what a PKI is, what its benefits are for network security, and have a solid grounding in the essentials of cryptography, let's install Windows Certificate Services and start issuing digital certificates.

Certificate Services is an optional component that is installed on Windows Server. It is that service which actually accepts requests for certificates and creates them. You don't have to buy your certificates one at a time from a commercial CA, you can be your own Certification Authority.



What Needs To Be Done Before I Install The CA?

The next few sections will walk through the decision process for designing a Windows PKI and the installation steps for installing the CAs.

Install The Hardware Security Module (HSM)

If you plan to use a Hardware Security Module (HSM) to store the private key of the CA, now is the time to install it. HSM vendors, such as Thales (www.thales-esecurity.com) and SafeNet (www.safenet-inc.com), will have extensive documentation and recommendations that must be understood before moving any further.

It is possible to use a smart card or Trusted Platform Module (TPM) chip to generate and protect the CA's private keys. This is good for security, but not ideal for fault tolerance. If the smart card or TPM fails, then the private keys being protected are gone forever. It is better to invest in an HSM specifically designed for a Windows PKI.

Upgrade The Active Directory Schema If Necessary

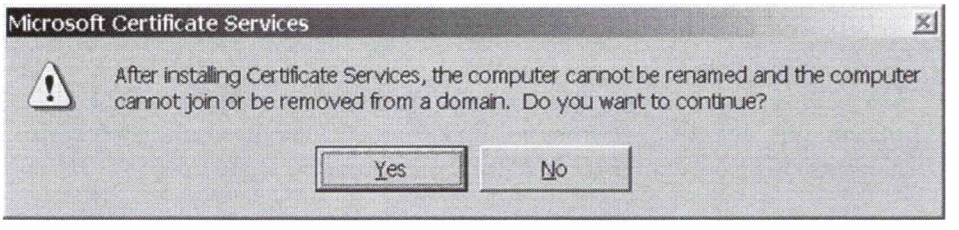
The absolute minimum forest functionality level for a PKI is the Server 2003 level, but, in general, it's best to upgrade to the latest available before you install Certificate Services for the first time. There are many dependencies between PKI and Active Directory, and it's safer to upgrade the AD schema before the PKI is deployed rather than afterwards. If you want to use OCSP for revocation checking, for example, you must have the Server 2008 or later schema.

For guidance on how to upgrade the AD schema, search Microsoft's web site for the terms "Active Directory Migration Tool" and "ADPREP.EXE", especially if you have

very old domain controllers. You may need to upgrade the OS on the Schema Operations Master and the Infrastructure Master domain controllers in the AD forest. Upgrading an AD schema is not a particularly risky or complex operation, it is done all the time, but it does require some planning and testing first. If you have Microsoft Exchange, then those administrators are probably familiar with the procedure and they will especially need to be involved in the planning and testing.

Computer Name and Domain Membership

Once Certificate Services is installed on a computer, that computer's name and domain membership cannot be changed without uninstalling Certificate Services first. If you install on a domain controller, it also means you can't uninstall and then reinstall Active Directory in order to troubleshoot problems with AD.



Also, a Windows CA can be a CA for an entire forest of domains, or a single domain can contain multiple CAs. A stand-alone CA need not be a member of a domain at all. We will discuss domain membership more when we talk about Enterprise versus Stand-Alone CAs.

CAPOLICY.INF

When a CA certificate is created or renewed, a number of options can be defined for it by creating and modifying a configuration file named Capolicy.inf. This file does not exist by default. The Capolicy.inf file must be created in the %WinDir% folder *before* Certificate Services is installed or *before* the CA's certificate is renewed again.

A sample Capolicy.inf file is on the course CD handed out by the instructor.

How do we know when this file should be created? When is it necessary? If the answer to any of the following questions is Yes, then you should create the file:

- Do you want to publish a CRL that users can check to confirm that the certificate of your root CA has not been revoked? This is not done by default for fault tolerance, performance and application compatibility reasons.

- Do you want to make copies of the root CA's certificates available at more network locations than just Active Directory? This is not done by default because root CA certificates are already published in AD and installed on clients through Group Policy.

- Do you need to include a URL to a "Certificate Policy Statement" in each certificate issued by the CA? This URL might point towards a PDF or HTML document that might include legal disclaimers, information about the CA, contact information, etc. This might be required for business or legal purposes, especially in the European Union.

- Do you want to limit the types or purposes of the certificates issued by the CA? By defining Enhanced Key Usage (EKU) extension OID numbers, your CA can be deliberately constrained, such as for delegation of authority.

- Do you need to prevent the CA chain or path from the root CA to all of its subordinates from growing too long? You can prevent a CA from issuing subordinate certificates to other CAs. Conversely, you could allow a CA to issue certificates only to other CAs, never to end users or computers.

- Do you want to enforce a particular CA public key length during renewals? Note that the key length for a new public key is defined at creation time, not here in the capolicy.inf file, which is only used for renewals.

- Do you want to enforce a particular CA public key validity period for renewals? Note that the initial validity period for a new key is defined at creation time.

- Are you deploying a "Cross CA" or "Bridge CA"? Do you need to implement name constraints or enforce application policies for the cross/bridge relationship? This will be rare and is not covered in this course.

If the answer is Yes to any of the above, search on "capolicy.inf" in the Certificate Services documentation on Microsoft's web site for guidance. Also recommended is Brian Komar's book, *Windows Server PKI And Certificate Security*, which provides the same guidance.

Certificate Policy Statement vs. Certificate Practices Statement
A Certificate Policy Statement is a relatively high-level description of the CA, the CA's ownership and country/state of origin, the purpose(s) of the certificate, legal liability disclaimers, and other general information. Instead of trying to include all this in the certificate itself, it is common to simply include a URL to a document or web page which discusses these issues in depth (and which can be changed).

In contrast to the Certificate *Policy* Statement is the Certification *Practices* Statement. The Practices Statement is a detailed, low-level description of the exact methods a CA uses to authenticate requests, generate keys, certificate extensions used, method of returning key pairs to principals, certificate archival, revocation procedures, and other important day-to-day operations of the CA. The Practices Statement generally should not be made publicly available, except perhaps to external parties with a legitimate need-to-know, e.g., auditors or large clients, with non-disclosure agreements. Commercial CAs

which provide PKI out-sourcing may need to make their Practices Statements public, however, as per the laws of the State where the CA is domiciled.

The Certificate Policy and Certificate Practices Statements are important because a PKI is only as secure as the people who manage it. Human beings are the weakest link in a PKI. For more information, see RFC 2527: "The Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework".

Plan Your OCSP and CRL Distribution Web Servers

Users and computers both inside the LAN and roaming around the Internet will need to be able to contact special web servers of yours which will support your PKI. These web servers are needed to support the checking of the revocation status of your issued certificates.

> **Note:** Certificate revocation checking and configuration will be covered later.

At least one web server will need to be accessible from the Internet and from the internal LAN, but it is highly recommended to have two or more web servers for the sake of fault tolerance and load balancing. If you would prefer to have different sets of web servers for internal versus external access, even better.

Specifically, you will need at least one physical or VM web server with the Web Server (IIS) role installed. The web server(s) will run the Online Certificate Status Protocol (OCSP) responder application. The web server(s) will also serve up Certificate Revocation List (CRL) files for download via HTTP. You will need to plan the FQDN of the server(s) and how the clustering will be accomplished, e.g., hardware load balancer, DNS round robin, reverse proxy server, etc. The FQDN and URLs needed will be discussed later in the section on certificate revocation.

User E-Mail Addresses In Active Directory

The e-mail address field on the General tab of a user's property sheet in AD Users and Computers needs to be completed before that user is issued a certificate intended for securing e-mail. Without that information the user's certificate will be issued without an e-mail address (E=) and cannot be used for signing mail. Moreover, the e-mail addresses of users cannot be changed after distributing certificates to them unless you are willing to issue new certificates. Therefore, the e-mail system will likely need to be planned and installed first. The e-mail address naming scheme should cohere with (or be identical to) the user account naming scheme in the forest as well.

If you have Exchange Server, then users' e-mail addresses are already in AD. If you have another e-mail system, you will likely need to import users' e-mail addresses into AD and then keep these addresses up-to-date and in sync with your e-mail system. Fortunately, PowerShell scripting of AD is relatively easy and we will see a number of examples this week.

Time Synchronization

For a variety of reasons, domain controllers and CAs should be time-synchronized with the *correct* time and date. Windows computers automatically synchronize their clocks with their domain controllers, domain controllers sync their clocks with the PDC Emulator operations master domain controller in their home domains, and the PDC Emulator in each subdomain sync its clock with the PDC Emulator in the root domain of the forest. Hence, at a minimum, the PDC Emulator domain controller in the root domain of the forest should be synchronized via Network Time Protocol (NTP) with a reliable time server, such as pool.ntp.org. This can be done with the w32tm.exe:

```
w32tm.exe /config /manualpeerlist:pool.ntp.org
 /syncfromflags:manual /update
```

Even more secure is to set up your own internal NTP server using an ordinary GPS receiver. With a free clock-to-GPS synchronization utility, your internal NTP server will not be dependent on Internet access and will thus be less vulnerable to NTP attacks. Configure your domain controllers, firewalls, IDS sensors and other time-sensitive devices to use your own NTP server (domain members can just use their controllers). Search Microsoft's web site on the terms "Windows Time Service" and/or "w32tm.exe" for more information.

Active Directory Sizing and Replication

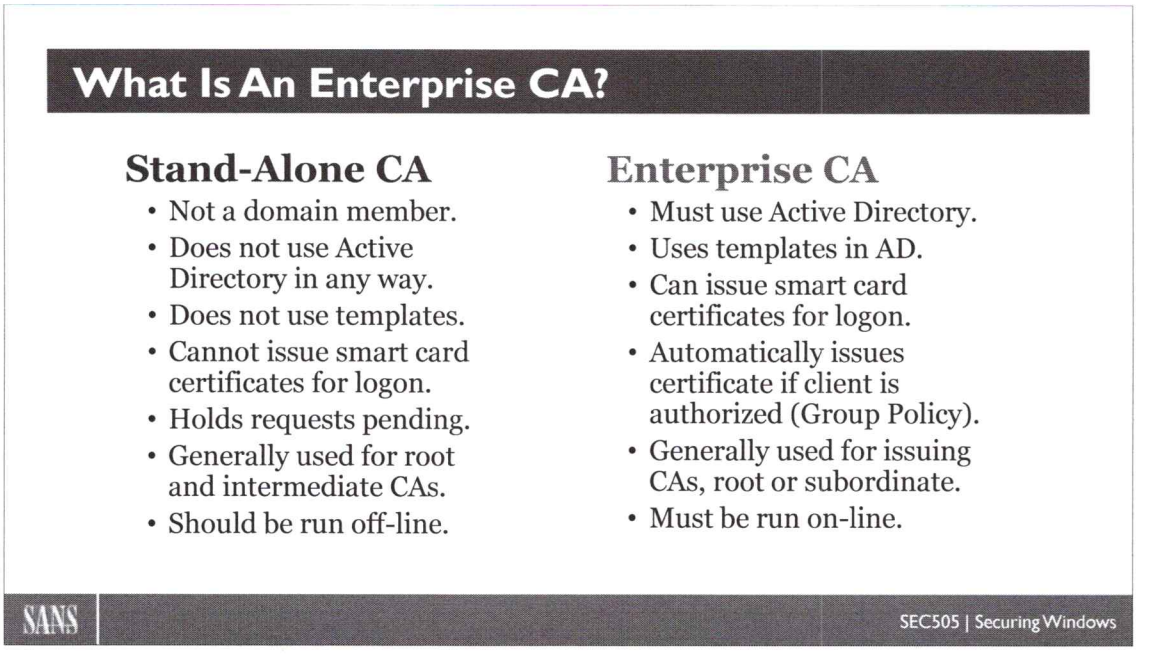
Copies of CRLs and most certificates are stored in the Active Directory Global Catalog. This means every user's certificate is replicated to every domain controller in the entire enterprise. Your domain controllers need both the storage space and bandwidth to replicate this data. This author has seen a client CA database with over 800,000 certificates, but this company had tens of thousands of users and computers.

Failover Clustering

If you require maximum fault tolerance, you can also have a two-node active/passive failover cluster for your CA with Server 2008 and later. You can also simply have multiple certificate servers subordinate to your root CA; if one CA fails, the others can continue issuing certificates. However, it is rarely the CAs themselves that require clustering and load balancing, it is usually more important to provide load balancing and fault tolerance for the web servers which support OCSP and CRL distribution.

Design Your CA Hierarchy

Finally, you must design your Certification Authority hierarchy before installing CAs. Next few slides!



What Is An Enterprise CA?

When Certificate Services is installed, there is a choice of "policy module" which determines how Certificate Services will authenticate users, issue certificates, enroll computer accounts, and publish other information. This is the most important choice for the type of CA you install.

There are two modules to choose from: Enterprise and Stand-Alone. "Stand-alone" is used with two senses. Any computer is a stand-alone if it is not a member of a domain. A CA is a "stand-alone CA" if its policy module is not the enterprise policy module. These are two different meanings, and a stand-alone CA does not necessarily have to be a stand-alone computer with respect to domain membership.

The following table summarizes the differences between enterprise and stand-alone CA policy modules. The modules are discussed below.

	Stand-Alone CA	Enterprise CA
Generally used for root and intermediate CAs	Yes	No
Generally used for issuing CAs	No	Yes
Uses Active Directory	No (not by design)	Yes. Required.
Generally run "off-line" or disconnected from network	Yes	Cannot, since access to Active Directory required.
Uses templates to create certificates	No. Cannot.	Yes. Required.
Can issue Smart Card User or	No	Yes

Smart Card Logon certificates		
Supports computer auto-enrollment	No	Yes
Must be installed on a computer which is a member of a Windows domain	Optional. Often not done for the sake of added security.	Yes. Access to Active Directory is required.
Can be installed on a domain controller	Yes. Not much point in doing this though.	Yes. Might be done to increase performance, but hassles occur if you try to reinstall Active Directory.
Can use the Certificate Services Website	Yes	Yes
Certificate Services Website must be installed on the CA itself	No	No
Policy DLL can be replaced or customized	Yes	Not recommended, but yes
Automatically issues certificates after receiving requests	No, requests remain pending until approved by an administrator.	Yes, since the user authenticated to Active Directory.
All information in certificate request must be entered by hand	Yes, since neither templates nor Active Directory is used.	No, since templates and information from Active Directory are used to populate these fields.
Can issue certificates to users or computers in other domains of the forest	Yes, even to users completely outside of the forest.	Yes, but not to anyone outside the forest.
Must manually publish CA certificate and CRLs to Active Directory	Yes, for the most part.	No, integration with Active Directory is automatic.

Enterprise CA Details

Enterprise CAs are typically used as subordinate issuing CAs, but it is possible for them to be root or intermediate CAs as well. An enterprise CA must be a member server or domain controller. Keep in mind, though, that you cannot change the domain membership of a CA once certificate services is installed, hence, you won't be able to uninstall and/or reinstall Active Directory to troubleshoot AD after you've made the box a CA.

> **Note**: You must be a member of the Enterprise Admins group in order to install a CA with the Enterprise Policy Module.

The enterprise policy module uses Active Directory to authenticate users, to find the information required for the credentials in certificates, and to make published certificates

available for search on the network. Hence, an enterprise CA must be run "on-line" or connected to a network with access to Active Directory.

A user or computer must have an account in Active Directory in order to request a certificate from an enterprise CA. If a user or computer successfully authenticates to the domain, an enterprise CA will rely on this authentication and *automatically* issue any requested certificate to the user/computer for which the user/computer has the requisite permissions.

> **Important!** Assuming a user has the necessary permissions, an enterprise CA will issue a certificate to any user who successfully authenticates to the domain. No manual administrator approval is required. Hence, you must enforce strong passphrase and lockout policies to help prevent attackers from acquiring certificates.

An enterprise CA will use the account information in Active Directory, such as e-mail addresses, to populate the credentials in requested certificates. This means users do not have to manually type in their credential information when they request a certificate, and "typo" errors are significantly reduced.

Enterprise CAs use templates to format all certificate requests. The template extracts the necessary identifying information from Active Directory when the certificate is created.

After the certificate is created, the enterprise CA will publish it to Active Directory automatically. This makes it available to all users and computers throughout the forest because certificates are a part of the Global Catalog.

Only enterprise CAs support computer auto-enrollment and smart card certificates that can be used for Kerberos authentication to the domain. Stand-alone CAs do not support either of these features.

Stand-Alone CA Details

Stand-alone CAs are usually root or intermediate CAs that do not issue certificates directly to end users. The exception is when the users are outside of one's organization.

A stand-alone CA is typically a stand-alone server, i.e., it is not a member of any domain. Stand-alones are often run "off-line" or completely disconnected from any network. (See below for on-line vs. off-line discussion.) A stand-alone CA operates independently of Active Directory.

Stand-alone CA's issue certificates primarily through the Certificate Services Website (http://*servername*/certsrv/). All interactions are manual. A user must enter all identifying credential information by hand. The certificate request will remain pending until the CA administrator explicitly reviews and approves the request. Then the user must manually download and install the certificate. The certificate is not published to Active Directory automatically, but it may be imported there manually. Usually, the CA

administrator and "user" are the same person because the certificate being requested is for an intermediate or issuing CA.

> **Note:** If a stand-alone CA is installed in the root domain of the forest as a member server, then the CA's certificate and CRL will be published to Active Directory, but other certificates will not. CERTUTIL.EXE can always be used to publish the CA's certificates to AD, however, no matter whether it is a member server or not.

Stand-alone CAs do not use templates to create certificates. Hence, all information for the certificate issued must be entered by hand.

Smart card logon certificates and computer auto-enrollment are not supported. Smart card certificates for purposes *other* than log on, e.g., secure e-mail, can be issued by a stand-alone CA however.

Importantly, the stand-alone CA policy can be customized or replaced by another policy DLL if desired. This permits third-party developers to integrate Certificate Services with their own PKI solutions.

On-Line vs. Off-Line Operation

An enterprise CA must operate on-line in order to use Active Directory and to transparently enroll users and computers. A stand-alone CA can be on-line as well, but this is an unnecessary risk.

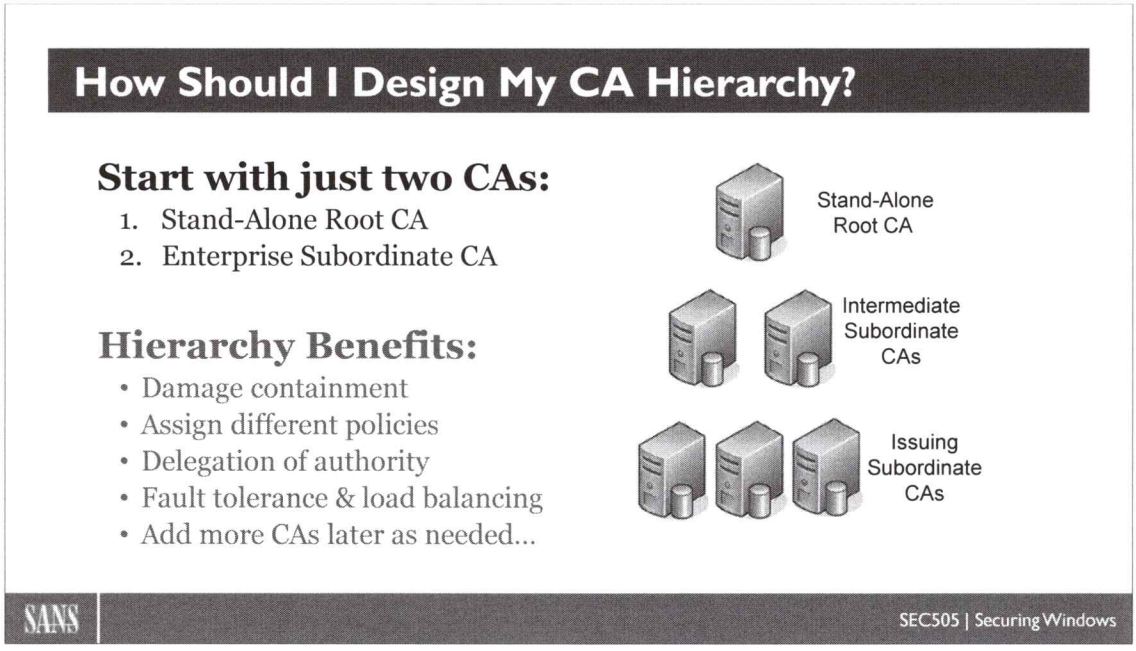
When not in use, the hard drive(s) of root/intermediate stand-alone CAs should be removed and stored in a safe or lockbox. (This is facilitated by using cartridge drives, or drives mounted in caddies that are inserted into a special bay on the front of the computer.) The drive is inserted and the computer booted only for those relatively rare occasions when the private key of the CA is needed, e.g., issuing a new intermediate CA certificate, signing a CRL which revokes a CA certificate, etc. Even then, the certificate or CRL generated should be transported by hand with a flash disk, not transmitted over the network, since the off-line CA should never be connected to any network.

Off-line CAs are easy and cheap to protect: lock their drives in a safe. It is the on-line CAs which may require expensive cryptographic hardware.

> **Note**: Some templates are tagged as "off-line templates", but this is a different use of the term "off-line". Off-line templates are used with the Certificate Services Website, which might nonetheless be accessible from the network or Internet.

When installing a subordinate CA with a certificate generated by an off-line root CA, the certificate will be in the form of a PKCS#12 file (.pfx extension). During the installation, you will have the option of either generating a public/private key pair or importing the

pair from a PKCS#12 file. You will use the import option. Import is easy because a Wizard will walk you through the process. After importing the certificate and private key, the flash disk with the PKCS#12 file should be locked in the safe with the hard drives of the off-line CAs as a backup.



How Should I Design My CA Hierarchy?

Only a root CA obtains its certificate from itself. Most CAs receive their certificates from other CAs. A root CA can create a certificate, not for a user, but for another CA; this CA in turn can issue certificates to other CAs, thus forming a chain or hierarchy, until end users and computers receive their certificates.

Root CA

A "root CA" has a public/private key pair where the private key was used to sign the public key. A root CA is "self-signed" in that the issuer and subject fields in the root CA's certificate are identical. A hierarchical PKI begins with a root CA.

Subordinate CA

But not all CAs are root CAs. In fact, the majority of CAs are not roots. The public key certificate of a CA must be signed by somebody, but it could another CA, and that other CA could be owned and controlled by another organization (like VeriSign) or by one's own organization.

If you examine the properties of any certificate, the Certificate Path tab in the properties shows where that certificate came from, i.e., who signed and issued it.

For example, in the screenshot above we can see the Certification Path tab from the properties of a certificate for a CA named "DOD EMAIL CA-32". The United States Department of Defense has a root CA named "DoD Root CA2" and its private key was used to sign this certificate, the one whose properties we are viewing, which is not the root. Both are CAs, the one at the top is a root CA, while the lower one is a subordinate CA to that root.

Hence, a "subordinate CA" is a CA whose public key was signed with the private key of another CA. The parent CA is often a root CA, but the parent signer could actually be another subordinate. Hence, there can be a chain of signings from the root CA down through one or more subordinate CAs. This chain can be followed up or down, so it is often referred to as a "path", i.e., "walking the CA path up to the root" or "the application validates the certificate path" or "a trusted certificate must chain up to a trusted root". A user or computer certificate must be signed by a CA, but the issuing CA can be either a root or a subordinate CA anywhere in the path.

For example, in the following screenshot from the certificate of a subordinate CA, the path starts with the root at the top, a subordinate in the middle (called an "intermediate subordinate"), and another subordinate at the bottom, which is the certificate whose properties are shown in the screenshot.

In this course we will soon install a root enterprise CA, but that is because we only have one VM to use. In real life, though, the recommendation in the next section is that you install 1) a stand-alone root CA and use it to sign the certificate of 2) an enterprise

subordinate CA. You could then install additional subordinate CAs as you wish, either stand-alone or enterprise, when you design your CA hierarchy.

Terminology

CAs are categorized by where they obtain their certificates and to whom they issue certificates:

- **Root CA**. A "root CA" has a self-signed certificate. That is to say, the subject and issuer fields in its CA certificate are identical. Root CAs can issue certificates directly to end users, but typically it will only issue certificates to other CAs instead.

- **Intermediate CA**. If a subordinate CA only issues certificates to other CAs, not to end users, then it is called an "intermediate CA".

- **Issuing CA**. If a CA issues certificates directly to end users and computers, then it is called an "issuing CA".

A "CA hierarchy" (also called a "PKI hierarchy") is two or more CAs which are in a parent-subordinate relationship. One CA will be the root, while the other(s) will be subordinate to it directly or indirectly. At the bottom of the hierarchy are the issuing CAs which create certificates only for end users, not other CAs. In the middle layer(s) of the hierarchy are zero or more intermediate CAs.

Are All PKIs Single-Root Hierarchies?

No, there are other PKI structures besides single-root hierarchies. A "cross-certified network" of CAs exists when a CA has more than one certificate from more than one hierarchy. Windows can use Certificate Trust Lists (CTLs) or an organization can install multiple CAs to achieve a similar result to cross-certification.

There is also the "web of trust" model made popular by the Pretty Good Privacy (PGP) program (http://www.pgp.com). In a web of trust, there are no official CAs: each certificate holder can sign others' certificates to vouch that the credentials in the certificate are correct, and others can sign one's own certificate to vouch that your certificate really is yours. In this way, a web of signatures can bind a network of certificate users together.

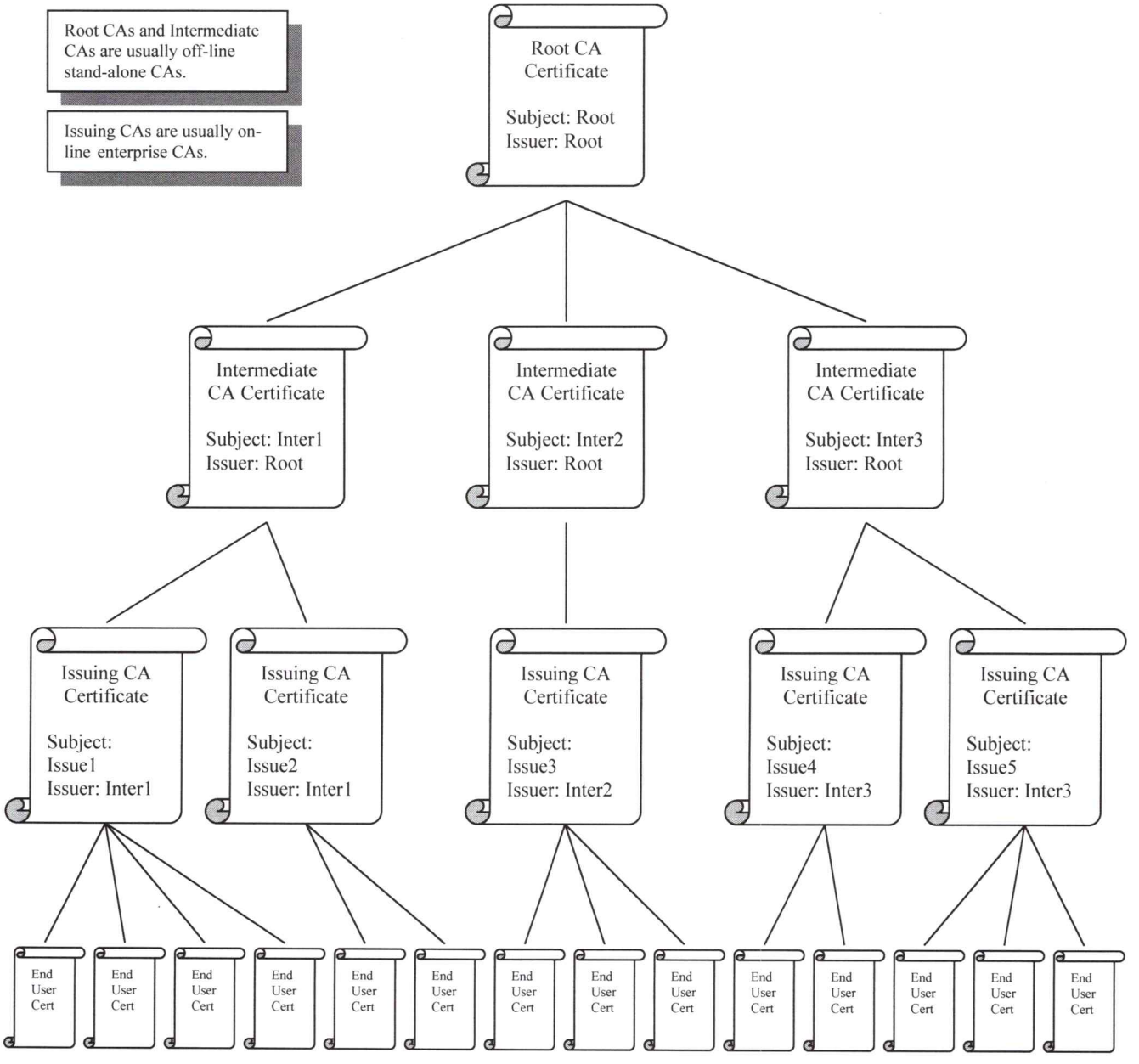
Windows CAs do not follow the web-of-trust model.

Do I Have To Deploy A Three-Tier CA Hierarchy?

No, but there are benefits to having at least a two-tier CA hierarchy. The benefits are discussed below.

A CA used for small-scale enrollments used internally could be just the root CA. No subordinate CAs are needed if the sensitivity of the certificates is low, enrollments are infrequent, and there are no special legal or regulatory requirements.

A two-tier hierarchy makes more sense in a medium-sized organization. A three- or *n*-tier hierarchy might be appropriate in a multi-national corporation or large government agency with numerous departments. The design is flexible and, if you start with a two-tier hierarchy, you can always add more CAs later.



In the diagram above, the root CA at the top signs its own certificate (so, Subject = Issuer). All CA certificates below the root are called "subordinate CAs". The CAs at the bottom, which enroll end users and computers, are called "issuing CAs". All the CAs in between the root and issuing CAs are called "intermediate CAs", and there can be zero, one or many layers of intermediate CAs. The end-user certificates at the bottom are not CA certificates. They are used by people or computers for encryption, digital signatures, key exchange, etc.

What Are The Benefits Of A CA Hierarchy?

The benefits of having at least a two-tier CA hierarchy are the following:

- **Flexible management and security policies**. Not all divisions and departments within an organization will require the exact same PKI security policies. These policies concern how identities are verified, how certificates are requested and published, the protection of private keys, key renewal and revocation guidelines, permitted certificate purposes, delegation of authority over the CA, etc. Each CA in a hierarchy can have a different management or security policy to match the type and sensitivity of the certificates it issues.

- **Damage containment**. If a root CA is compromised, the entire PKI is compromised. If only an intermediate or issuing CA is compromised, then only that branch of the PKI is affected and new certificates can be issued to replace the CA and user certificates involved.

- **Load balancing and fault tolerance**. Multiple issuing CAs provides load-balancing and fault-tolerance for certificate enrollment. Also, a CA's CRL only includes the certificates it has revoked, not all revoked certificates from all CAs; this helps to keep the size of the CRL as small as possible.

- **Delegation of authority**. Each CA can be controlled by a different division or administrator.

Can My Organization Have Multiple Hierarchies?

Yes, though the value of this would have to be carefully planned. The bottom line is: what CAs do my users and computers trust? Users and computers can be made to trust many root CAs simultaneously, and one or more of these root CAs can be controlled by the local organization, and one or more of these root CAs can be controlled by outside organizations if they are deemed trustworthy.

Though it would complicate administration somewhat, there are no in-principle problems with owning multiple CA hierarchies. In fact, there are two cases where having multiple hierarchies can be very advantageous:

- Certificates for partner networks: vendors, suppliers, distributors, etc.

- Certificates for secure e-mail

Certificates for Partner Networks. A partner network may already have a PKI installed. To provide you with access to the partner's network, the partner may issue a subordinate CA certificate to your organization. You will install an issuing CA with the certificate and distribute personal certificates to those users who require access to the partner network. The partner company is trusting you to be a responsible issuer of their certificates, but, at the same time, the CA certificate given to you is strictly limited in its purposes and life-span. On the partner network, your partner-issued certificates will be mapped to user accounts that will be tightly controlled.

Certificates for Secure E-Mail. Certificates for secure e-mail are problematic because the recipients of one's e-mail must trust your CA. Within an organization this is not a problem, but when random people on the Internet receive e-mail from you, they will not trust your organization's CA, hence, they will not be able to verify your digital signatures. The solution is to use e-mail certificates that have been issued by one of the popular commercial CAs whose certificates are already built-into most e-mail programs, e.g., VeriSign's CA certificates are built into Windows and are thus trusted by Microsoft Outlook.

To obtain e-mail certificates from a popular commercial CA, these CAs will sell personal e-mail certificates one certificate at a time. But this is expensive, does not scale, and does not provide one with any administrative or security control over the certificates. Therefore, many of the commercial CAs will issue subordinate CA certificates (subordinate to the commercial CA, that is) to organizations that wish to issue and manage their own e-mail certificates. Your company could install a Windows CA with the subordinate CA certificate, then issue certificates whose trust path leads up to the commercial CA's root, not one's own private root CA.

Some subordinate CA certificate vendors include:

- http://www.wisekey.com
- http://www.verizonbusiness.com
- http://www.chosensecurity.com

CA Hierarchy Design Recommendations: How To Get Started

Start with an off-line stand-alone root CA and use it to issue a subordinate certificate to a single on-line enterprise CA. Try to issue all certificates from the one enterprise CA. This is a starting point and it may be sufficient for small- to medium-sized networks.

For each division or department that requires a significantly different security policy, or which has a large number of users, or which demands independence for political reasons, the root CA should issue another subordinate CA certificate. If the user base is large enough, this should be an intermediate CA certificate for an off-line stand-alone CA, with the intention that it will create issuing CA certificates as necessary underneath it.

Otherwise, the division or department should install just an issuing, on-line enterprise CA from the root without an intervening intermediate CA.

As needed for load-balancing, efficiency and fault tolerance, additional issuing, on-line enterprise CAs can be installed. A single enterprise CA can manage hundreds of thousands of certificates, but keep in mind that certificates are replicated through the Global Catalog, hence, Active Directory replication links must be able to handle the bandwidth, and domain controllers must be able to handle the storage requirements.

For partner networks and secure e-mail, a subordinate CA certificate should be obtained from the partner company or from a popular commercial CA. These certificates will be used to install on-line enterprise CAs which are subordinate to the partner or commercial CA, yet still under one's administrative control.

By starting with a two-tier hierarchy, you can always add more subordinate CAs later to make the hierarchy as deep or wide as you require. And you can always add more root CAs later too, though it's best to try to stick with just a single internal root CA if possible.

How Do I Install Certificate Services?

Certificate Services is installed with PowerShell or from the Server Manager console. We will use Server Manager in this course, but here is how it can be done in PowerShell:

```
Install-WindowsFeature -Name ADCS-Cert-Authority,ADCS-Web-
Enrollment,ADCS-Online-Cert -IncludeManagementTools

Install-AdcsCertificationAuthority -CAType EnterpriseRootCA
-KeyLength 4096 -ValidityPeriod Years -ValidityPeriodUnits 10
-CACommonName Testing-CA -Force

Install-AdcsWebEnrollment -Force

Install-AdcsOnlineResponder -Force
```

What are these PowerShell command-line options? Instead of covering them here, let's discuss them while running the graphical Server Manager in the next lab. If you would rather install Certificate Services using PowerShell instead of Server Manager, the above script is here: C:\SANS\Day3-PKI\Install-CertificateServices.ps1.

Before using Server Manager or PowerShell to install the CA, here are a few of the important concepts. This will help us to understand what the Server Manager wizard is asking and what the above PowerShell parameters are doing.

Cryptographic Service Provider (CSP)

Your Cryptographic Service Provider (CSP) is that module, hardware or software, which actually performs the work encryption, decryption, hashing, etc. If you've installed a Hardware Security Module (HSM) and driver, you'd select the HSM from the pull-down

list of CSPs during installation with Server Manager. If you are not using an HSM and you don't have a special reason to choose another CSP, then choose the "RSA#Microsoft Software Key Storage Provider" CSP (consider this to be the best default).

If you are using a smart card or other special CSP, and that CSP provides the option to require human interaction during cryptographic operations, and you want to require such interaction to maximize security, then check the box to "Allow administrator interaction when the private key is accessed by the CA" during the installation with Server Manager. With a smart card CSP, for example, an administrator would need to enter the PIN for each operation. While such human interaction may be desirable for an off-line root or intermediate CA, it would not be practical for an on-line issuing CA. If in doubt, leave the box unchecked in Server Manager.

CA Cipher and Key Size

Depending on the CSP chosen, you can select an RSA key size up to 16384 bits, but, as a practical matter, never choose a key size larger than 4096 bits. In fact, if you have older Cisco hardware or older Java applications (circa 2001) which use certificates, then choose a key size of 2048 bits, since these older products can't support key sizes larger than 2048 bits. If you have any doubt, and you want to lean towards backwards compatibility at the price of optimal security, then choose 2048 bits. If you want to lean towards security at the price of optimal performance, then choose 4096 bits. But never use an RSA public key smaller than 1024 bits for any purpose whatsoever, whether it's for your CA, servers, workstations or applications.

For the sake of backwards compatibility, it's best to use RSA as the public key cipher for the CA. Perhaps in five to fifteen years it will be safe to choose a faster and more secure cipher, such as an elliptic curve cipher, but it's still too soon. The exception would be if you know for certain that you will only use your PKI internally.

CA Hashing Algorithm

The choice of hashing algorithm is problematic. Never choose MD5. If you need to maximize backwards compatibility, choose SHA-1. But note that SHA-1 has been deprecated and Windows stopped supporting SHA-1 on January 1, 2017 for most certificates!

> **Warning:** SHA-1 has been deprecated and Windows stopped supporting most certificates that use SHA-1 on January1, 2017!

SHA-2 is supported on Windows XP-SP3, Server 2003-SP2, Vista, Server 2008, and later operating systems. If you know the computers of your users, clients and customers meet these minimum OS requirements, then choose one of the SHA-2 hashing algorithms: SHA-224, SHA-256, SHA-384 or SHA-512. SHA-256 will probably become the standard, but there will be problems during the transition from SHA-1. This is going to require careful testing in your environment and there is no way around it!

For more information, please perform these Internet searches in Bing or Google:

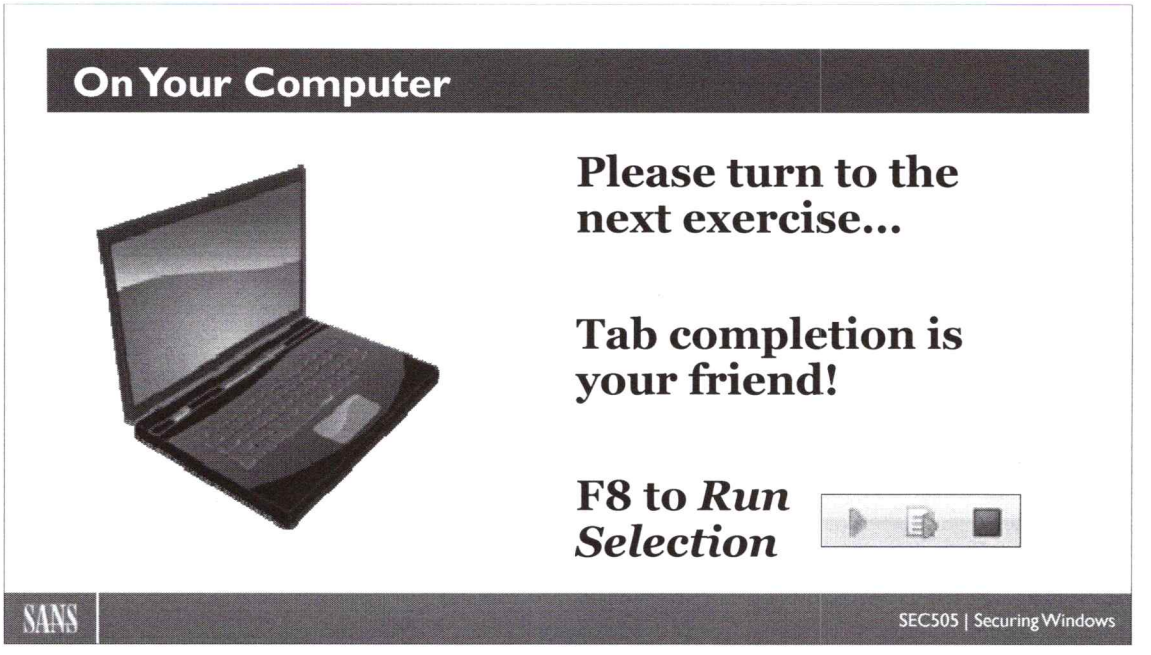
- site:technet.com sha1 deprecation policy
- site:microsoft.com Migrating a Certification Authority from a CSP to a KSP
- KB968730
- KB938397

CA Name

When you choose a name for the CA, keep in mind that this will be the CA's name forever, even after upgrading the OS or moving the server from one geographical region to another. This is a name that will potentially be used for decades. Hence, choose a name which does not identify the Windows version, city, region, responsible IT group, or any other item which may change in the years to come. Also, avoid using space characters anywhere in the name, use dashes instead.

CA Certificate Time to Live

Keep in mind that a CA cannot issue new certificates with TTL's longer than its own, hence, a rule-of-thumb is to enter a TTL which is two or three times the TTL of any certificate that will likely be issued from the CA itself. If in doubt, give the root CA a TTL of 20 years. If the CA is a subordinate, give it a TTL of 10 years or half the lifetime of the CA above it in the PKI hierarchy, i.e., it's issuer. End-entity certificates typically have a lifetime of six months to two years, but some have a TTL as short as a few minutes or hours. If in doubt about end-entity certificate TTLs, use one year.

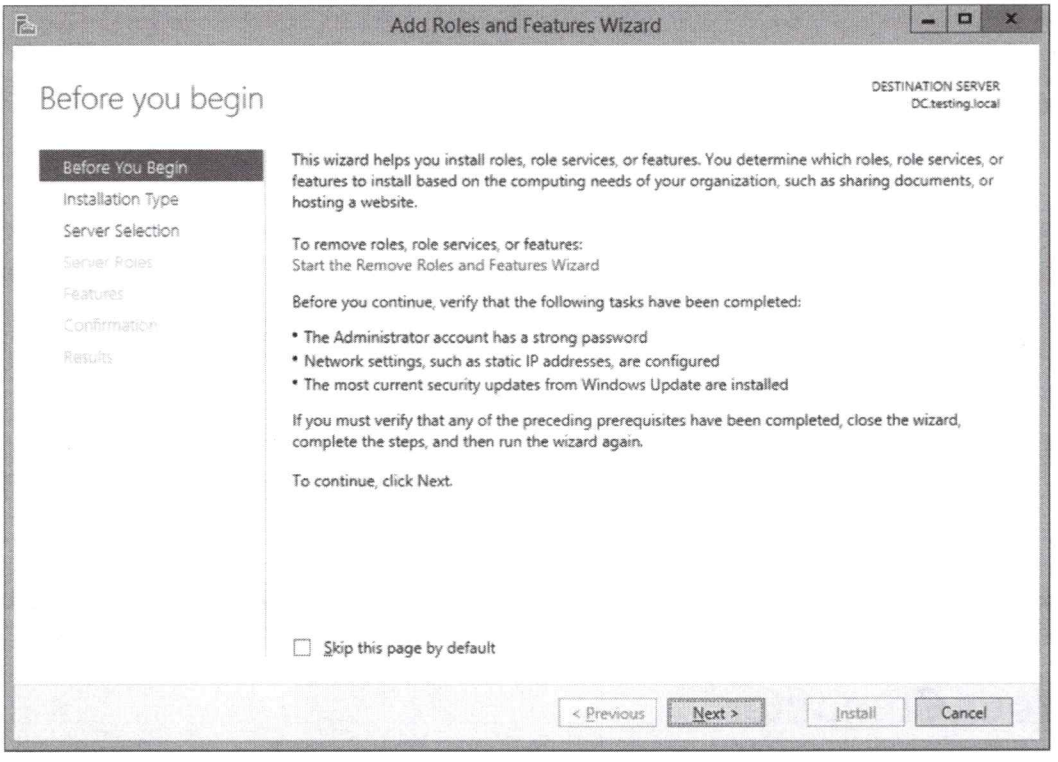


On Your Computer

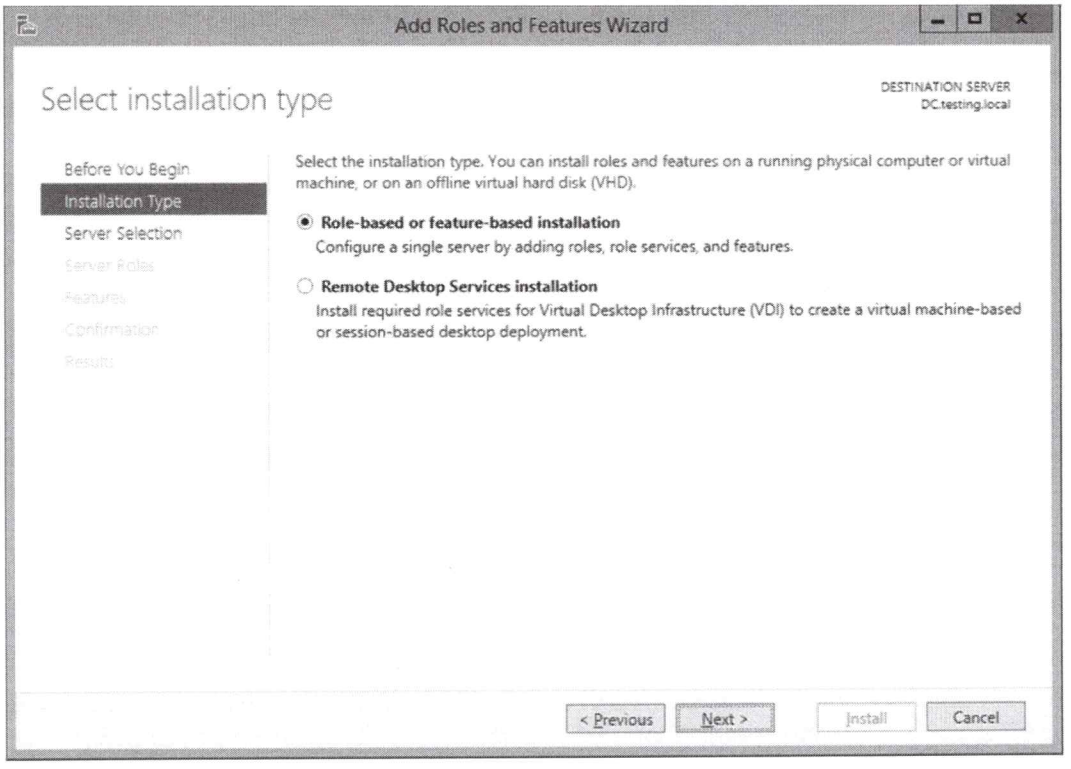
In this lab, we will install Certificate Services using Server Manager.

> **Note:** If you would rather install it using PowerShell instead, please run the following script: C:\SANS\Day3-PKI\Install-CertificateServices.ps1. However, do not run the script and also use Server Manager, it is one or the other, not both.

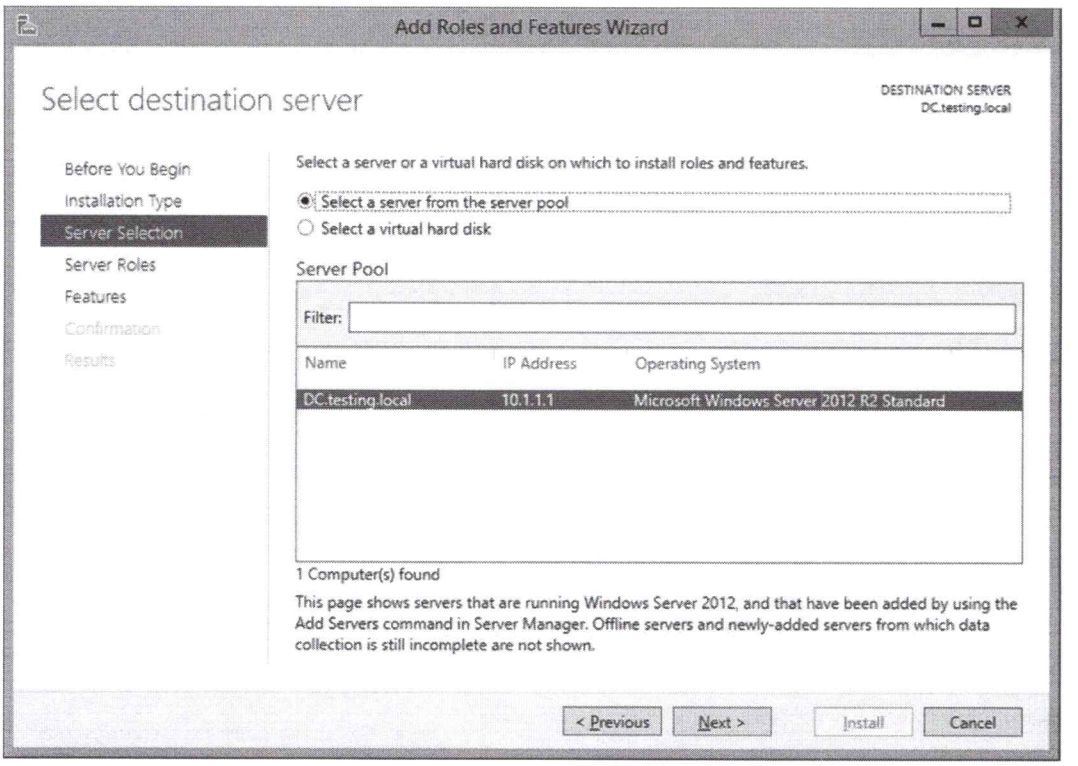
In Server Manager, pull down the Manage menu > Add Roles and Features.



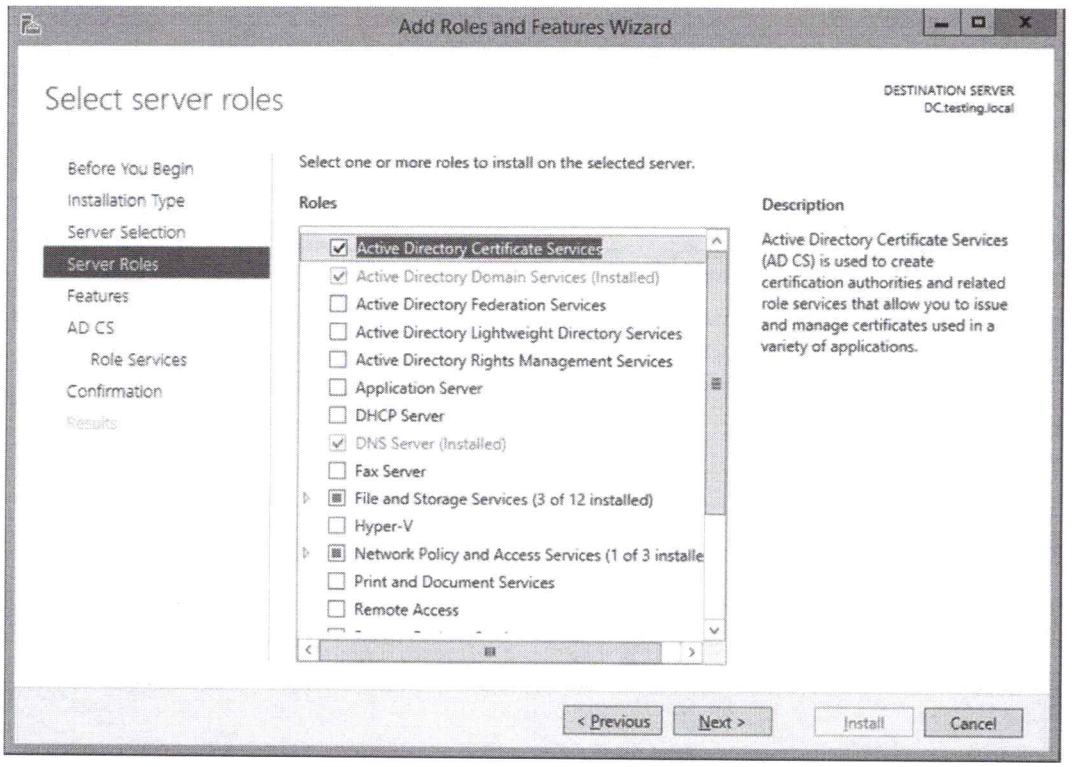
Next > select "Role-based or feature-based installation".



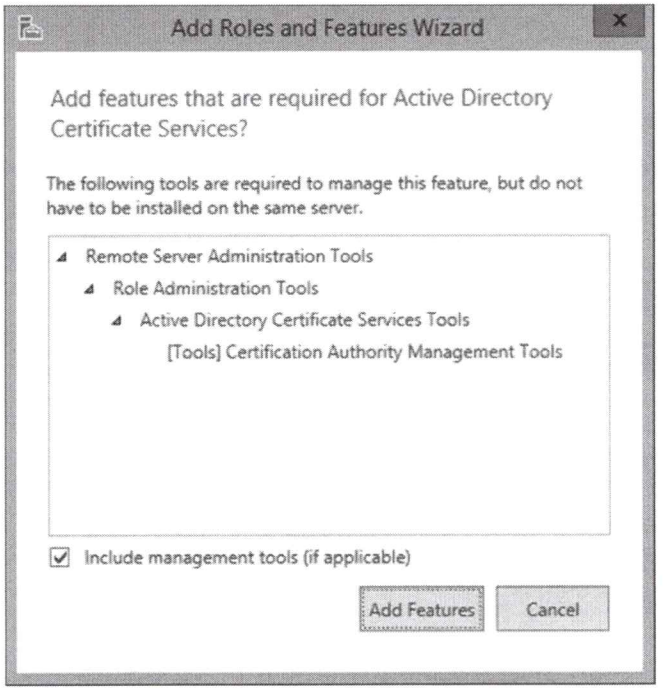
Next > select your local server.



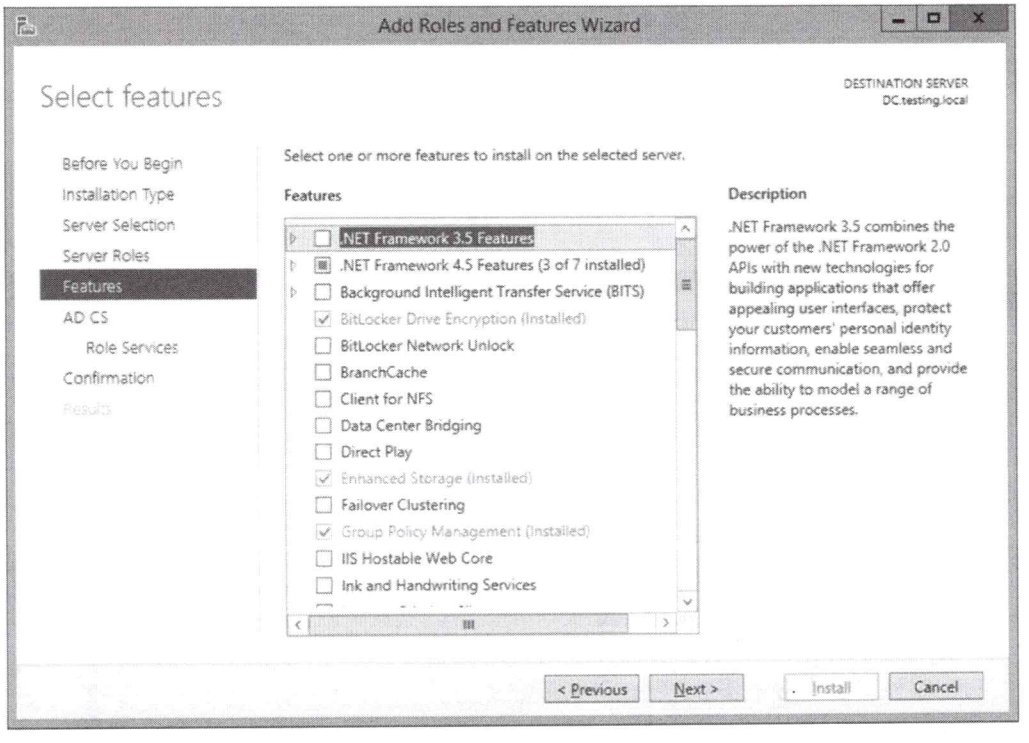
Next > check the box for "Active Directory Certificate Services".



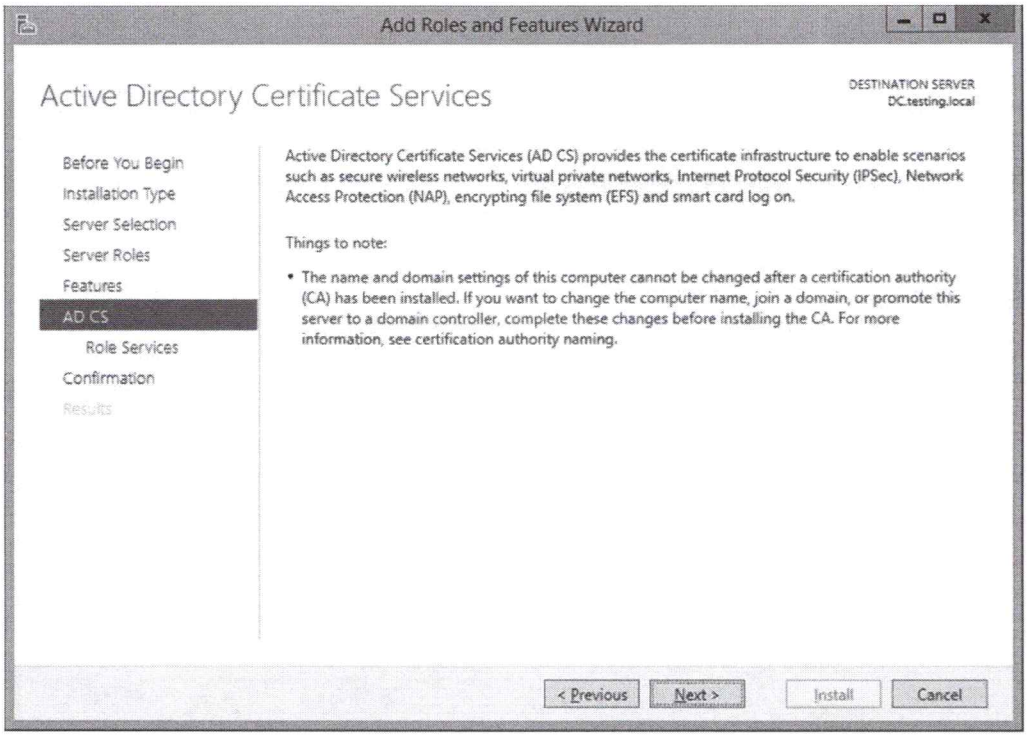
When you check the box, you will be prompted to add additional features. Whenever this dialog box appears, click the "Add Features" button.



Next > Next again, since there are no additional roles/features to add now.



Next again.

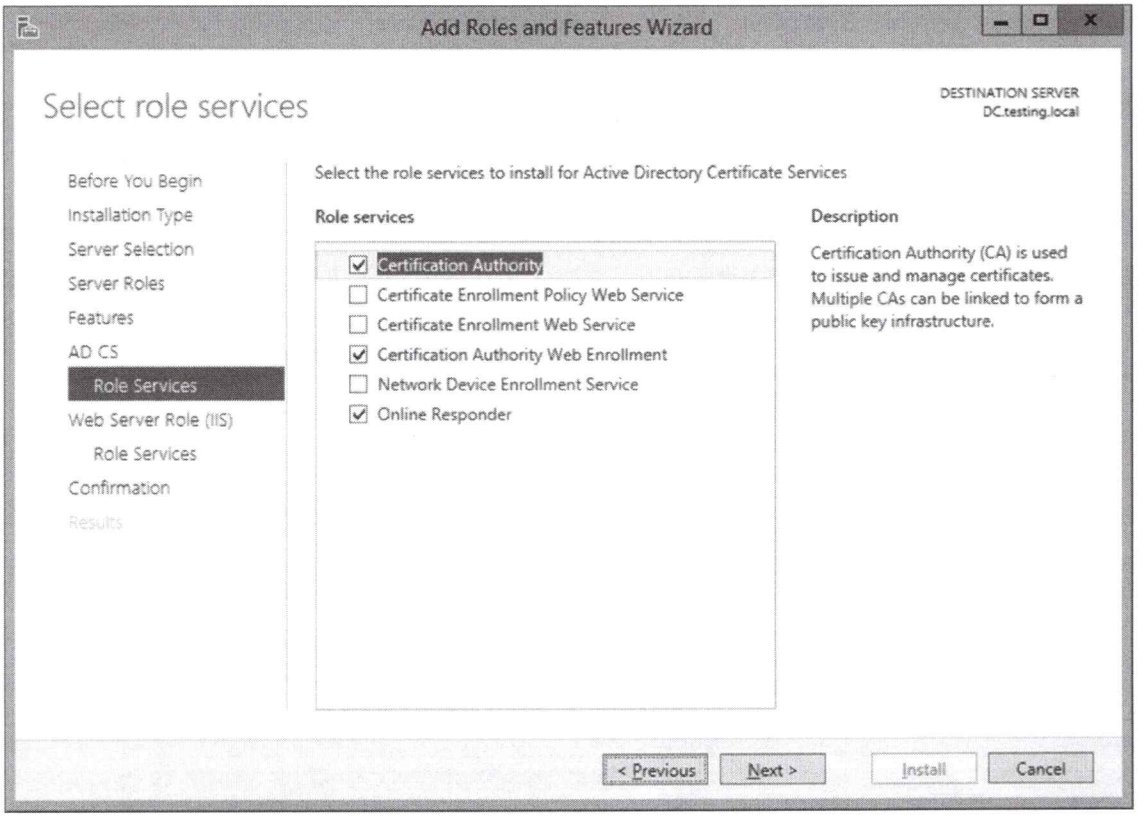


Next again > only check the boxes for:

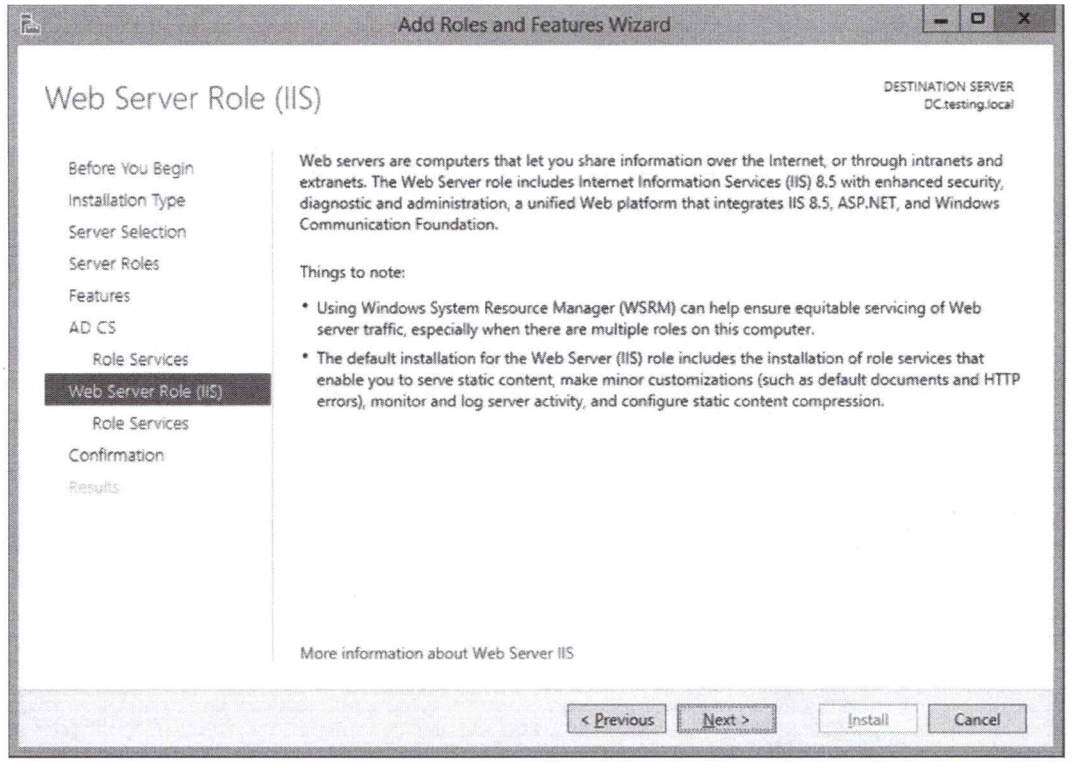
- Certification Authority
- Certification Authority Web Enrollment (not "Web Service")
- Online Responder

Click the "Add Features" button in any pop-up dialog boxes.

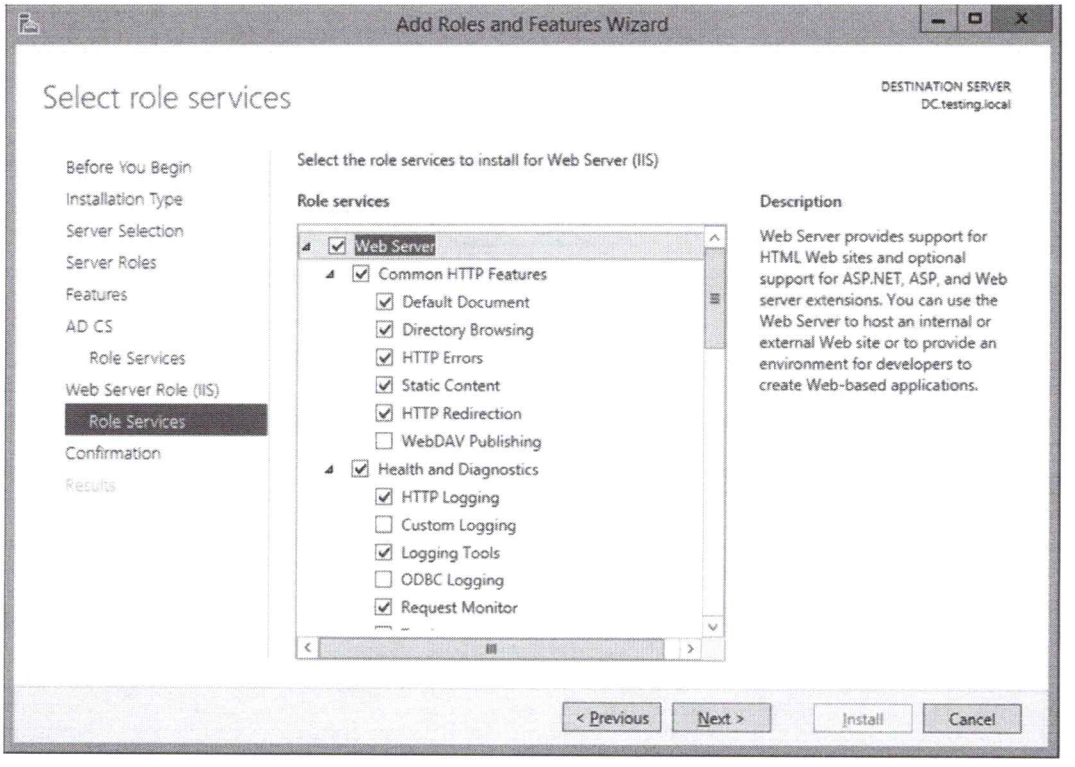
> **Note:** The Web Enrollment component permits interaction with the CA via an IIS web application at https://*servername*/certsrv/. The Online Responder component is for Online Certificate Status Protocol (OCSP) support. The network device enrollment component is for Cisco Simple Certificate Enrollment Protocol (SCEP) support. The other components we cannot discuss right now.
>
> **Important:** If you are prompted to provide the path to the installation media, and if you have mounted the DVD or ISO file on drive letter "D:", then click the link at the bottom to provide an alternate path of "**d:\sources\sxs**".



Next again.



Next again, accepting the defaults for the Web Server role.



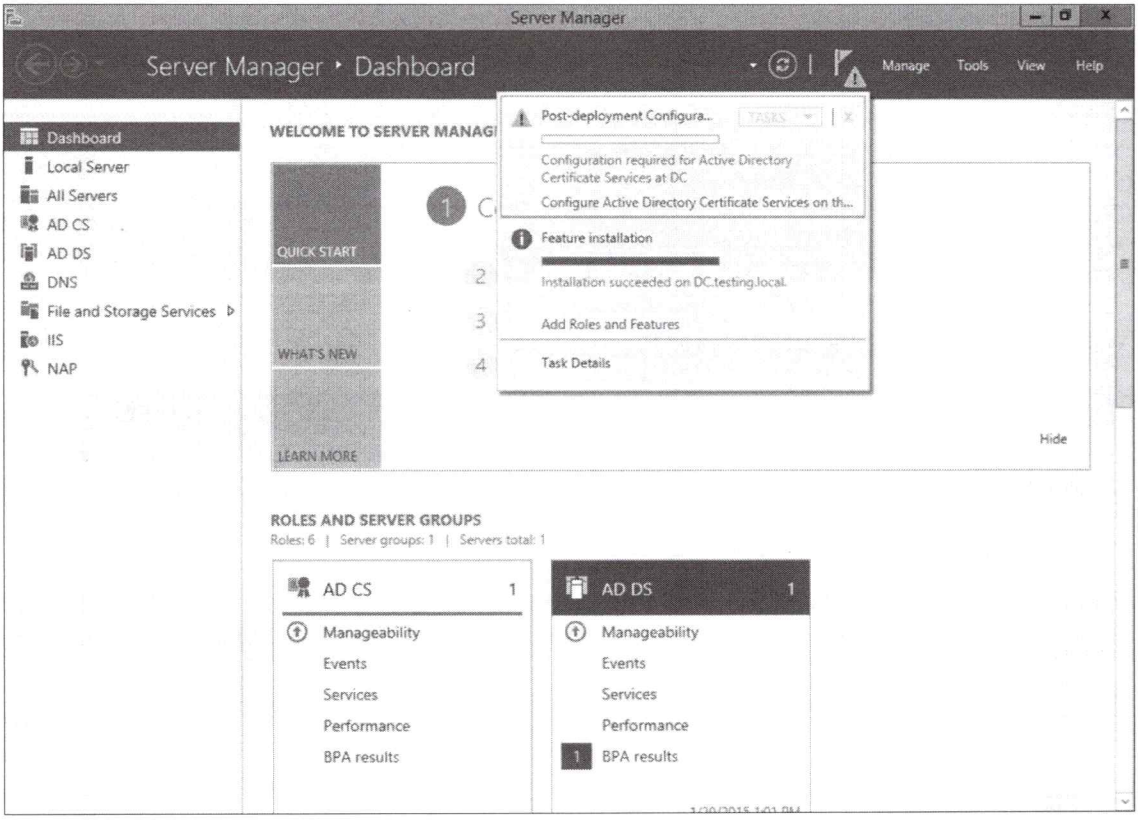
Next > click the Install button.

Click the Close button to return to Server Manager even though the CA is still installing.

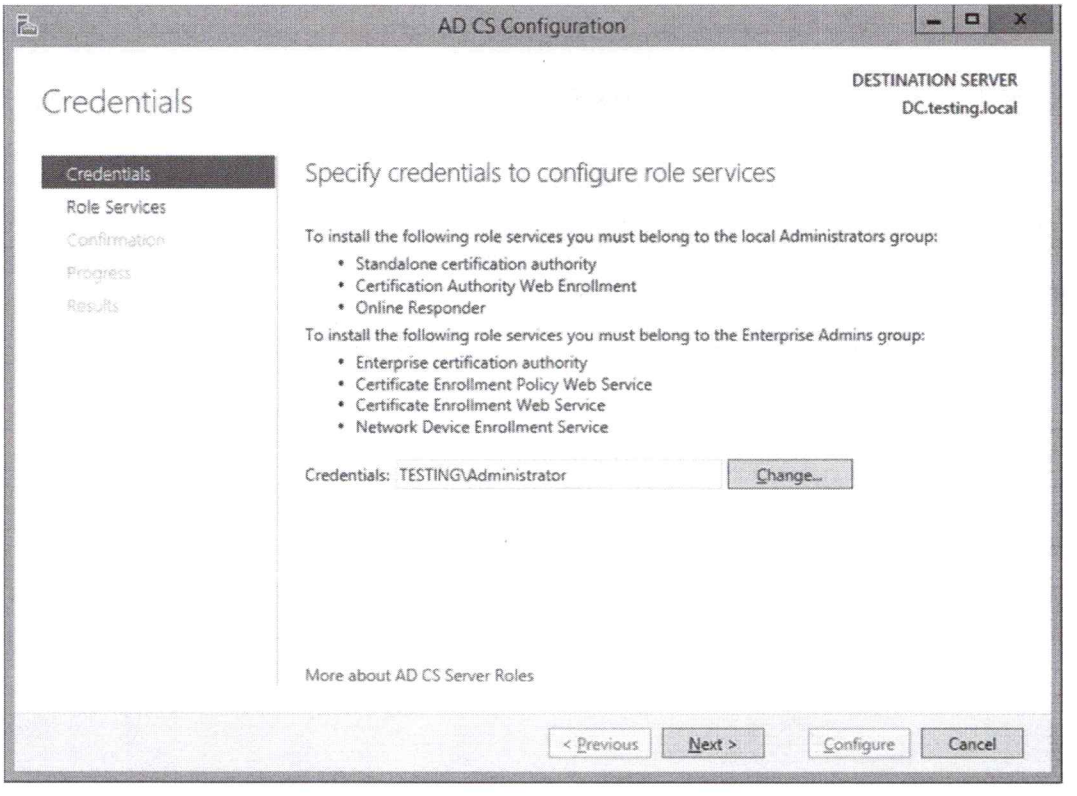
Wait 3-5 minutes in Server Manager. How do we know it's done installing?

In Server Manager, click the circular refresh button in the top toolbar, then click the triangular pennant button to the right of it. This will pull down a history of events and show a moving progress bar for each event. After a few minutes, there will be a yellow triangle next to the pennant button.

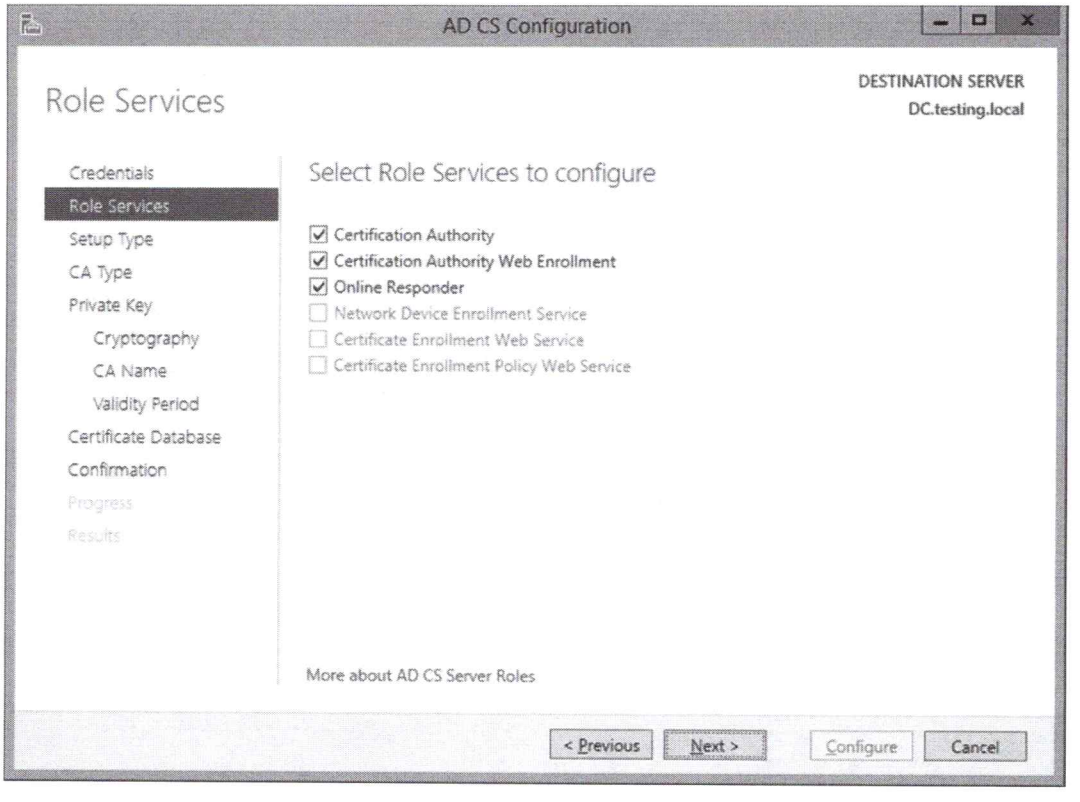
When you see the event with a hyperlink that says "Configure Active Directory Certificate Services", click on that hyperlink to launch a configuration wizard (see the next screenshot for the hyperlink in the pull-down menu).



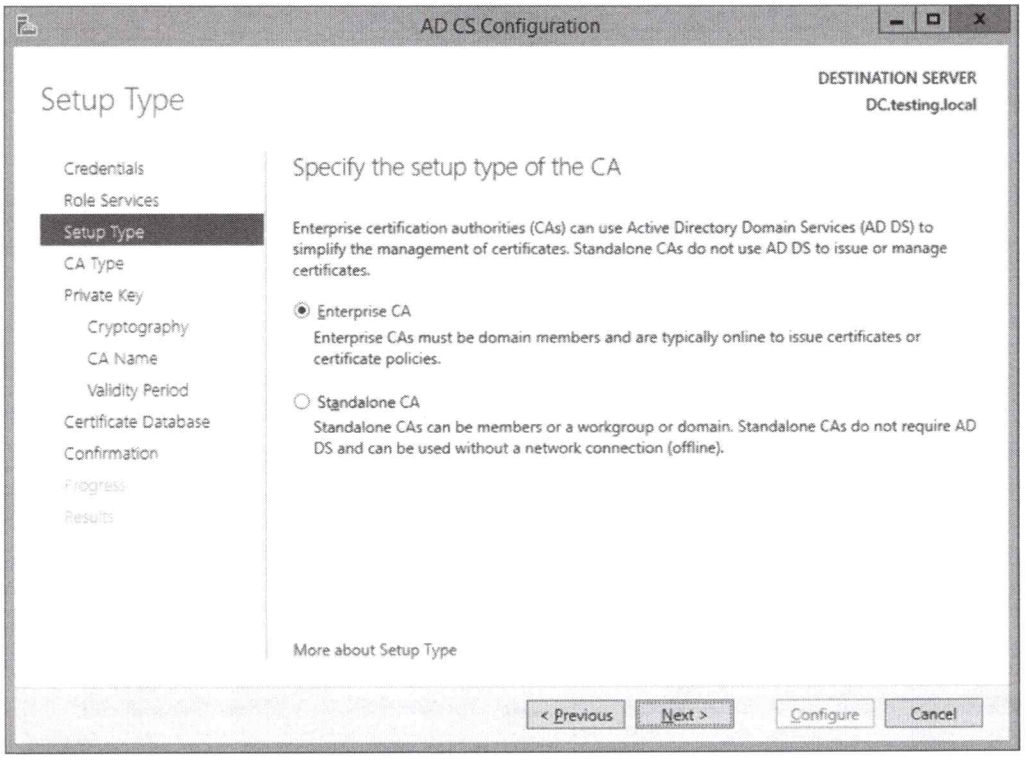
Next again.



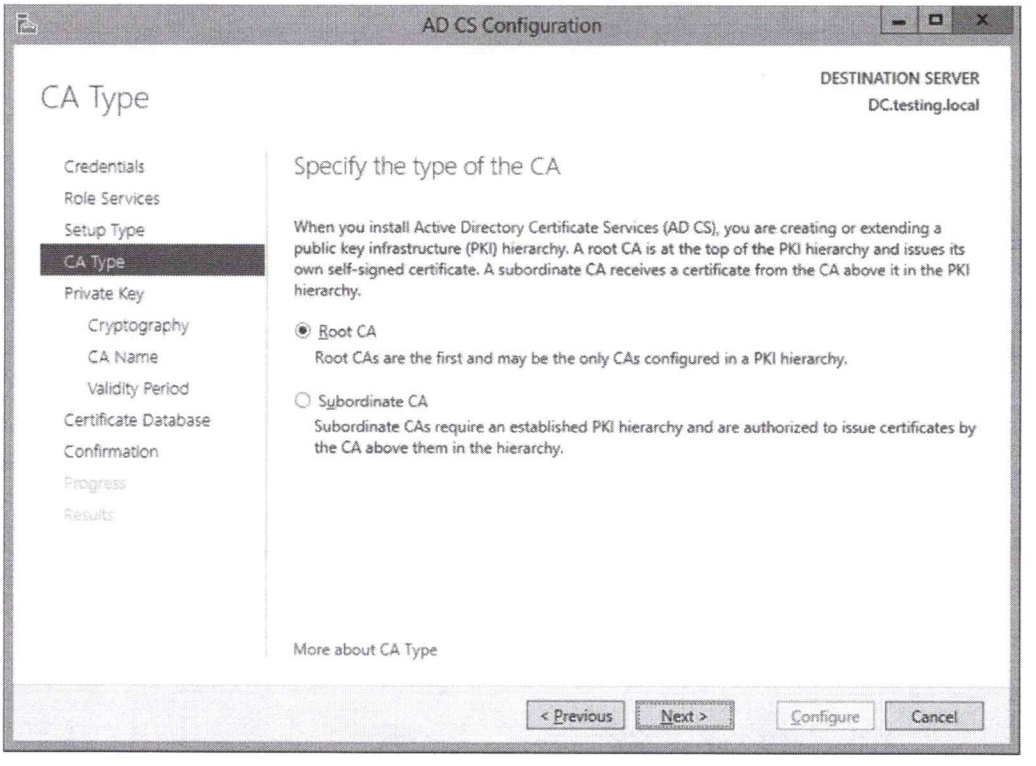
Check the boxes shown below to configure role services > Next.



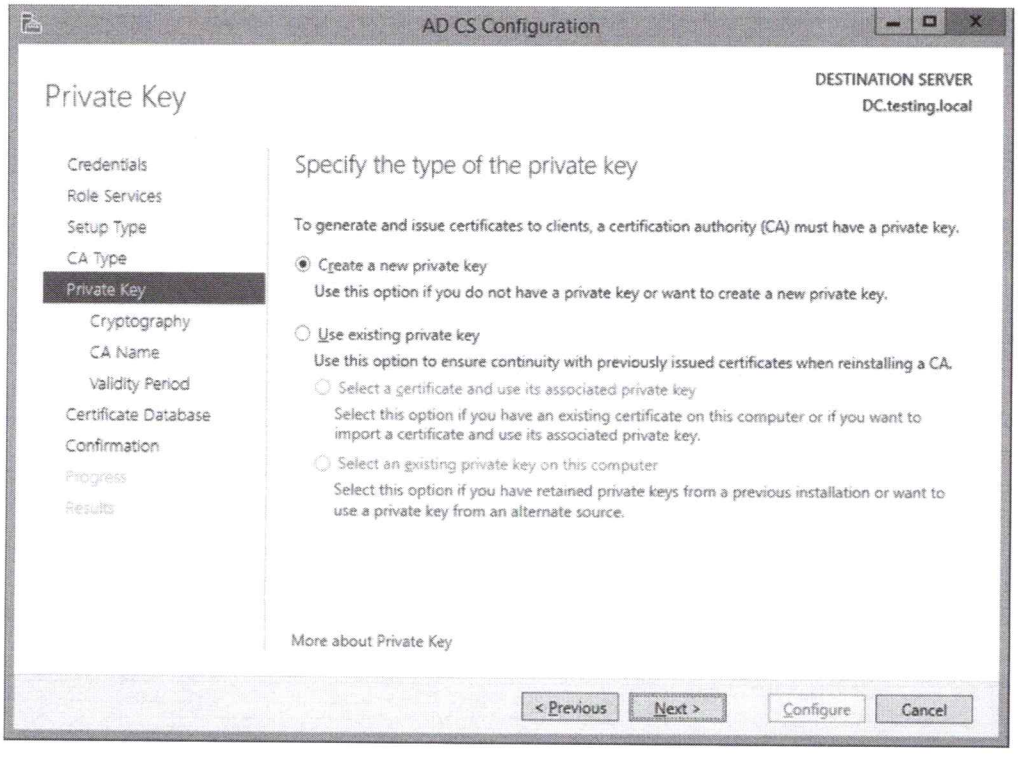
Choose Enterprise CA > Next.



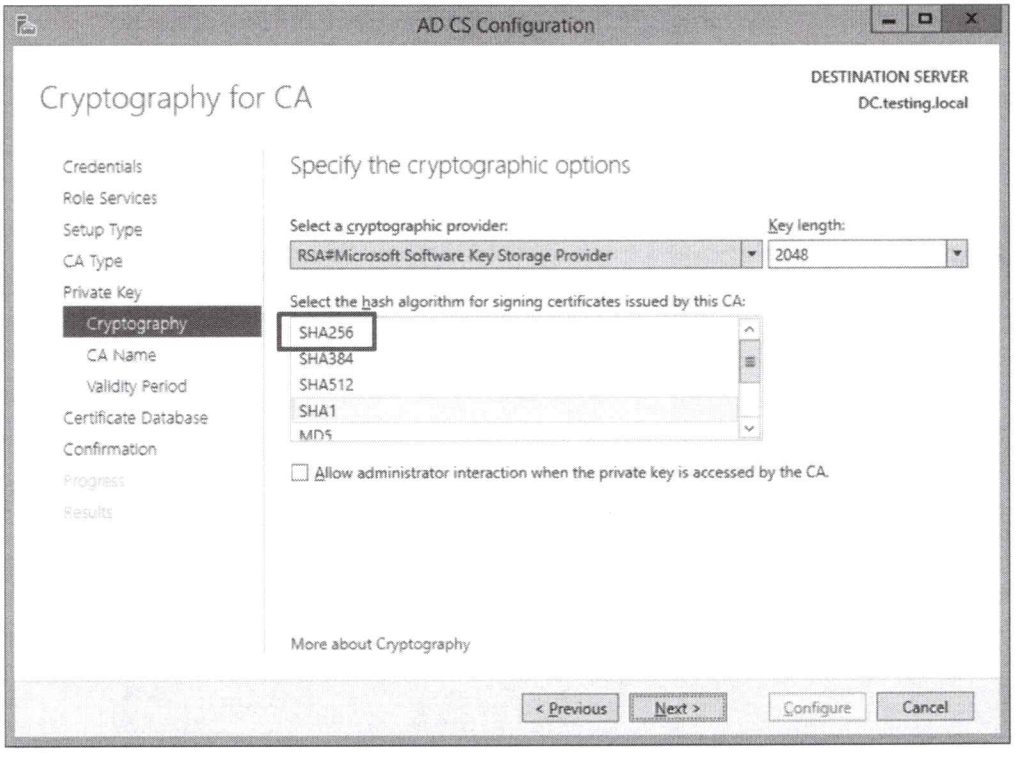
Choose Root CA > Next.



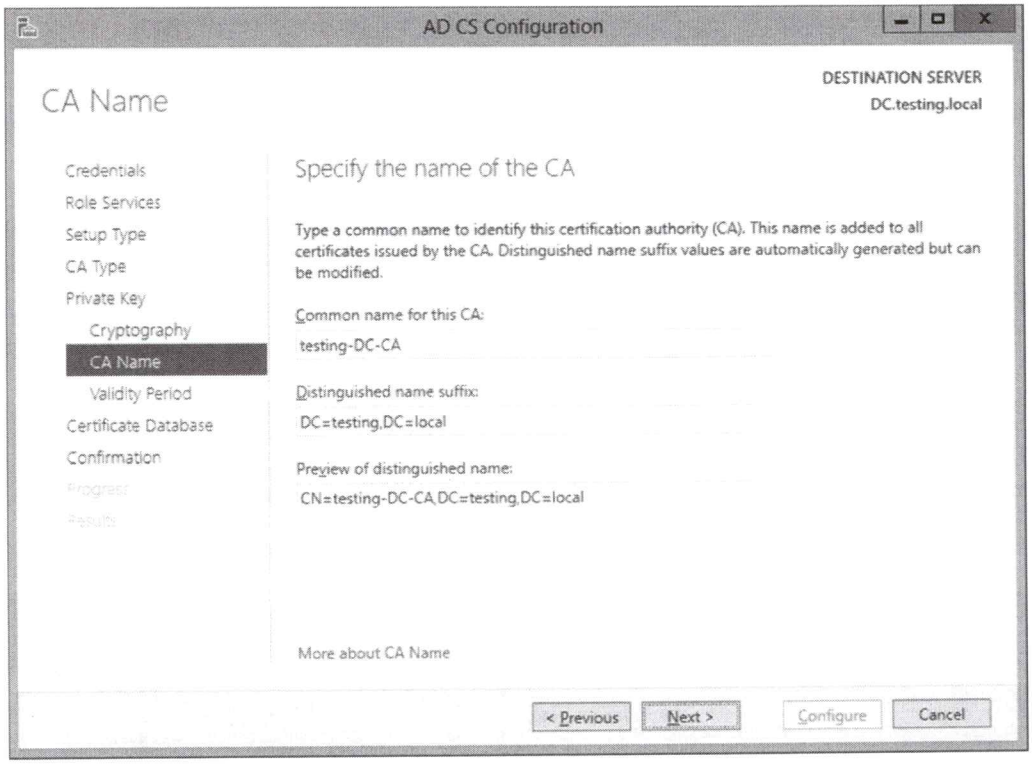
Choose "Create a new private key" > Next.



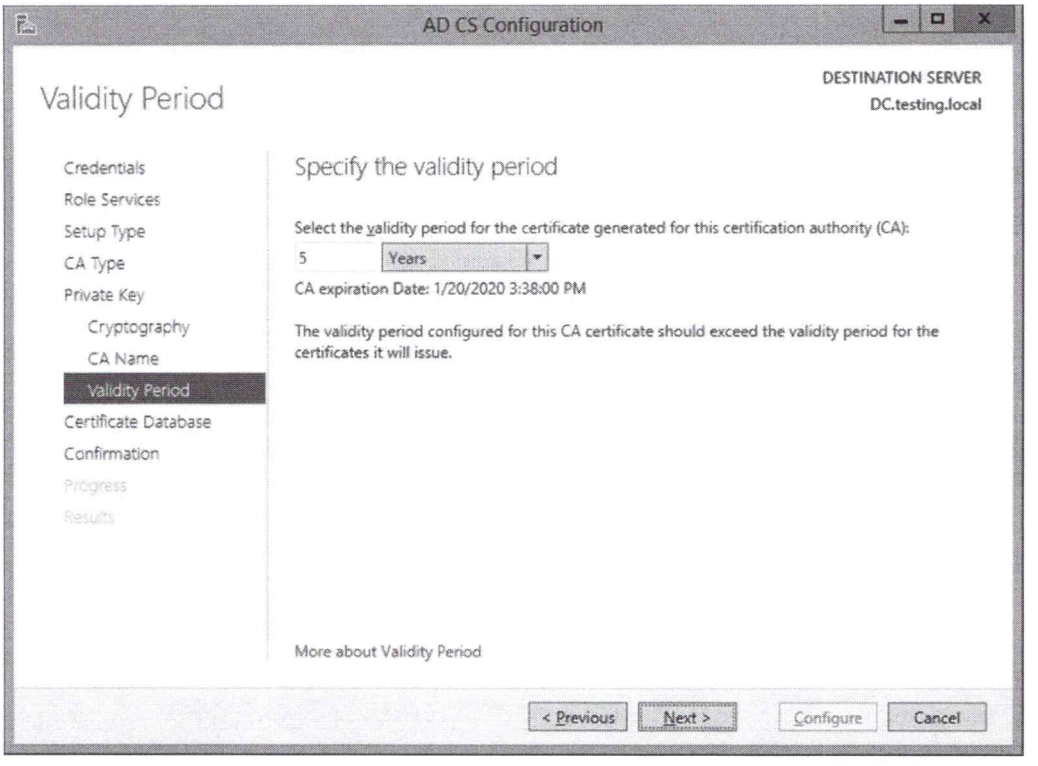
Accept the default cryptographic options (RSA, 2048 key, SHA256) > Next.



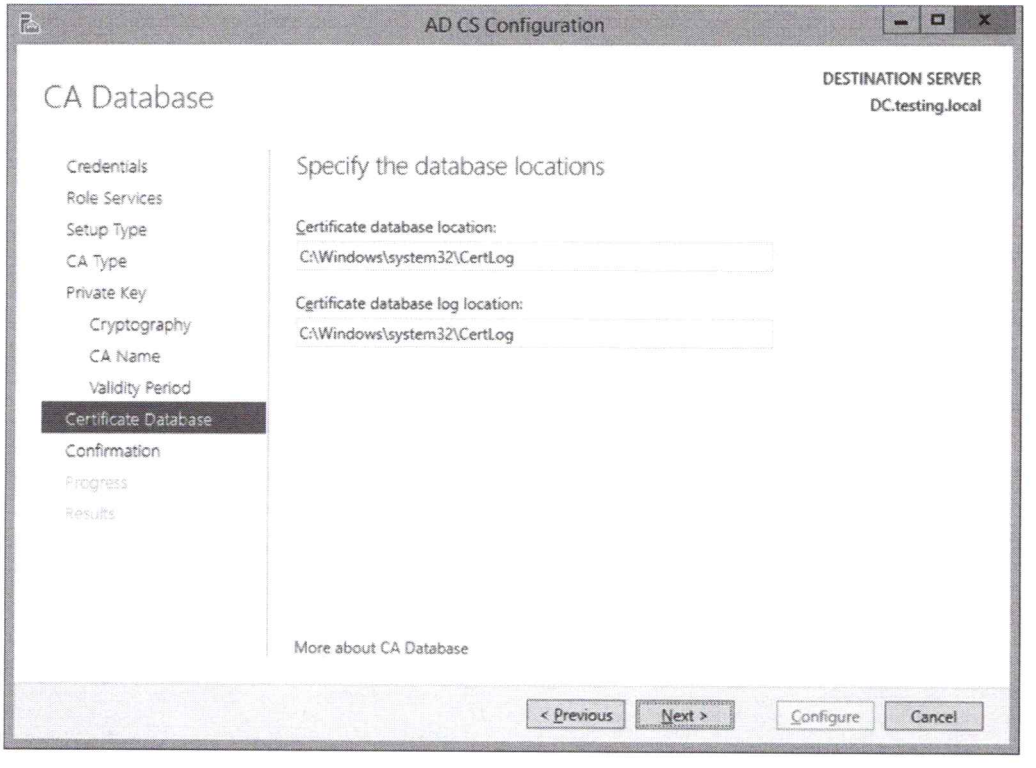
Accept the default name for the CA > Next.



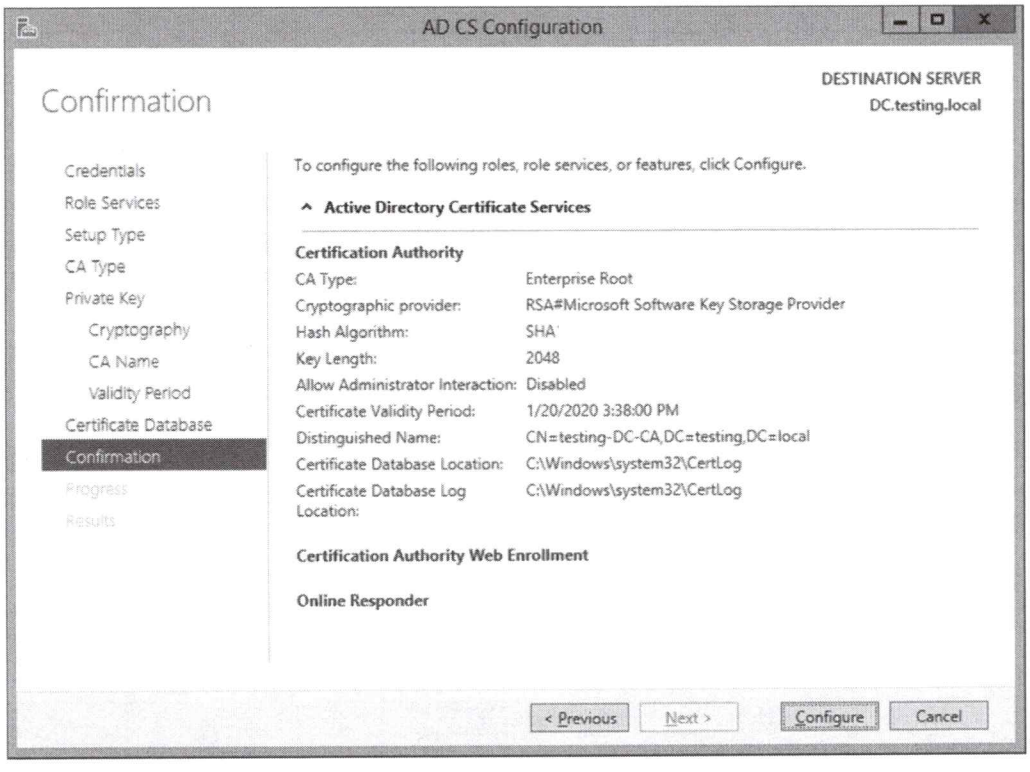
Accept the default validity period > Next.



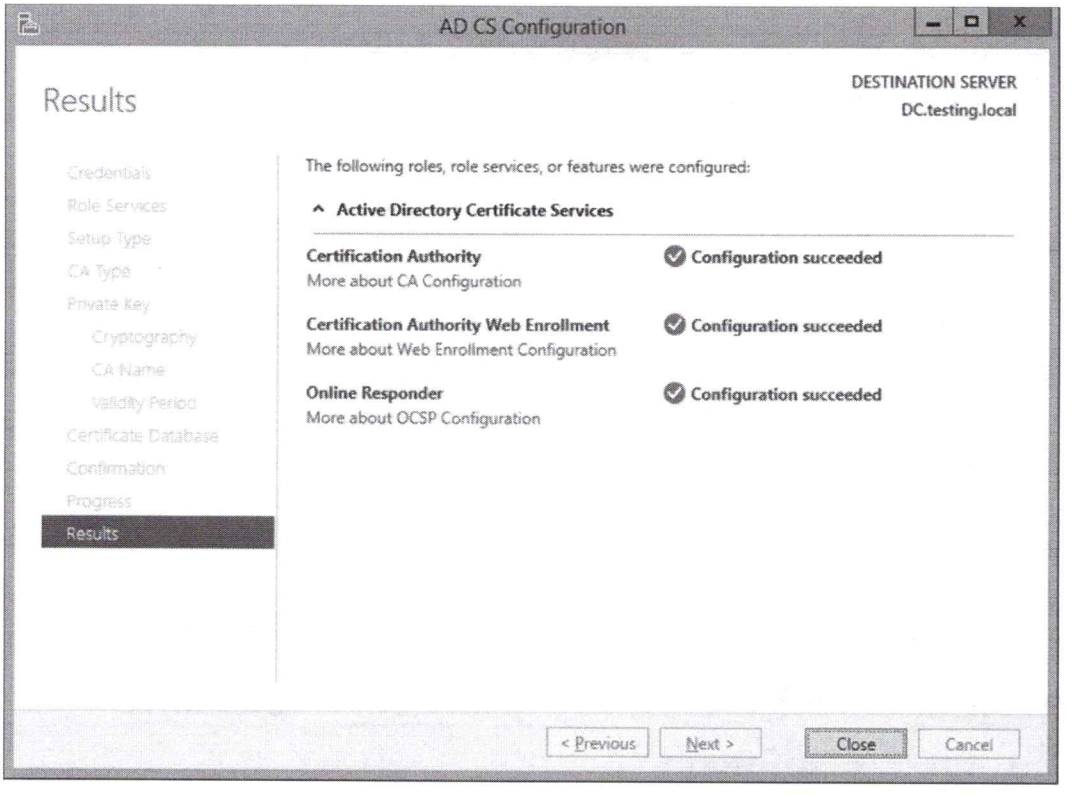
Accept the default database location > Next.



Click the Configure button.



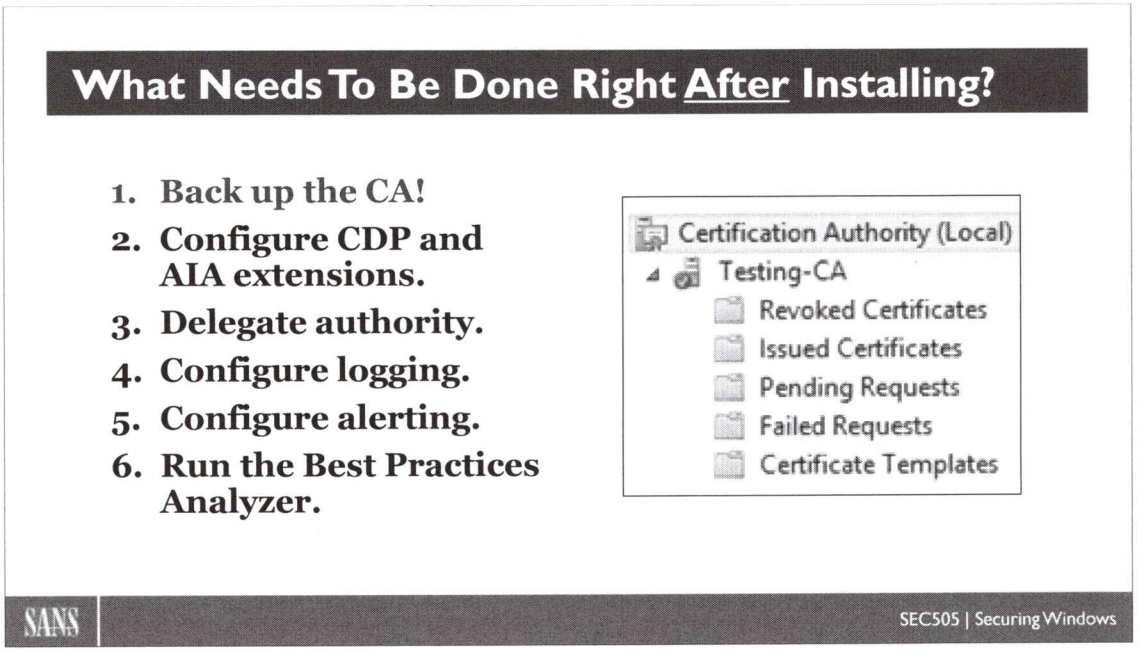
Click Close. You're done, the CA is installed!



In Server Manager, pull down the Tools menu and open "Certification Authority".

If you would like to create your own custom MMC.EXE console, you can now add the following snap-in tools (File menu > Add/Remove Snap-In):

- Certificates (My User Account)
- Certificates (Computer Account)
- Certification Authority
- Certificate Templates
- Online Responder Management



What Needs To Be Done Right After Installing Certificate Services?

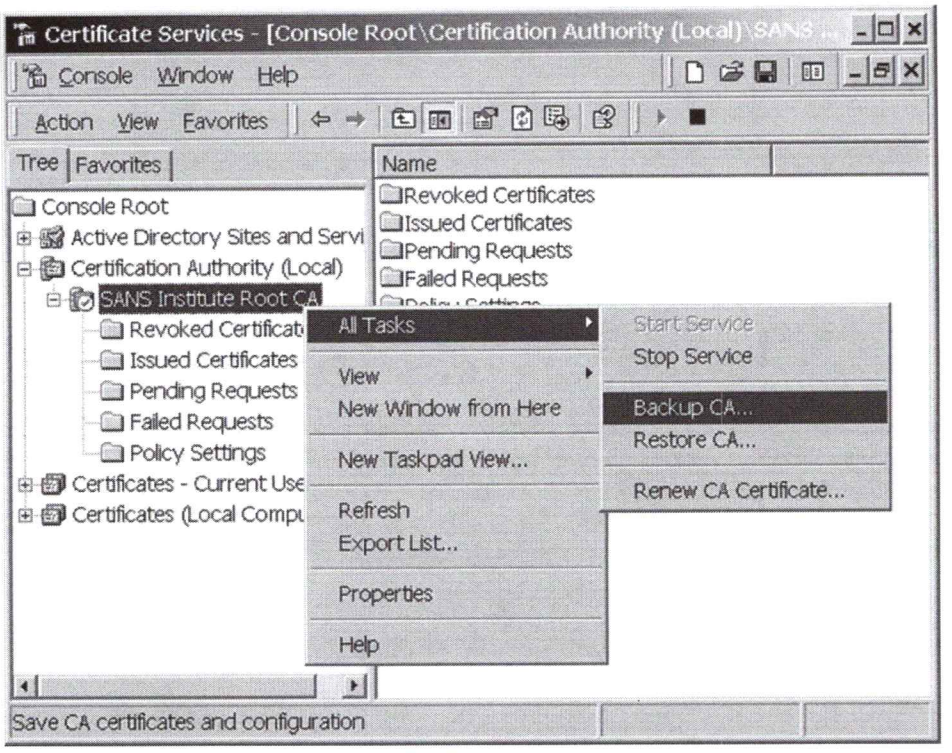
Immediately after installing Certificate Services, and before you begin to issue certificates, you should do the following:

1. Back up the CA certificate, private key, database, and IIS metabase.
2. Configure the CA's policy and exit modules.
3. Edit certificate templates as necessary.
4. Control who can enroll for which certificate types.
5. Delegate authority over the CA.

Backup The CA Database, Private Key and Configuration

You should immediately back up the CA's entire file system and export the CA's public/private key pair. Windows backups can be performed while the server is running and continuing to perform certificate-related operations, just as Exchange Server can be backed up without loss while it is processing e-mail. Back up both the entire file system and the System State, or, even better, make a full binary disk image backup. (Note that a System State-only backup will not include the private keys on Server 2008 and later.)

The CA's database and log can be backed up with the Certification Authority snap-in as well, even while Certificate Services is running and managing certificates.



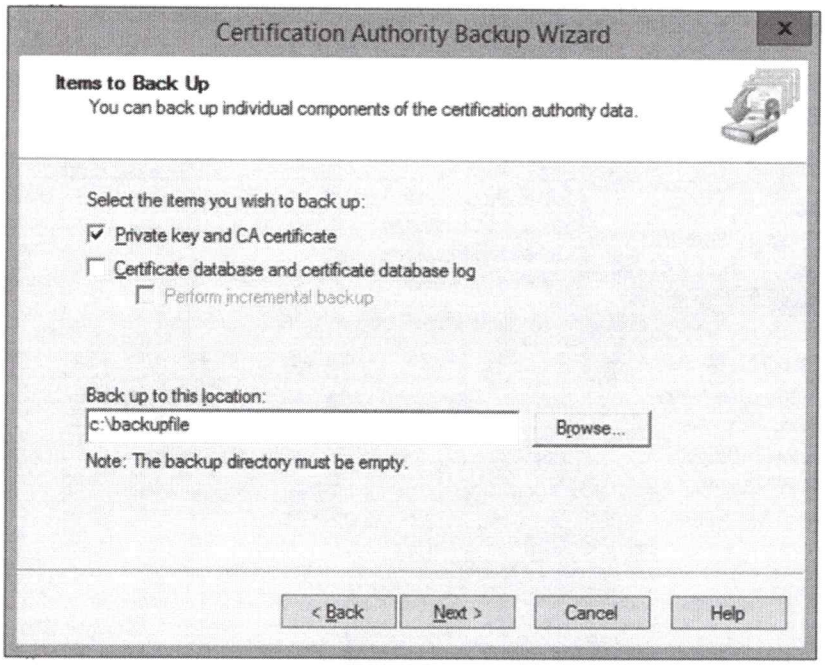
Try It Now!

To back up the Certificate Services database and log, open the Certification Authority snap-in > right-click on a CA > All Tasks > Backup CA > Next > check the box "Certificate database and certificate database log". You also have the option of doing an "Incremental backup", which is actually just a backup of the log files. Select an empty directory where the database and log file copies will be stored > Next > Finish.

The CA's certificate and private key can be exported to an encrypted PKCS#12 file (.pfx). During export you will be prompted for a password, which will be hashed and used as a key to encrypt the data. Use a long and complex password up to 32 characters in length, and save the PKCS#12 directly to removable storage media, like a USB flash drive. Put the storage media in a safe. Saving to the hard drive may leave digital trace images of the file on the drive's surface, in the Recycle Bin, in the swapfile, etc.

Try It Now!

To back up the CA's certificate and private key, open the Certification Authority snap-in > right-click on a CA > All Tasks > Backup CA > Next > check the box "Private Key and CA Certificate" > enter the path to the flash drive > Next > enter your password > Next > Finish. Label the flash disk, store it in a safe or lockbox, write down the password and store the password in a different secure location.



On Server 2012 R2 and later, the backup can be done using the Backup-CaRoleService cmdlet in PowerShell too. There is a corresponding Restore-CaRoleService.

```
$pw = ConvertTo-SecureString -String "MyPasfraiz" -AsPlainText

Backup-CaRoleService -KeyOnly -Path C:\Temp -Password $pw
```

Finally, export and save the CA's configuration settings from the registry. You may wish to repeat this again after the CA has been fully configured. You can use regedit.exe or reg.exe to export HKLM\SYSTEM\CurrentControlSet\Services\CertSvc to a file.

```
reg.exe export HKLM\SYSTEM\CurrentControlSet\Services\CertSvc
 exportedfile.reg /y
```

Configure CDP and AIA Extensions

The policy module on a stand-alone CA can be configured to either hold all certificate requests pending until an administrator approves the request (the default) or simply issue requested certificates automatically (not recommended). Enterprise CAs typically issue certificates automatically; they rely upon successful authentication to the domain, and other features, to control certificate enrollment.

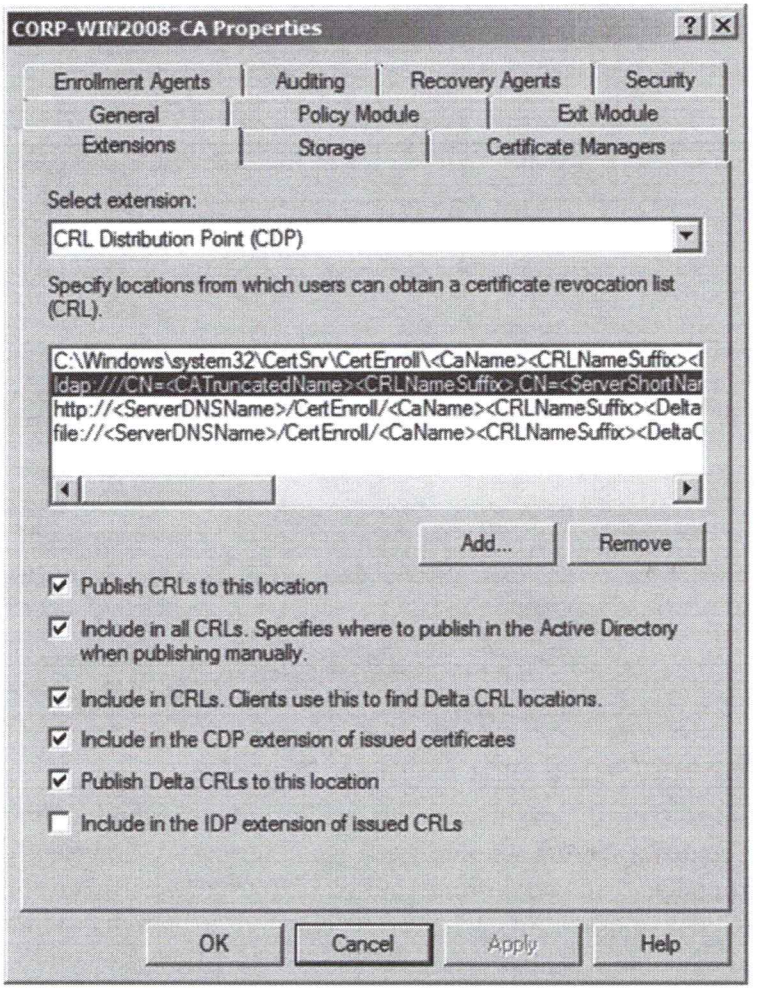
The policy module can also be used to change the default CRL Distribution Point (CDP) and Authority Information Access (AIA) information embedded in every certificate the CA will issue.

Important! If the CA is a stand-alone CA, and you wish to publish its certificate and CRLs to Active Directory, you must edit the AIA and CDP fields before issuing any certificates. Later, you will have to manually publish these items to Active Directory with CERTUTIL.EXE, but only for stand-alone CAs.

The AIA field is a list of URLs where the CA's certificate can be located. The URLs can be of type HTTP, FTP, LDAP or FILE. The CA's certificate needs to be obtained by others when they check the validity of the CA's signature on the certificates it issues. The CDP field is a list of URLs where the CA's Certificate Revocation List (CRL) can be obtained. Certificates are revoked when the CA no longer wants anyone to trust the certificate, perhaps because the certificate's private key has been compromised (see below in the sections labeled "How Do I Revoke Certificates?" and "OCSP").

Try It Now!

To configure a CA's policy module, open the Certification Authority snap-in > right-click the CA > Properties > Policy Module tab > Properties button. To configure the AIA and CDP fields written by the policy module into every certificate issued, go to the Extensions tab.



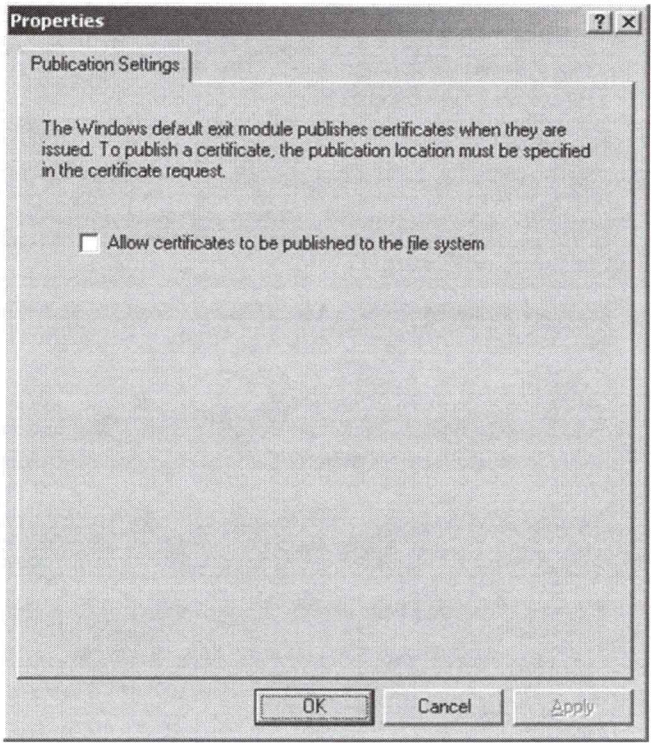
In the section on OCSP we will discuss how OCSP responders sign their responses, but there is an issue that may occur when the certificate of the CA itself is renewed (we cannot discuss it right now). To fix this issue related to OCSP and the renewal of the CA's own certificate, run the following command after installing the CA:

```
certutil.exe -setreg ca\UseDefinedCACertInRequest 1
```

The exit module is configured to determine where issued certificates are "published", i.e., where copies of issued certificates should be saved. A certificate request will name one or more locations where the issued certificate should be published; the exit policy only controls which requested locations are permitted. Enterprise CA's only publish to Active Directory by default, but writing certificates to file system can be permitted. Stand-alone CA's can only publish to the file system.

Try It Now!

To configure a CA's exit module, open the Certification Authority snap-in > right-click the CA > Properties > Exit Module tab > Properties button.



Cert Publishers Group In A Multi-Domain Forest

There is a built-in security group named "Cert Publishers" which contains the computer accounts of Windows servers running Certificate Services. Importantly, the Cert Publishers group has read and write permissions to users' certificates in the domain of the CA (the userCertificate field of each user account). It is by being a member of this group that enterprise CAs have the permission to publish certificates.

Note: No non-CA computer account or user account should be a member of the Cert Publishers group unless you have an application that requires this.

If you desire to issue certificates to users in one domain from a CA in a different domain, then you must give the remote CA read and write permission to the userCertificate field of every user account in the local domain. In a Windows Server 2003 or later forest, the Cert Publishers group is a domain local group, hence, just add the computer accounts of the other CAs in the other domains to one's own domain local Cert Publishers group, and vice versa. If you have very old controllers, see KB219059 and KB300532 for more information.

Common Criteria Role Separation

The Common Criteria for Information Technology Security Evaluation (CCITSE) is an internationally-recognized set of standards for evaluating security products (ISO/IEC 15408). CCITSE requires a certain separation of roles and duties for CA administrators and managers. Only Windows Server Enterprise Edition and later supports role separation, not Standard Edition (and not Windows 2000 Server).

The CA roles supported on Windows (and the permission or user right to which it corresponds in parentheses) are as follows:

- CA Administrator (Manage CA permission)
- Certificate Manager (Issue and Manage Certificates permission)
- Auditor (Manage Auditing and Security Log user right)
- Backup Operator (Backup/Restore Files and Directories user rights)

No single user account can have more than two roles, including any members of the local Administrators group, who will have both the Auditor and Backup Operators roles. If a user has the permissions or rights of more than one role when CCITSE role separation feature is enabled, that user will *not* be able to perform *any* CA management operations whatsoever until the permissions/rights are changed. This includes the local Administrator account.

To enable role separation:

```
certutil.exe -setreg CA\RoleSeparationEnabled 1
```

To disable role separation:

```
certutil.exe -delreg CA\RoleSeparationEnabled
```

To display your current state:

```
certutil.exe -getreg CA\RoleSeparationEnabled
```

Remember that you must regulate access to the private key of the CA very tightly. By default, the local Administrator account, local Administrators, Domain Admins, Enterprise Admins, and Backup Operators can all get access to this key. Do you trust every single one of them? If not, consider investing in a hardware security module which can generate and store the private key more securely. If nothing else, use a smart card.

> **Warning!** Anyone who is a member of the local Administrators group on the CA can extract the CA's private key if you are not using a HSM designed to stop this!

To read more about role separation, search Microsoft's website for the keyword "RoleSeparationEnabled".

Configure Enhanced Auditing

A Windows Server 2003 and later CA can audit the following events and write information to the security Event Log, provided that the "Audit Object Access" audit policy has been enabled on the CA:

- Back up and restore the CA database
- Change CA configuration
- Change CA security settings
- Issue and manage certificate requests
- Revoke certificates and publish CRLs
- Store and retrieve archived keys
- Start and stop Certificate Services

To enable this auditing, go the Properties of your CA > Auditing tab > check the boxes for the information you want to log. The recommendation is to check them all.

Configure Alerting (CAMONITOR.VBS)

PKI can be complex to manage. Part of the difficulty stems from complicated inter-connected protocols and the fact that certificates and CRLs periodically expire. Now is the time to get ahead of the game and configure automated monitoring and e-mail alerts when issues are detected. This can be done with a scheduled VBScript downloadable from http://www.microsoft.com/technet/scriptcenter/solutions/camon.mspx. The name of the script is "camonitor.vbs" and it can write to the event logs on the CA and send e-mail alerts too. The script runs on the CA itself and requires CAPICOM 2.0 or later to be installed. The CAPICOM.DLL is a free download from Microsoft too (to find it, Google on "site:microsoft.com capicom download").

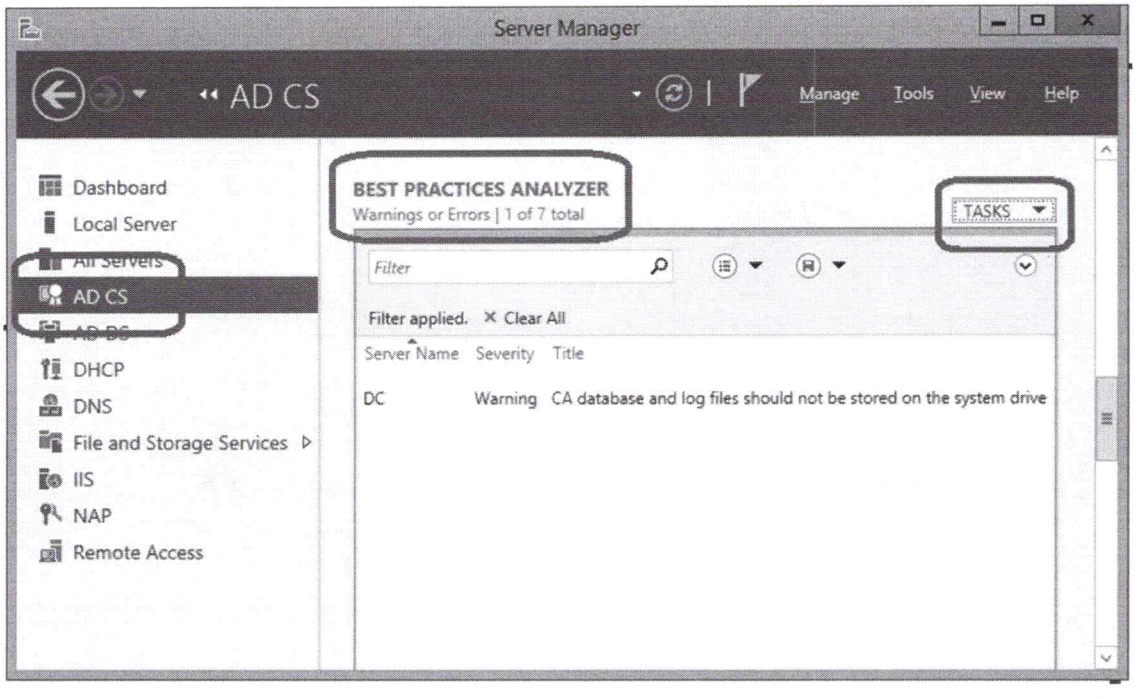
The script will monitor and log the following:

- KRA Certificate expired (Event Log ID 30)
- KRA Certificate remaining validity is less than one month (Event Log ID 31)
- KRA Certificate has been revoked (Event Log ID 32)

- KRA Certificate is not trusted (Event Log ID 33)
- Certificate Services service client RPC interface offline (Event Log ID 1)
- Certificate Services service admin RPC interface offline (Event Log ID 2)
- CA Certificate expired (Event Log ID 10)
- CA Certificate remaining validity is less than one month (Event Log ID 11)
- CA Certificate remaining validity is less than half its lifetime (Event Log ID 12)
- CA Certificate has been revoked (Event Log ID 13)
- CRL expired (Event Log ID 20)
- CRL overdue (Event Log ID 21)
- CRL cannot be retrieved from Active Directory (Event Log ID 22)
- CRL cannot be retrieved from Web server (Event Log ID 23)

Run The Best Practices Analyzer (2008-R2 and Later)

Finally, in Server 2008-R2 and later, run the Certificate Services Best Practices Analyzer by opening Server Manager > select the "AD CS" role on the left > on the right-hand side scroll down to the Best Practices Analyzer section > Task pull-down > Start BPA Scan. It is very important to run the tool because many (if not most) PKIs are set up incorrectly, which later results in problems which are hard to diagnose or reverse out.



The Best Practices Analyzer will display a list of tests performed against the configuration of your CA and Active Directory, then offer guidance on how to resolve any issues.

Security Patches versus Updates for Known Issues

Of course we should always apply the latest patches, especially security patches, but there are some PKI updates which are not distributed through the automatic Microsoft update mechanisms. These updates might be called "hotfixes for known issues" or some other PR verbiage, but some or all of these should be installed too. The problem is how to find them.

At the time of this writing, the following web site is a convenient way to research these known-issue updates (and offers other useful PKI information too):

http://pkisolutions.com/adcs-hotfixes/

If this web site falls behind or is abandoned, then we are back to doing keyword searches on Microsoft's web site, usually after a problem occurs. Ideally, though, it would be better to deploy these updates before the problems occur.

How Do I Copy and Edit Certificate Templates?

You can only edit templates on:

- Enterprise Certification Authorities (not Stand-Alones)
- Windows Server 2003 Enterprise Edition or Later

Certificate Templates MMC Snap-In

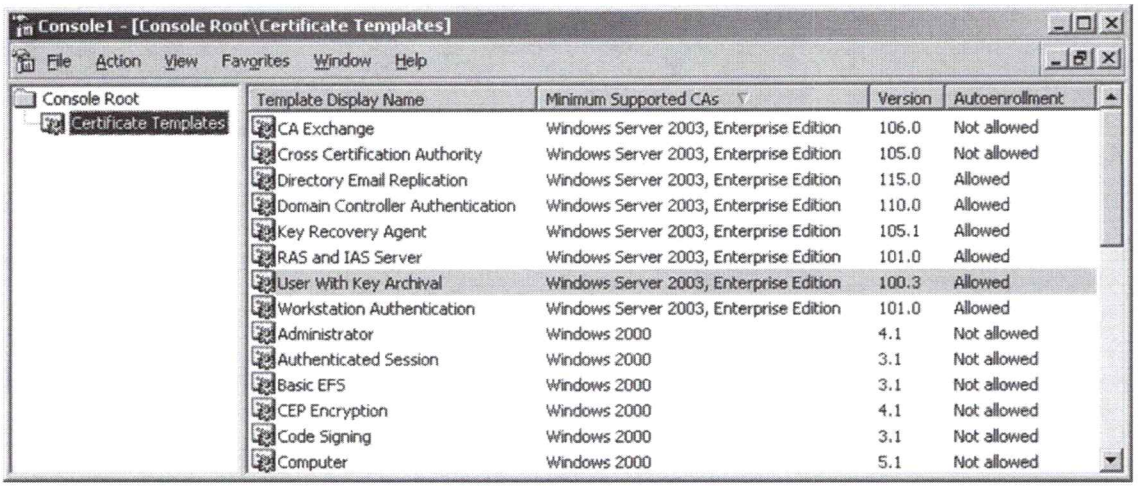
- **Some example uses:**
  - Copy an older template to make an editable duplicate
  - Require private key archival
  - Change permissions on the template itself
  - Enable auto-enrollment
  - Force re-enrollment for all current certificate holders

SANS SEC505 | Securing Windows

How Do I Copy And Edit Certificate Templates?

Every certificate issued from an Enterprise CA is based on a template. If your Enterprise CA is installed on Windows Server 2003 Enterprise Edition or later machine, then you can copy and edit your templates to suit your needs (note that starting with Server 2012 there is no longer an Enterprise Edition, hence, the ability to edit certificate templates is included in Standard Edition as well on Server 2012 and later). The CA must be an Enterprise CA, however, to use edited templates, it cannot be a Stand-Alone CA.

The primary tool for copying and editing templates is the Certificate Templates snap-in.



Template Display Name	Minimum Supported CAs	Version	Autoenrollment
CA Exchange	Windows Server 2003, Enterprise Edition	106.0	Not allowed
Cross Certification Authority	Windows Server 2003, Enterprise Edition	105.0	Not allowed
Directory Email Replication	Windows Server 2003, Enterprise Edition	115.0	Allowed
Domain Controller Authentication	Windows Server 2003, Enterprise Edition	110.0	Allowed
Key Recovery Agent	Windows Server 2003, Enterprise Edition	105.1	Allowed
RAS and IAS Server	Windows Server 2003, Enterprise Edition	101.0	Allowed
User With Key Archival	Windows Server 2003, Enterprise Edition	100.3	Allowed
Workstation Authentication	Windows Server 2003, Enterprise Edition	101.0	Allowed
Administrator	Windows 2000	4.1	Not allowed
Authenticated Session	Windows 2000	3.1	Not allowed
Basic EFS	Windows 2000	3.1	Not allowed
CEP Encryption	Windows 2000	4.1	Not allowed
Code Signing	Windows 2000	3.1	Not allowed
Computer	Windows 2000	5.1	Not allowed

The most likely reasons you'll need to edit a template are to enable auto-enrollment of user-related certificates, force the re-enrollment for an updated certificate of a certain type, and to enable automatic private key archival on certain templates.

Example: Make A Copy Of A Template

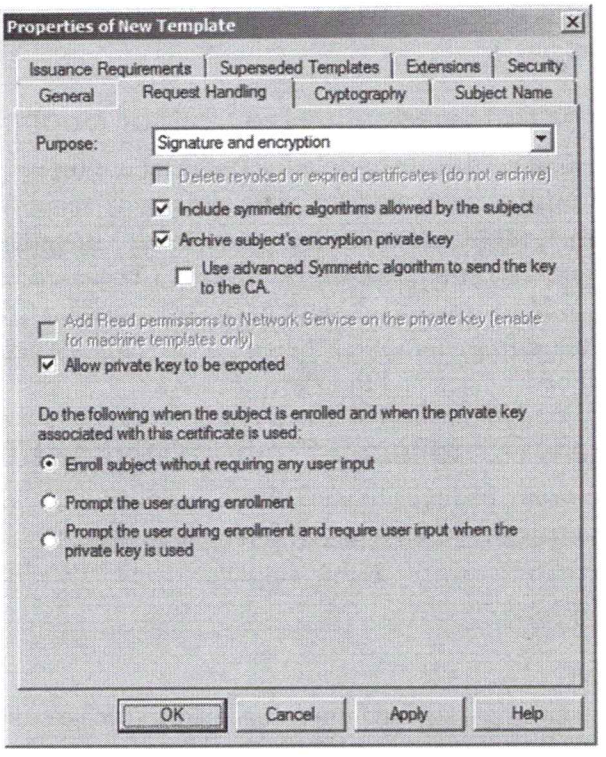
Your Windows Server Enterprise Edition CA may have templates installed that are backwards-compatible with older Windows CAs. This is because you may have a mixture of CA versions in your organization. However, these older templates are not editable. You can, however, make a copy of an older template and save it in the newer (editable) format. This is something you'll have to do very often.

Try It Now!

To make an editable copy of a template, right-click that template and select Duplicate Template. You will be prompted to choose the oldest version CA in use because newer templates have features not supported by older CAs. After selecting the oldest CA version that must be supported, on the General tab give your new copy a different name; for example, if you are duplicating the User template, you might name the new version "User Cert With Key Archival" as a reminder both of where the new template came from and what its intended purpose is.

Example: Require Private Key Archival

Once you've got an editable template (either because it's built-in or you've made a copy of an older one) then you can enable private key archival for all certificates issued from that template. This will place an encrypted copy of the certificate's associated private key in the database of the Enterprise CA (private key archival will be discussed later).



Try It Now!

To mark an editable template for automatic private key archival, right-click the template > Properties > Request Handling tab > check the box "Archive Subject's Encryption Private Key" > OK.

Example: Enable Certificate Auto-Enrollment

Once you've got an editable template (either because it's built-in or you've made a copy of an older one) then one of the necessary steps for setting up certificate auto-enrollment is to change the permissions on the template to allow it. You can also select how much user intervention you want during the enrollment process (auto-enrollment will be discussed later).

Try It Now!

To permit auto-enrollment of certificates from an editable template, right-click the template > Properties > Security tab > add the necessary group(s) and grant them the Read, Enroll and Autoenroll permissions on the template. Next, go to the Request Handling tab and choose the level of user involvement you desire with the radio buttons in the middle, e.g., "Enroll Subject Without Requiring Any User Input". Click OK.

Note that if auto-enrollment is for a smart card certificate, the user will get a pop-up message indicating that the certificate is ready to be installed on the card (assuming they have a card, know the PIN, they're capable of inserting it without causing physical harm to themselves or the computer, etc.). If the type of smart card reader displayed is not on the user's machine, the user should repeatedly click Cancel until the correct reader model

is displayed (assuming they know what this model is, they're capable of clicking Cancel without freaking out, etc.).

Example: Do Not Automatically Reenroll for Duplicate Certificates

On the General tab of a certificate template there is an option named "Do not automatically reenroll if a duplicate certificate already exists in Active Directory". You will almost certainly want this box checked on templates intended for users. This option goes with the credential roaming feature very nicely (discussed later) to help ensure that each user gets only one certificate of that type and that that certificate (and private key) follows the user around from computer to computer. This is very important, for example, for S/MIME e-mail certificates and their private keys.

Example: Force Re-Enrollment From An Updated Template

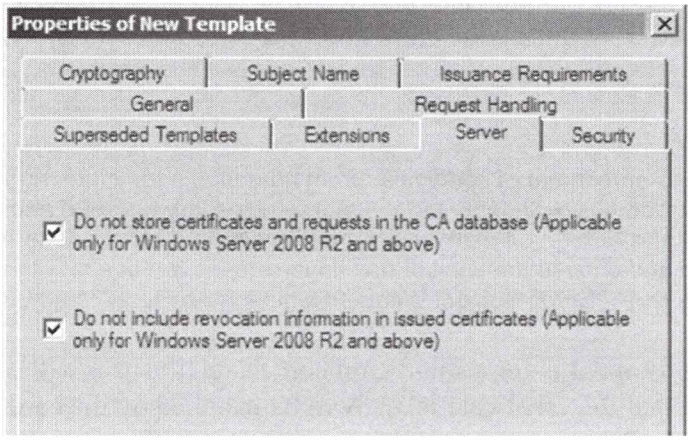
If you've made important changes to an editable template with which users have already auto-enrolled for certificates, then you can force those users to re-enroll for an updated certificate from the newer version of the template. The re-enrollment will occur with the next Group Policy refresh.

Try It Now!

To force auto-enrollment again from an updated certificate template, right-click that template and select "Reenroll All Certificate Holders".

Example: Do Not Store Certificates And Requests In The CA Database

If you are issuing certificates with very short TTLs, such as for NAP, or if you are issuing a very large and growing number of certificates for any reason, then Server 2008-R2 and later CAs have an option to prevent the storage of certificates and requests in the CA database. This will prevent the database from growing rapidly, but of course you will no longer have a CA backup of these certificates or their private keys. You also have the option to exclude revocation URLs (the CDP field) from these certificates.

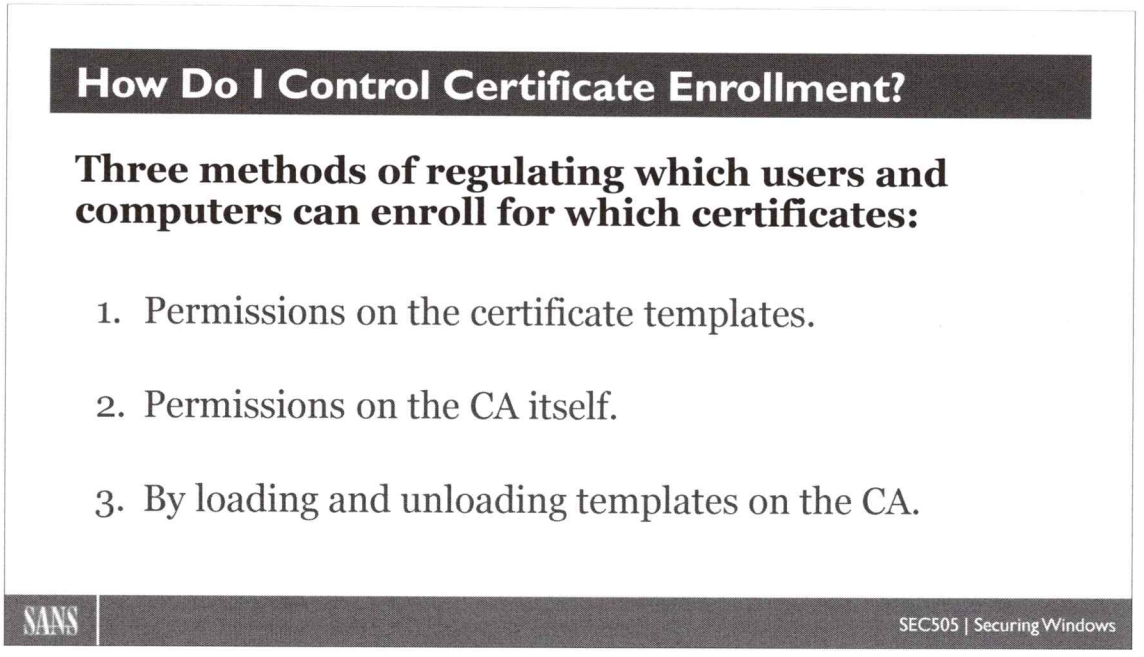


Cross-Forest Replication of Templates

Prior to Windows Server 2008-R2, you could only request certificates from CAs within your own AD forest. With a single 2008-R2 or later CA, however, you can issue certificates to clients in any mutually-trusted forest. This capability requires replicating the certificate templates in the home forest of the CA to the other forests, and this is currently only possible with a Microsoft-supplied PowerShell script which should be scheduled to run periodically to ensure that all templates are in sync.

What About All The Other Options In Certificate Templates?

Unfortunately, there's just not enough time to discuss all the configurable options inside your templates. However, we've discussed the most likely reasons you'll need to use this capability. If you're interested in the other options, please pull down the Help menu in your console and read more about the Certificate Templates snap-in there.



How Do I Control Certificate Enrollment?

There are three methods of controlling who can enroll and for what kind of certificate:

- Permissions on the CA itself.
- By (un)loading templates into the CA.
- Permissions on individual certificate templates.

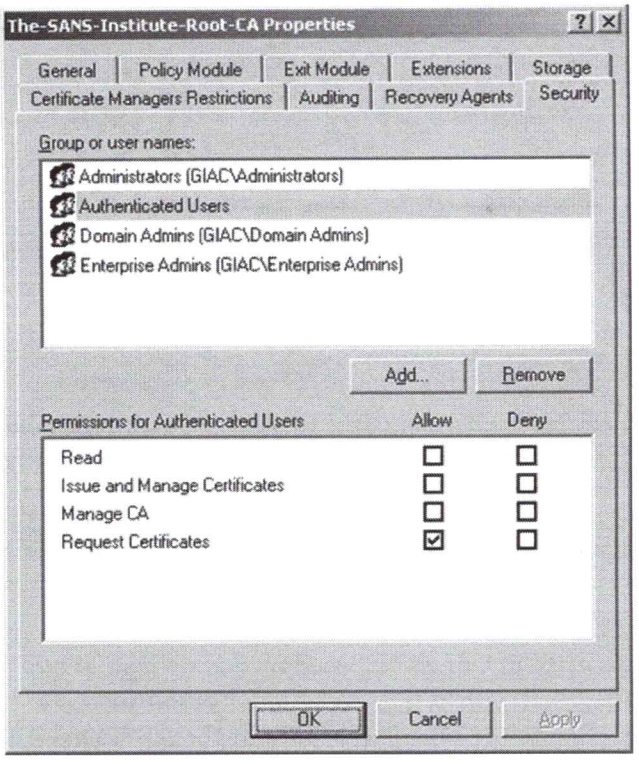
Note: Additional issuance requirements can be mandated in editable templates to further regulate enrollment, but these will have to wait for a week-long PKI course.

Permissions On The CA

A user or computer must have at least the Request Certificates permission on the CA itself in order to request a certificate. CA permissions are configured with the Certification Authority snap-in, properties of the CA computer.

By default, the Authenticated Users group will have Request Certificates, but this should be changed to include only the group(s) who actually need to request certificates, following the principle of least privilege.

> ***Try It Now!***
> To control who can enroll for any type of certificate on a CA, open the Certification Authority snap-in > properties of the CA > Security tab.



Permissions on the CA is also how you delegate authority over the CA to others. You don't have to be a full Domain Admin to manage a CA, you only need the Manage permission on the CA itself.

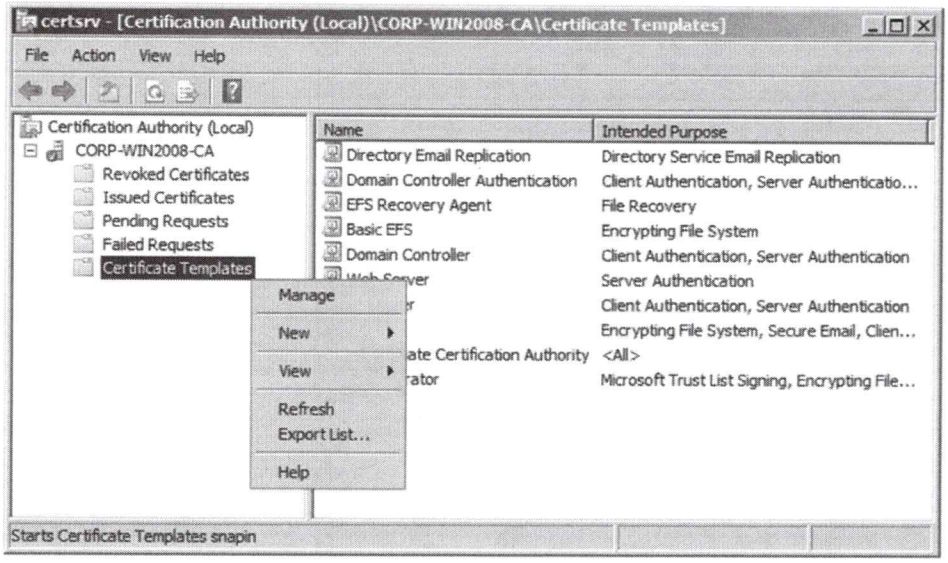
Be aware that when clients choose a CA for enrollment, neither the CA nor the client is site-aware in this regard, hence, use the permissions on the CA to compel clients to choose the correct CA when there are multiple geographically-distributed CAs from which to choose.

Loaded Certificate Templates

Before a template can be used on an enterprise CA, the CA must permit the use of the template. The Certification Authority snap-in includes a Certificate Templates container under the CA server. A template must be listed in this container in order to be used by the CA.

Try It Now!

To make a certificate template available for use on a CA, open the Certification Authority snap-in > right-click the Certificate Templates container > New > Certificate to Issue > select a certificate template > OK. To remove the template, right-click > Delete.



Certificate Template Permissions

All certificates issued from an Enterprise CA must be based on a certificate template. There are a number of templates included with Windows. Each has one or more purposes. The template defines the contents of many of the fields in a X.509 certificate. The following is a list of the built-in templates:

- Administrator
- CA
- CEPEncryption (off-line)
- ClientAuth
- CodeSigning
- CTLSigning
- DomainController
- EFS
- EFSRecovery
- EnrollmentAgent
- ExchangeUser (off-line)
- ExchangeUserSignature (off-line)
- IPSECIntermediate (off-line)
- IPSECIntermediate
- Machine
- MachineEnrollmentAgent
- Router (off-line)
- SmartCardLogon
- SmartCardUser
- SubCA
- User
- UserSignature
- WebServer

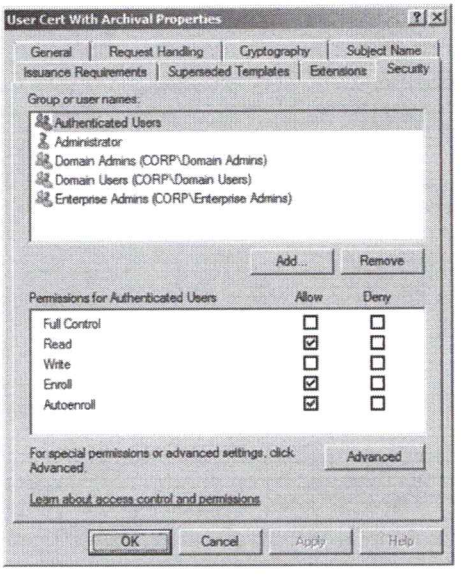
When a template is tagged for "Off-Line Requests", the Certificate Services Website should be used to request the certificate. Off-line requests are for users that do not have Active Directory accounts or cannot request certificates directly from an enterprise CA. For example, the CEPEncryption template is for Cisco router IPSec authentication, but the certificate must be installed manually on the router.

Templates are stored in Active Directory, hence, each template can have its own permissions ACL. A user or computer must have at least the Enroll permission to a template in order to request a certificate based on it. The permissions on certificate

templates are the locus of enrollment control. Permissions are set using the Certificate Templates snap-in.

Try It Now!

To change the permissions on a certificate template, open the Certification Authority snap-in > right-click the Certificate Templates container > Manage > right-click the desired template in the new console > Properties > Security tab. A user or computer must have the Enroll permission to obtain a certificate based on the template, or, if the request is through Group Policy, the Autoenroll permission.



Stand-alone CAs do not use templates. All certificate requests on stand-alone CAs remain pending until an administrator explicitly approves the certificate enrollment. Hence, the control of certificate enrollment on stand-alone CAs is directly in the hands of administrators-- they control enrollment by their decisions.

Best Practices

- Remove the Authenticated Users group from the permissions ACL on the CA itself. Add only those groups whose users and computers will actually need to obtain certificates. Don't forget that computers can be members of groups.

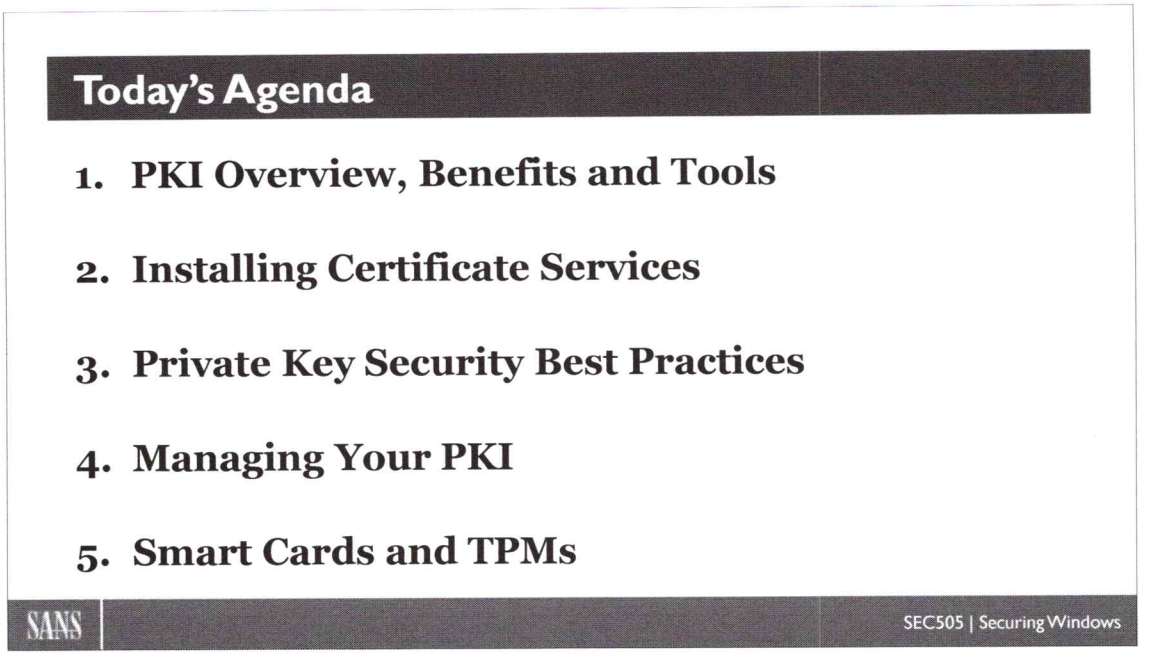
- Create a local group named "CA Admins" on each CA computer. Give this group the Manage permission on the CA. Use this group to delegate authority over the CA, if necessary, without granting the delegates full administrative power on the computer or in the domain.

- Remove all unneeded certificate templates from the Certificate Templates container. Keep in mind that a template can be added temporarily when needed and then removed again. When in doubt, remove the template.

- Assign the Everyone group Deny/No Access to all templates, then change this permission as needed to allow just the desired users and computers to request certificates. Keep in mind that permissions can be changed temporarily as needed, then reset to Everyone:Deny. In particular, be very wary of changing Everyone:Deny for the following templates:

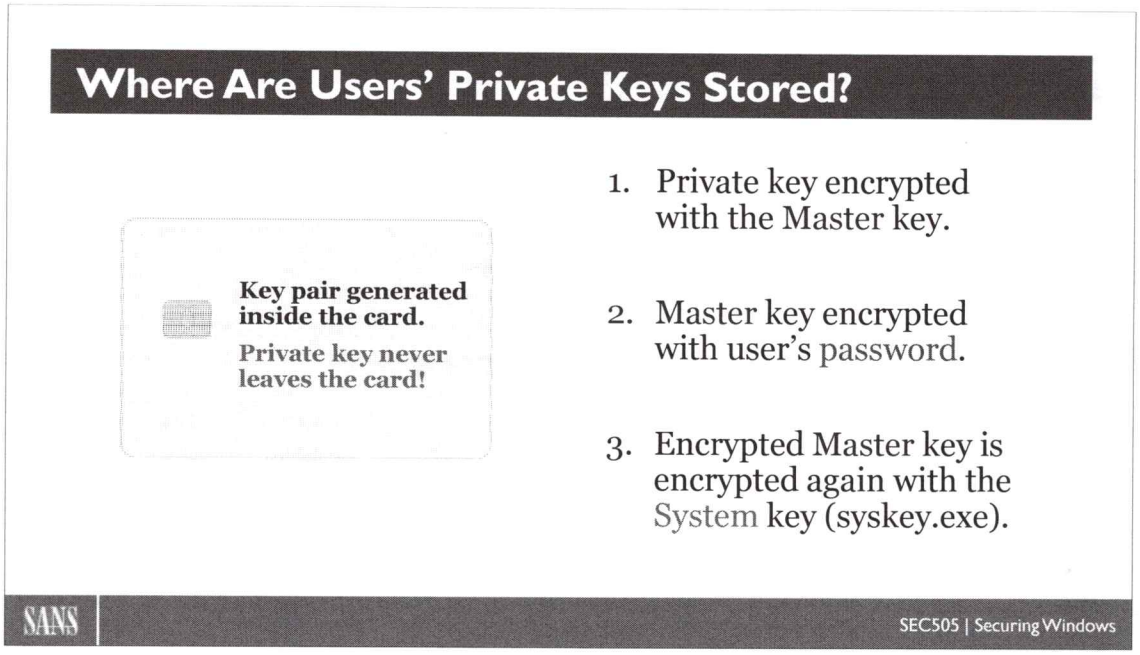
    - Administrator
    - CA
    - SubCA
    - EnrollmentAgent
    - EFSRecovery
    - CodeSigning
    - CTLSigning

- Audit all failed access to all certificate templates.



Today's Agenda

Now that your PKI is in place and your certificate server(s) are ready to start cranking out certificates, we must discuss the most important question to ask of any vendor's PKI: Where are private keys stored and how are they secured? The goal of this section is to come up with a strategy for enhancing the security of private key protection as much as possible without losing functionality.



Where Are Users' Private Keys Stored?

How a vendor implements private key storage and protection is critical for evaluating that vendor's PKI. On the one hand, private keys must be as secure against compromise as possible. On the other hand, private keys must be transparently accessible to the users and computers that own them to make the infrastructure as efficient and easy-to-use as possible. These two needs conflict with each other and must be balanced.

When using a smart card or smart USB token (or a TPM as a virtual smart card), the private key is generated within the device and never leaves it. Whenever data must be encrypted or decrypted with the private key, the data must be copied to and from the smart card. If a smart card permits the extraction of the private key, do not use it, this would defeat the main purpose of hardware-based key protection, namely, protection from malware running in kernel mode as a part of the operating system itself. Think of a smart card as a tiny computer within your computer.

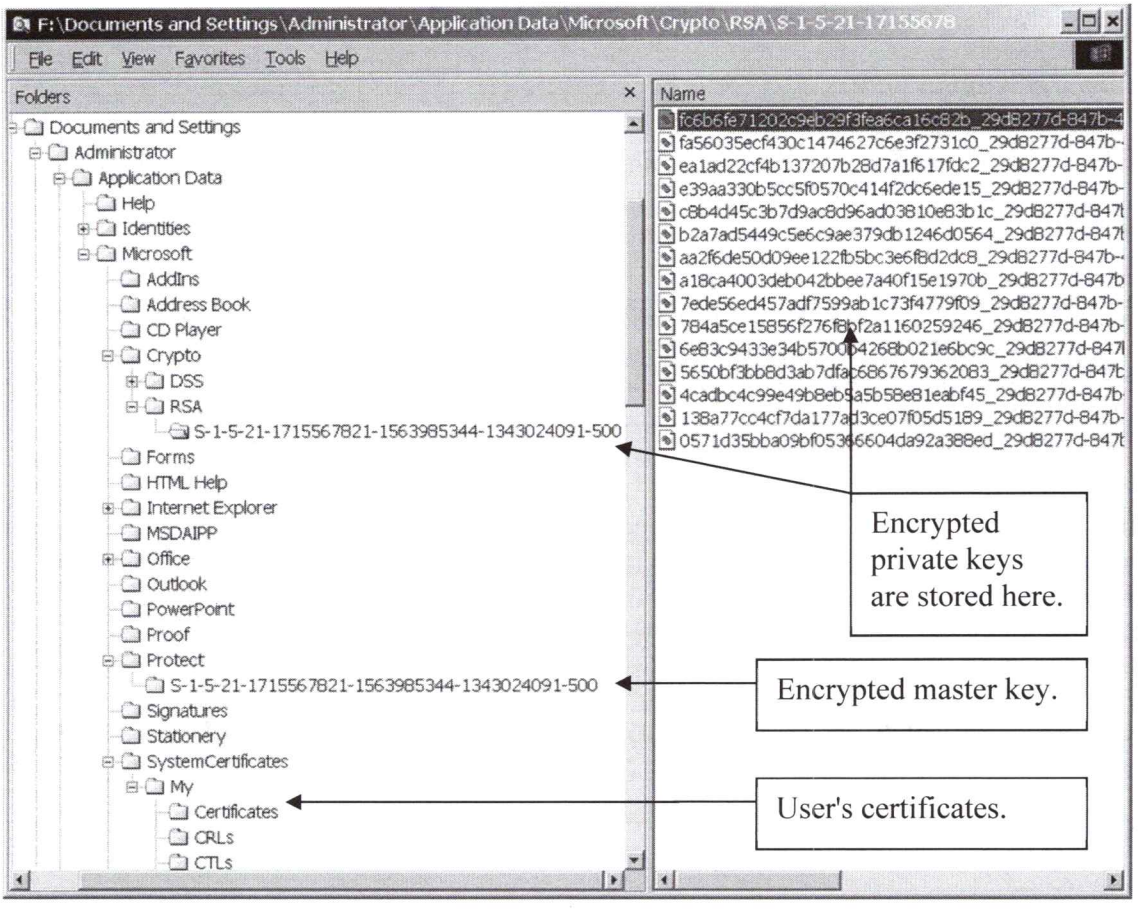
When not using a smart card, a user's private keys are stored in his or her user profile folder and are encrypted by the operating system (see below). If the profile is roaming, then the private keys are copied and moved with the rest of the profile. When a user logs off, their locally-stored copy of their profile is not deleted by default, so a copy of the user's encrypted private keys are left on every machine he or she uses (though there is a Group Policy option to delete the local profile after logoff). With roaming profiles, encrypted private keys are also kept on the file server containing the central copy of everyone's roaming profiles. If the credential roaming feature is enabled, which is discussed elsewhere in this manual, then encrypted copies of users' private keys are also stored in Active Directory, where each user's keys are stored as attributes of that user's account in AD.

Private Key Storage on Windows XP/2003

On Windows XP/2003, profiles are stored under the "C:\Documents and Settings" folder, with a subdirectory for each user who has logged onto that machine. The names of the subdirectories are the same as their usernames. Private keys are stored in subfolders in the profile, where the names of the folders are identical to the user's Security ID (SID) number:

%APPDATA%\Microsoft\Crypto\RSA\<*user's SID number*>\
%APPDATA%\Microsoft\Crypto\DSS\<*user's SID number*>\

Each file in this folder is an encrypted private key. The first 32 digits of the names of these private-key-files acts as an identifier to match it with its corresponding certificate (and public key).



If you open the private key with Notepad, you can see the above number at the beginning, perhaps padded with some extra 0's. It will look something like this:

4350af04-3bd3-48e6-979c-b7c44a4fc221 RSA1

If you open the corresponding certificate in Notepad, you will see the same number near the top of the file. One of the places where you can find your certificate(s) is in your profile in the \Certificates folder:

%APPDATA%\Microsoft\System Certificates\My\Certificates\

The names of each of the certificate files here is identical to the SHA thumbprint of your certificate. You can see this thumbprint number if you open the properties of your certificate file in the Certificates MMC snap-in. Match the fingerprint number against the filenames in this folder to locate the correct certificate.

Private Key Storage on Windows Vista, Server 2008 and Later

Windows XP/2003 uses the older CryptoAPI interface, while Windows Vista/2008 and later uses the Cryptography Next Generation (CNG) interface. In Windows Vista/2008 and later, a user's new private keys are created in the following folder:

%APPDATA%\Microsoft\Crypto\Keys

For example, for a user named "<*username*>" the path would be:

C:\Users\<*username*>\AppData\Roaming\Microsoft\Crypto\Keys

However, for backwards compatibility, private keys may also be stored in the same folders as on Windows XP/2003, hence, for a user with a Security ID number of "<*SID*>" also look in the following folder:

C:\Users\<*user*>\AppData\Roaming\Microsoft\Crypto\RSA\<*SID*>

If an application using the CNG interface desires to use a private key, if that key can't be found in the default folder, Windows will search the old XP/2003 folders automatically. Also, when an RSA-based key is created, two copies might be stored: one in the CNG folder and another in the older CryptoAPI XP/2003 folder; but this isn't always the case, it's up to the application which requested the key to be generated in the first place.

How Are My Private Keys Encrypted On Windows 7/2008 and Later?

Private keys are encrypted by the Data Protection API (DPAPI) in Windows. Unfortunately, the documentation we have about the inner workings of the DPAPI is somewhat sparse and the source code is unavailable.

> **Note:** The following information has been constructed from Microsoft whitepapers, the MSDN Library, help files, release notes, IETF draft documents, third-party reverse engineering articles, and other sources. Unfortunately, many of these documents contradict each other, and Microsoft has not documented the details of the differences between the various OS versions. Hence, the following is *likely* how private keys are protected, but this cannot be guaranteed.

Because the description is fairly complex, I've put it into the form of a Q&A to make it more digestible. The best references for this information are the following articles:

- Windows Data Protection
  http://msdn.microsoft.com/en-us/library/ms995355.aspx

- DPAPI Secrets
  http://www.passcape.com/index.php?section=blog&cmd=details&id=20

What are DPAPI and DPAPI-NG?
The Data Protection API (DPAPI) is a set of functions implemented by crypt32.dll which relays requests to encrypt/decrypt data down into the Local Security Authority process (lsass.exe), hence, DPAPI functions are just user-mode wrappers for these cryptographic services in the LSA. An application gives plaintext secrets to the DPAPI, encrypted blobs are returned, and the application stores the enciphered blobs wherever it wishes, e.g., in the registry, in a text file, in an Active Directory attribute, etc. The DPAPI does not provide storage of encrypted secrets, it only provides encryption/decryption services; it is up to the process using the DPAPI to store the encrypted data, to back up that data, to synchronize that data across multiple computers if it desires, etc.

DPAPI Next Generation (DPAPI-NG) is found on Windows 8, Server 2012 and later systems. It is also used with ASP.NET 4.5 and later. DPAPI-NG is built on top of CNG and was added to support the ability to encrypt a secret on one computer and still access that secret on another computer, a requirement for various cloud computing scenarios. Such multi-computer secrets are only available to Active Directory groups and ASP.NET web credentials -- similar, perhaps, to multi-server Managed Service Accounts, but little information is currently available.

What are examples of secrets protected by the DPAPI?
The DPAPI is used by many applications and for many secrets, including Internet Explorer saved passwords, Outlook saved passwords, keys for 802.11 wireless networks, Windows Credential Manager passwords, saved Remote Desktop passwords, VPN user passwords, and the private keys for both users and services. Private keys include those for S/MIME e-mail, Encrypting File System (EFS), and IIS web server SSL certificates.

How are private keys encrypted?
Each private key is encrypted with its own 256-bit AES key called the "Session Key" on Windows Vista, Server 2008 and later. (On XP and Server 2003, it is a 168-bit 3DES key instead.)

Very roughly, the following diagram summarizes the flow of events, where "↓" means "is used to create or encrypt the item below it", hence, we start with the user's passphrase and end with the user's private key being encrypted. Salts are just random bits generated by the OS and stored somewhere in the registry or file system in plaintext.

SHA-512 hash of user's passphrase + 128-bit salt

↓
Protection Key (256-bit AES)
↓
Random 512-bit Master "Key" + 128-bit salt
↓
Session Key (256-bit AES)
↓
User's Private Key

How is the Session Key derived?
For each private key that needs to be encrypted, a 128-bit random salt is generated. These bits and the 512-bit Master Key are hashed with SHA-512. This hash, along with an optional user-provided password for "Strong Protection" (see below), are hashed again through a CryptoAPI call (CryptHashData), but the hashing algorithm used here is not documented. Whatever this hash is, it is converted into a symmetric encryption key by passing it into another CryptoAPI call (CryptDeriveKey) which derives a key from the hash. The Session Key is that which actually encrypts the user's private key.

What is the Master Key and where is it stored? Where does SYSKEY.EXE fit in?
The Master "Key" is a blob of 512 random bits, but it is not really a key because it is not directly used to encrypt anything. The Master Key is AES-CBC encrypted with the 256-bit Master Key Protection Key (MKPK).

The OS changes the Master Key automatically every 90 days. Each version is assigned a unique GUID number. The 90-day refresh interval is hard-coded into the OS.

An HMAC-SHA-512 signature of the encrypted Master Key is also made with the Master Key Protection Key and stored with the Master Key. This HMAC signature, the cleartext GUID of the Master Key, the cleartext 128-bit salt (see next), and the cleartext iteration count (see next) are all stored with the encrypted Master Key in a file in the %AppData%\Microsoft\Protect*SIDnumber*\ folder. The file named "Preferred" in that folder identifies which Master Key is the most recent and should be used.

This file is also encrypted with the System Key while on the hard drive. The System Key is 128-bit RC4 and is managed with the SYSKEY.EXE utility.

Keep in mind that the Master Key is cached in the kernel memory space of the LSA (lsass.exe) after the user logs on, hence, malware also running in kernel mode can potentially acquire a copy of it to decrypt the user's private keys.

What is the Master Key Protection Key (MKPK) and how is it derived?
At logon, the user's password hash is hashed again with SHA-512. This SHA-512 hash, along with a randomly generated 128-bit salt and an iteration count (default: 5600) is handed into a PKCS#5 Password-Based Key Derivation (PBKDF2) function. PBKDF2 hashes the user's password hash and salt 5600 times again with SHA-512. The output is the Master Key Protection Key (MKPK), which is used to protect the Master Key.

What about when a user logs on with a smart card?
For smart card logon, what exactly gets hashed is not currently documented, but private conversation with a security consultant at Microsoft indicates that the hash of the user's password is decrypted with the private key from the smart card, having been previously encrypted with the card's public key. This hash is then presumably fed into the PKCS#5 function in the regular way. What password, you ask? A random password is generated for smart card users, encrypted with the public key from the card, and then stored locally. This is how it is possible to log on with cached credentials with a smart card even if disconnected from the network. What are the details of all this, you ask? It's just not documented.

What is that PKCS#5 function used to derive the Master Key Protection Key?
RSA Laboratories (www.emc.com) has authored a set of widely-used *de facto* PKI standards called "Public-Key Cryptography Standards (PKCS)" which are each numbered. PKCS#5 is the "Password-Based Cryptography Standard" for deriving a pseudo-random key of arbitrary length from some other arbitrarily long input, like a password, a salt, and an iteration count.

PKCS#5 is not secret and its specification can be downloaded from the RSA website. PKCS#5 is considered a secure method of deriving keys as long as the iteration count is higher than 1000. As the length and complexity of the password goes up, so does the strength of the PKCS#5 key derived from it. The salt and iteration count number can be stored in cleartext without compromising safety.

Windows 7 uses a default iteration count of 5600. This can be increased by adding a value named MasterKeyIterationCount under the following registry key: HKLM\Software\Microsoft\Cryptography\Protect\Providers*GUIDnumber*\.

If the Master Key is changed every three months, what about older Session Keys?
All previous Master Keys are archived with their GUID numbers to identify them. The old Master Keys are encrypted with the current Master Key Protection Key described above. Because the Session Key data are stored with the GUID of the Master Key that go with them, the correct Master Key can be located.

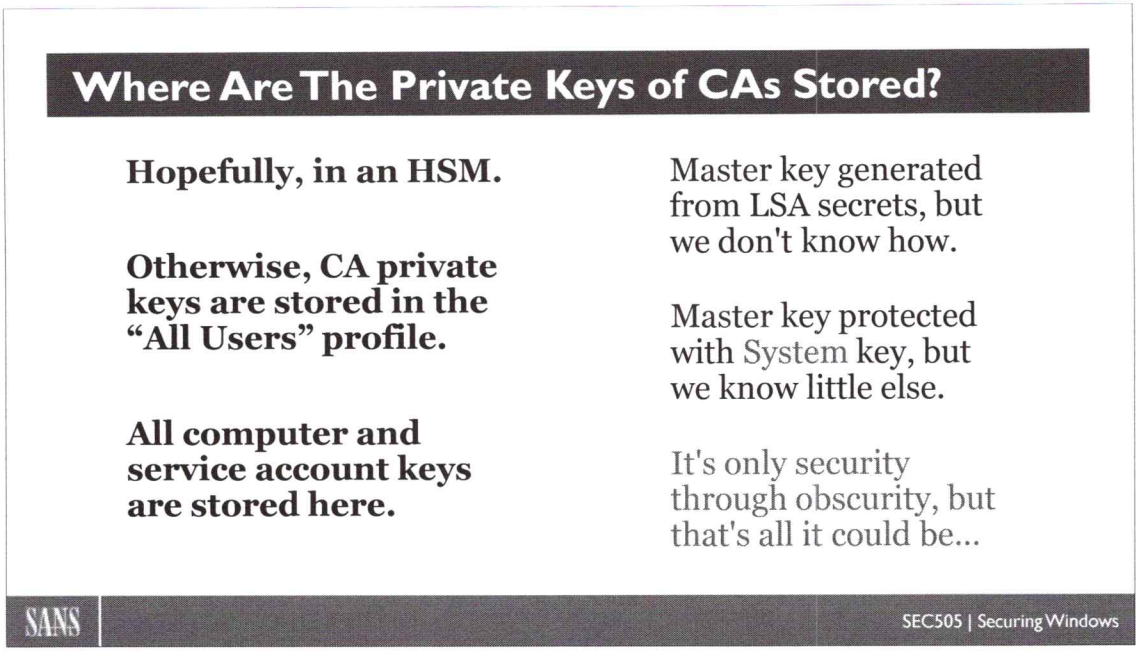
What happens when a user changes his or her password?
When a user changes his or her password, all Master Keys, current and archived, are re-encrypted using the new password.

Also, the SHA-512 hashes of all of one's prior passwords are encrypted and stored in the %UserProfile%\Application Data\Microsoft\Protect\CREDHIST file. Each new password is used to encrypt the prior password hash, which is then appended to the top of the CREDHIST file. If a prior password is needed to decrypt a prior Master Key, the current password is used to decrypt the last password's hash, that password hash is used to decrypt the prior one before that, and so on, until the necessary password hash is obtained for decrypting the desired archived Master Key.

What if a user without a roaming profile changes their password?
If roaming profiles are not used a problem occurs when a user changes their password and logs on at another machine where they already have a different Master Key. In this scenario you would expect the user to be incapable of decrypting their local non-roaming Master Key because their domain password has changed! Fortunately, there is another recovery mechanism to take care of this.

All domain controllers in a domain have an identical 2048-bit RSA public/private key pair used for the Data Protection API. This key pair is replicated (details are not documented). When a computer is a member of the domain, it downloads a copy of this public key from its DC. The documentation on this channel merely states that it is an "authenticated and encrypted RPC channel", but the details are not known. It is likely the NetLogon channel. The client computer encrypts a second copy of the user's Master Key with the public key from the DC. This second copy is also stored in the same file as the Master Key encrypted with the user's password. If the user cannot access their local non-roaming Master Key because they've changed their domain password, the public-key-encrypted copy of the Master Key is sent up to its DC over that RPC channel, decrypted with the DC's private key, and sent back down the channel to the client computer. The encryption of the RPC channel is the only thing protecting the Master Key. Also, what if that DPAPI private key replicated among the domain controllers is lost? An adversary could then, presumably, decrypt any private key on any machine in the domain!

How does the Password Recovery USB flash drive work with local accounts?
The password of a local user account (not global) can be reset if the user has chosen to create a Password Recovery Disk in Control Panel. This disk is usually a USB flash drive. A 2048-bit RSA public/private key pair will be created. The private key stays on the flash drive. The public key is copied to the hard drive and used to encrypt a copy of the user's password. Because of the CREDHIST file, a user will always be able to log on with a Password Recovery Disk and gain access to all their private keys again.



Where Are The Private Keys of CAs Stored?

If the CA's Cryptographic Service Provider (CSP) is a smart card or dedicated cryptographic hardware module, then the private keys will be stored on the card or module. All cryptographic operations which use the private key will also occur inside the device to isolate the key. This is true for both clients and CAs. In general, hardware CSPs provide the best private key protection.

Windows Server 2003

If the CA is Windows Server 2003 and using Microsoft's software CSP, the CA's private key will be stored in the profile for "All Users" in the \MachineKeys folder on the hard drive. This is true for both stand-alone and enterprise CAs. Almost always, the RSA-based CSP is used, but some government or military environments might use DSS.

…\All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys\
…\All Users\Application Data\Microsoft\Crypto\DSS\MachineKeys\

If various services on the computer have their own separate private keys, those keys might not be stored in the MachineKeys folder, they might be stored in folders named after the SID of the service account. Instead of MachineKeys, you might see folders similar to the following:

..\S-1-5-18 (for Local System)
..\S-1-5-19 (for Local Service)
..\S-1-5-20 (for Network Service)

Windows Server 2008 and Later

Windows Server 2008 and later uses the Cryptography Next Generation (CNG) interface and its non-user key storage locations are slightly different. If the CA is using Microsoft's software-based CSP, the key appears to be stored here:

%ALLUSERSPROFILE%\Application Data\Microsoft\Crypto\Keys

But after expanding the environment variable (%ALLUSERSPROFILE%) and following the folder junction (\Application Data) to its target folder, the real path on the hard drive will be something like this:

C:\ProgramData\Microsoft\Crypto\Keys

If various services have their own private keys, those keys might be stored in different locations depending on how the service is designed and under which account it runs. Here are the additional folders where you might find more service private keys:

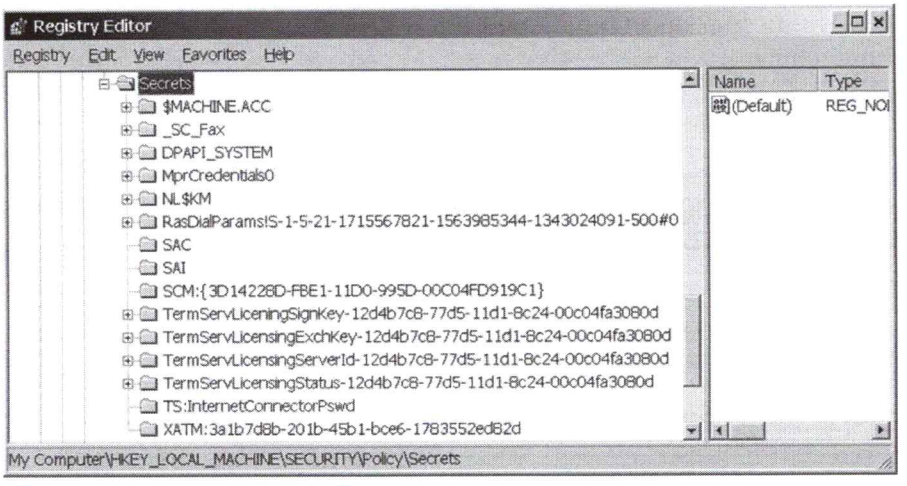
%ALLUSERSPROFILE%\Application Data\Microsoft\Crypto\SystemKeys	(for Local System)
%WINDIR%\ServiceProfiles\LocalService	(for Local Service)
%WINDIR%\ServiceProfiles\NetworkService	(for Network Service)

How Is The Private Key Of The CA Encrypted?

It is not really documented. Microsoft has stated in the past that the private keys of services are protected by the computer's Master Key, which is derived in part from the LSA Secrets of the computer, and that the LSA secrets are encrypted with the computer's System Key (the key managed with the SYSKEY.EXE tool). Beyond this, though, Microsoft hasn't disclosed any further details.

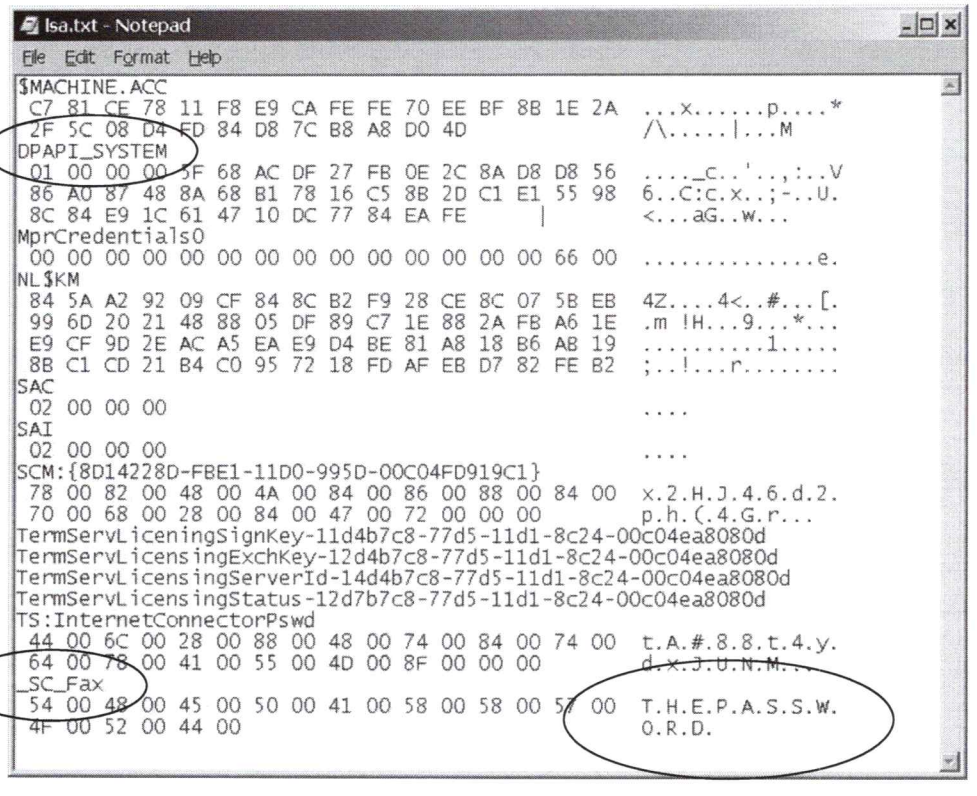
The computer's Master Keys are located under %WinDir%\System32\Microsoft\Protect in various subdirectories named after well-known system SID numbers.

What are "LSA secrets"? The Local Security Authority (LSA) is both a service and a protected subsystem which handles security-sensitive operations such as user logon (see lsass.exe in Task Manager). The LSA also stores encrypted data in the registry, such as cached user dial-up passwords and service account passwords used to start services at bootup. These are the "LSA secrets" and they stored in the registry under HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets. Use REGEDIT.EXE to change the registry permissions to allow you to see the data.

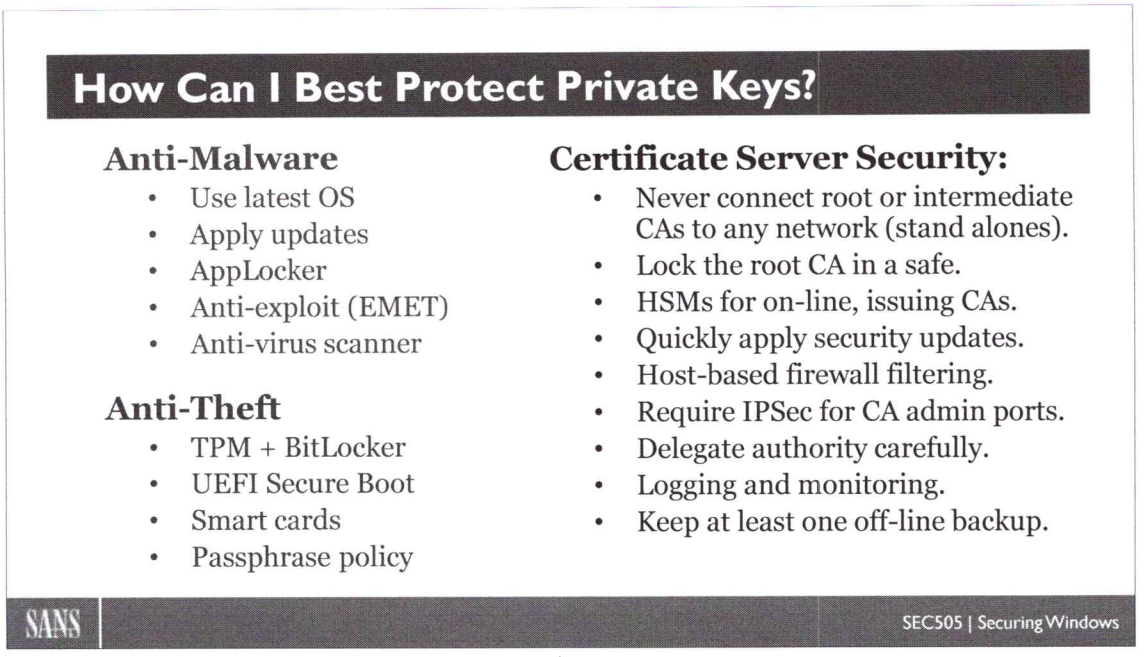


For example, any service not configured to start under the System account will have a subkey here named _SC_*servicename*, such as "_SC_Fax". This key will have a value that holds the password for the user account under which the service does start.

However, if you have the SeDebugPrivilege user right as a local Administrators member, you can see the plaintext LSA secrets with utilities like LSADUMP2.EXE or Cain (www.oxid.it). By default, only Administrators have this user right. The output of LSADUMP2.EXE looks like the following screenshot (notice the cleartext password for the Fax service at the bottom).



A computer's Master Key is derived from LSA Secret information *somehow*, but the details are undocumented. On the other hand, do you see that DPAPI_SYSTEM entry in the screenshot above? That data is probably used as part of the seed material for generating the key somehow. Needless to say, if Microsoft is confident of their method of protecting private keys, they should publish the exact details, perhaps even this bit of source code. If they are not confident, then the public should be warned. Security through obscurity is not very reassuring.



How Can I Best Protect Private Keys?

To help protect your private keys, follow these best practices.

Physical Computer and Firmware

Only purchase laptops, tablets and other portable computers which come equipped with a Trusted Platform Module (TPM) in the motherboard. A TPM is not something which can be added later. There are significant security benefits to be had with a TPM, and the TPM only adds a couple dollars to the manufacturing price of the device.

Physically secure computers which contain sensitive private keys, such as CAs and the laptops of administrators, corporate executives, lawyers, R&D engineers, etc.

Use whole drive encryption with TPM integration, preferably with UEFI Secure Boot.

Install anti-virus software and configure it to update itself at least once per day. If you have UEFI Secure Boot enabled, only purchase AV software that is designed for Secure Boot early loading.

Physically secure Password Reset USB drives for local accounts, or don't make them. Train users to not store these USB devices in their computer cases if they do create them.

Operating System

Prefer the latest operating system over earlier ones (Windows Vista at a minimum), apply the latest Service Pack, and apply new patches as quickly as practical.

Do not use Windows 2000 or Windows XP. The "immortal" Windows XP died in April of 2014 when Microsoft stopped releasing any new security patches for it, and Windows 2000 was hopelessly insecure even before its official end of life.

Consider not using the hibernation feature unless whole drive encryption is also used. It is possible that key material is being written to the hiberfil.sys file.

Require reauthentication when resuming from hibernation, standby, sleep mode, and the screensaver. Screensaver should trigger after 30 minutes of idle time at a minimum.

Use SYSKEY.EXE to move the System Key off the computer using either the passphrase or floppy disk option.

Certification Authority Servers

Root and intermediate CAs should be installed as off-line stand-alone CAs. These CAs should not be configured as domain members. Store their hard drives in a safe when not in use. When in use, do not connect to the network; transport certificates and CRLs with flash drives instead.

Use a Hardware Security Module (HSM) for CAs which must be kept on-line.

Require authentication and encryption on "secure channels" with Group Policy since this is likely the RPC channel the Protected Storage service uses when encrypting users' Master keys. IPSec is a powerful way of accomplishing this.

Remember, too, that any member of the local Administrators group on a CA can export a copy of the CA's private key by default. Hence, regulate who can be a local administrator and strongly consider using a HSM.

Users

Use smart cards (or smart tokens) for user authentication and allow only smart card authentication in the properties of the users in AD. A smart card not only protects the private keys on the card itself, but, if only smart card logon is permitted for a user in AD, then all of the user's other private keys in their profile are protected as well. This is because the Master key used to protect the other private keys in the profile is secured via the public key for the smart card.

If you can't use physical or virtual smart cards, at least enforce a strong passphrase policy where passphrases must be at least 15 characters long. Strong passphrases improve private key security because the Master key is encrypted with the user's password and SID number.

On Windows 8/RT and later, private keys stored on the hard disk can be protected by the TPM instead of the user's logon passphrase. TPM security for private keys is much better than a passphrase, even when that passphrase is long and complex. The certificate

template in Active Directory will specify the TPM options for private key protection, including the use of an unlock PIN.

Because of the way Windows generates random numbers for encryption keys and salts, require users to log off of their computers every night instead of just locking their desktops. Each process has its own pseudo-random number generator, and one of the ways the generator is reseeded is when it first starts.

If practical, use the "Store Private Key with Strong Protection" option and require a password whenever it is accessed. It cannot be used with computer certificates, including CA private keys, or certificates used for IPSec or EFS though.

Accounts with sensitive private keys, such as recovery agents, should not have roaming profiles. There is also a Group Policy option to delete the locally cached copy of one's roaming profile at logoff. In the GPO, go to Computer Configuration > Policies > Administrative Templates > System > User Profiles > Delete Cached Copies of Roaming Profiles.

Assign NTFS permissions to each user's roaming profile stored on the fileserver so that only the profile's owner, System and Administrators can access it. The same goes for redirected Application Data folders. This should be the default already.

SYSKEY.EXE

SYSKEY.EXE became famous with Windows NT 4.0 Service Pack 3 as a utility to encrypt password hashes in the SAM database to defend against old L0phtCrack. In Windows 2000 and later, a system key is used by default.

SYSKEY.EXE creates a 128-bit RC4 key called the "System key". The System key is used to encrypt the following information:

- Master keys that are used to protect private keys.
- Protection keys for users' passwords in AD or the local SAM database.
- Protection keys for LSA secrets, such as service account passwords.
- The protection key for the local administrator account password that is used for recovery in safe mode.

SYSKEY.EXE can be executed from the Run line at any time to reconfigure how SYSKEY.EXE manages the System key. The System key can be stored or derived in one of the following ways:

- The System key can be stored locally with a "complex obscuring function" in the registry of the computer. This is the default. This option is good in that a computer can boot in the absence of a user to type a password or insert the System key floppy disk. Unattended reboots are useful after power spikes or brownouts. This option is bad in that hackers have long ago figured out how to reconstruct the original key.

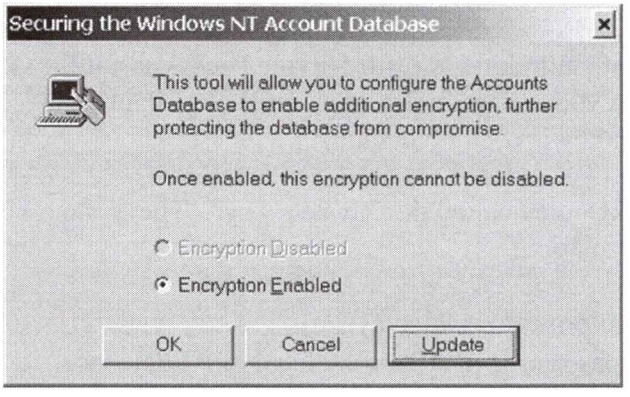
- The System key can be derived from a password a user must enter during reboot. The password can be up to 128 characters long, and is hashed with MD5 to create the 128-bit System key. This option is more secure, but a person must be present when the computer boots to enter the password. Without the password, the operating system will not boot. Make sure the password is long and random.

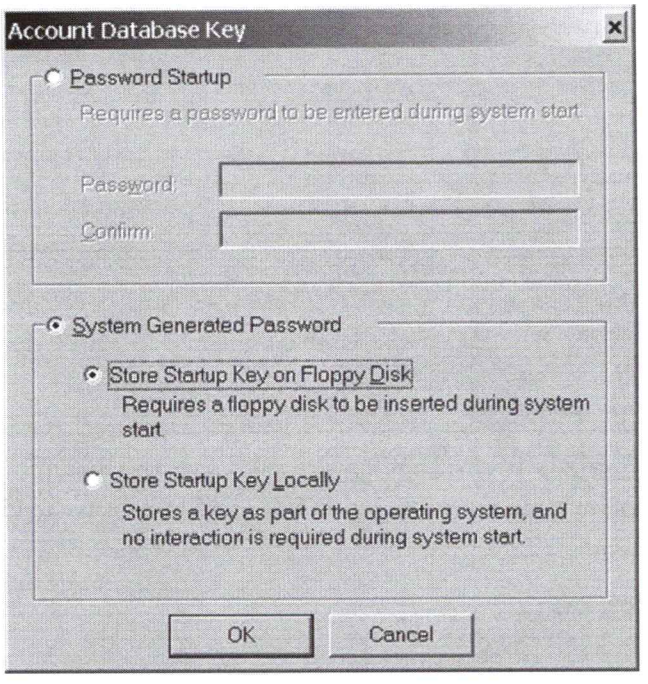
- The System key can also be stored on a floppy disk. This is also more secure than storing the System key locally, but the floppy disk must be in the drive when the computer boots. Hence, either the floppy is left in the computer at all times or a person must be there when the computer boots.

For a root or intermediate CA whose hard drives are locked in a safe, the password and floppy disk options may simply add another thing that can go wrong instead of adding another layer of security. For on-line and issuing CAs locked in secure rooms, however, the password method can add another layer of security and passwords can be entered remotely using a KVM-over-IP switch or iLO; but beware that every reboot will require the password to be entered, the password can be lost/forgotten, and passwords can be captured with keystroke loggers.

Try It Now!

To change your current System key configuration, run SYSKEY.EXE from the Run line.





Each computer can have its own separate SYSKEY.EXE configuration. Domain controllers do not replicate System keys, and each can be configured differently with different System Key options.

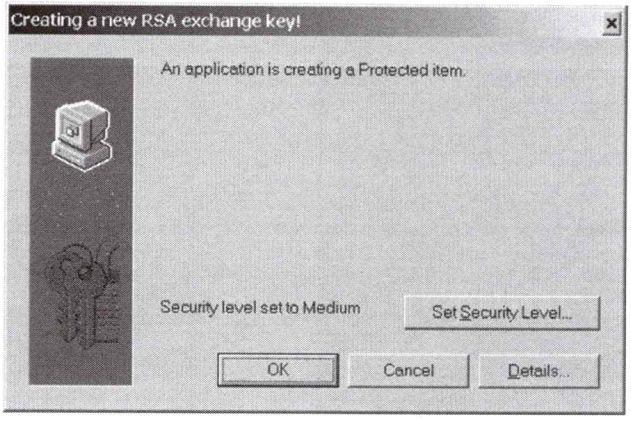
If a floppy disk is compromised, change the SYSKEY.EXE option to "Password Startup", reboot, then change the option back to "Store Startup Key on Floppy Disk". A new System Key will be generated. Keep a few of your old floppy disks, however, in case you need to restore from older tapes.

"Store Private Key With Strong Protection" Option

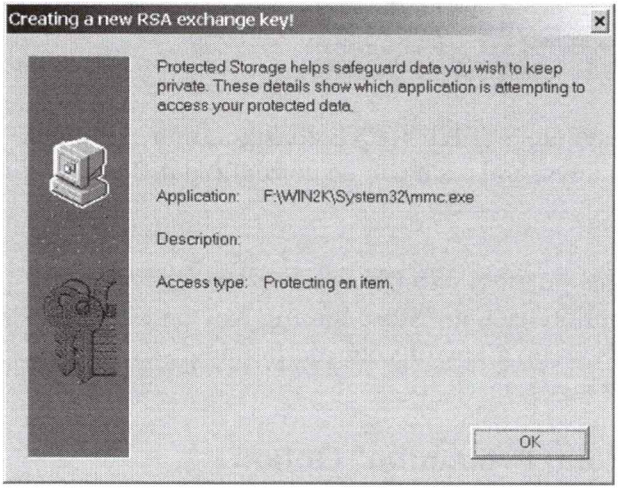
A feature of the Data Protection API (DPAPI) is the ability of the user to set an alert whenever a private key is being accessed. The alert will name the process attempting to access the item, and give the user the option to grant or block it.

The user also has the option to require a password whenever granting access to the item (32 characters maximum). These options are set on a per-key basis.

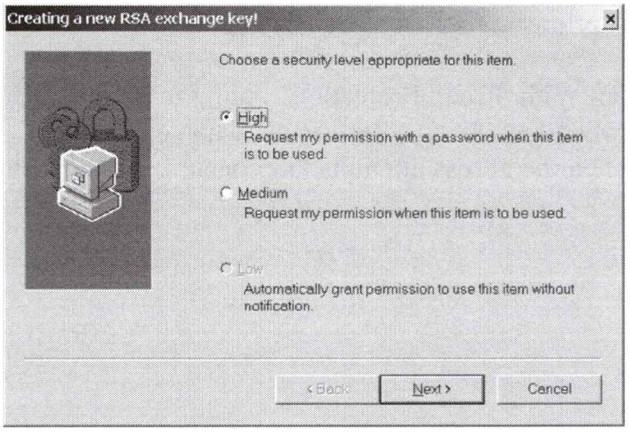
You will see the "'Store Private Key With Strong Protection" option in various places, e.g., under the Advanced options when requesting a certificate, when exporting a certificate and private key, etc. When the access attempts later occurs, the following dialog box will appear.



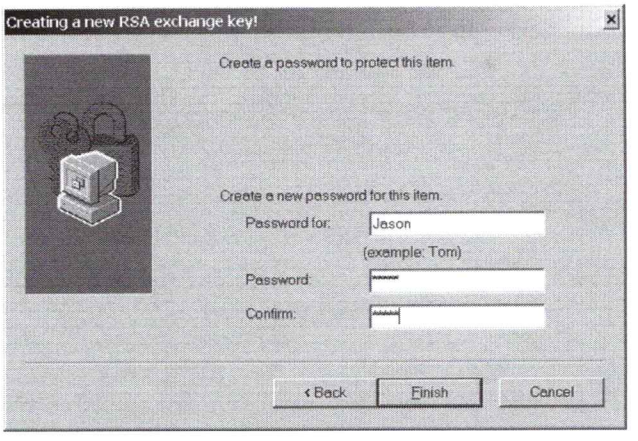
If you click OK, the access is permitted. If you click Details, you can see which process making the attempt. The next screenshot shows the dialog box displayed after clicking Details.



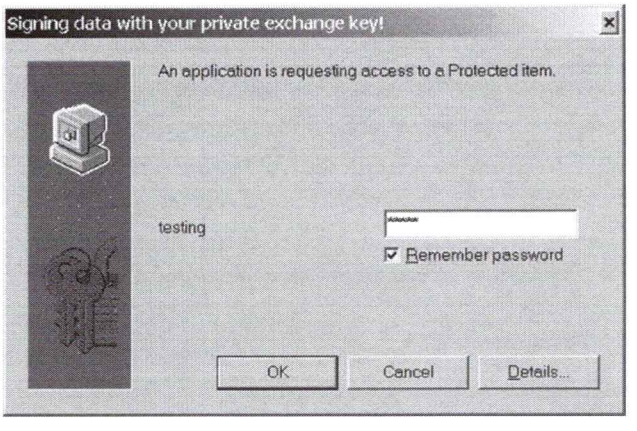
If instead you select "Set Security Level", the following dialog box appears. High security will require a password, Medium security simply prompts the user for yes/no permission, and Low security does not prompt the user at all.



If High security is selected, the user is prompted for a username and password. The password must be entered whenever the private key is used, but the username appears to do nothing and can be different from the username of the currently logged-on user.



After the password is entered, you will have the option to "Remember password" for the duration of the logon session so that it does not need to be entered again. (The password does not become an LSA Secret…I checked.)



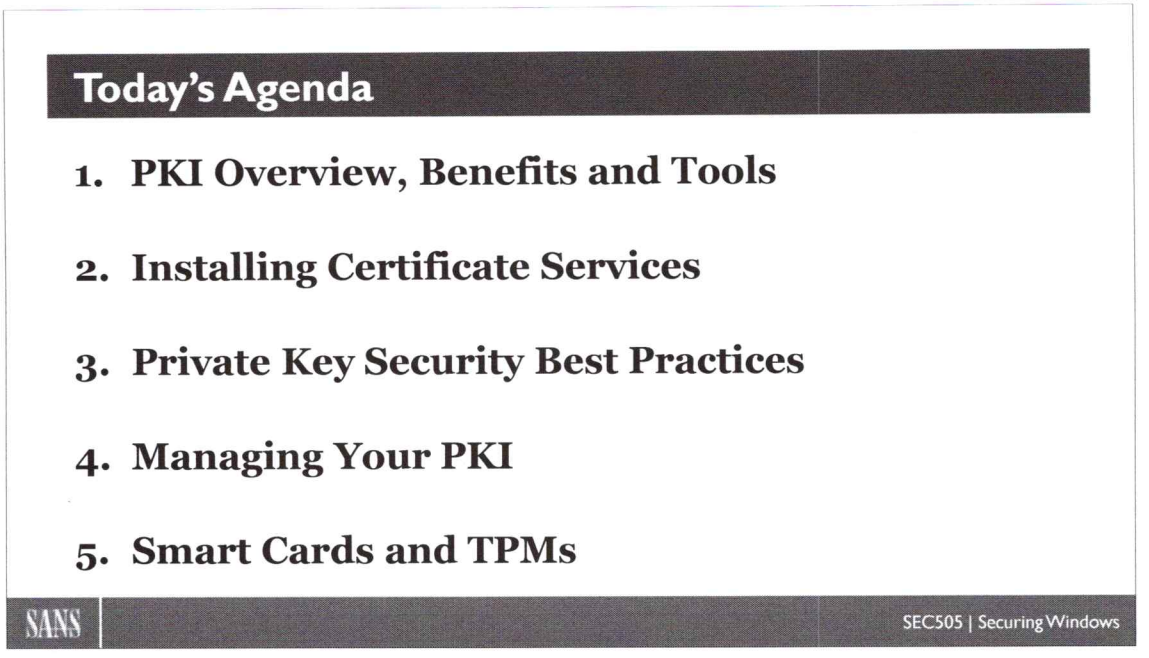
The password is used as additional keying material when creating the Session Key to encrypt and decrypt the private key (see the section in this manual on "Where Are Users' Private Keys Stored?"). The password entered is not stored anywhere on the drive. Without the password, the private key is lost forever because some of the data used to derive the Session Key is based on this password, and the Session Key encrypts/decrypts the private key.

Password protection is a feature of the Data Protection API and is used for other items besides private keys. Hence, you may see the above dialog boxes in other contexts.

"Strong Private Key Protection" with a password required is a good choice for the private keys of recovery agents, CTL signing certificates, and smart card enrollment agents (see

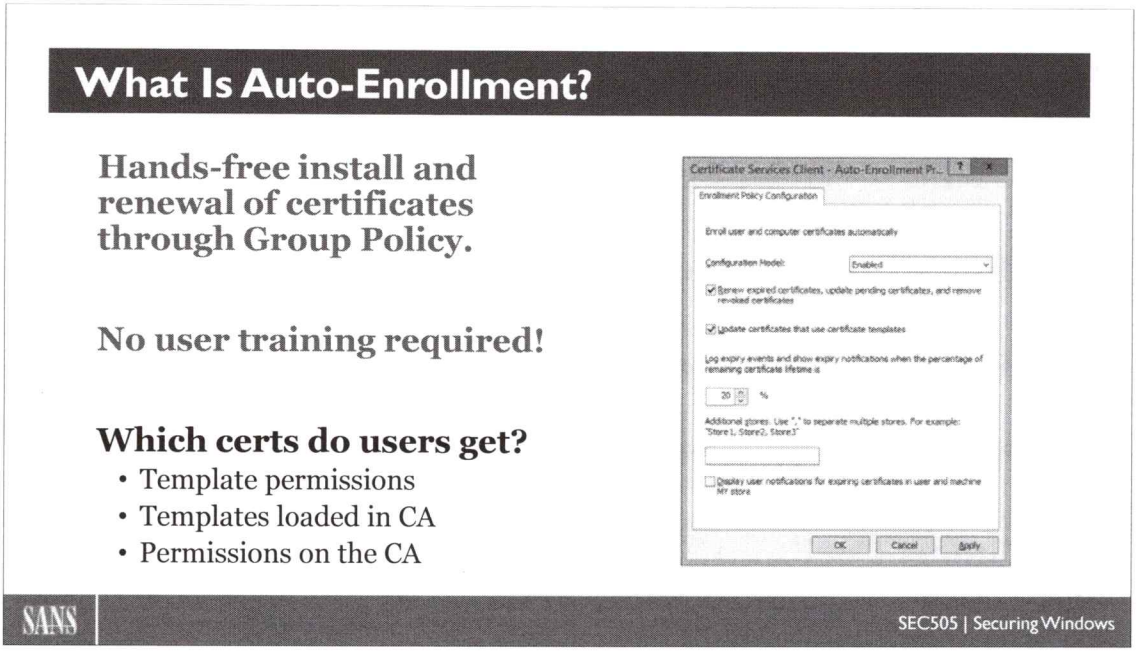
below). The feature is not available for computer certificates, including CA certificates. Nor can it be used with certificates for IPSec or EFS.

"Strong Private Key Protection" is not compatible with older programs, so if incompatibilities arise, you will need to re-export and then import the key pair again without Strong Protection enabled.



Today's Agenda

With the PKI in place and private keys locked down, the on-going management of the infrastructure is the next issue. In this section we will talk about such items as certificate auto-enrollment, importing certificates into Active Directory, controlling trusted root CA settings, and certificate revocation.



What Is Auto-Enrollment?

Windows computers can enroll for certificates from enterprise CAs automatically. Which certificates they get is determined through Group Policy. This is one of the best features of the Windows PKI and potentially offers the most cost savings over competing solutions.

Windows computers authenticate to the domain with their own computer accounts just like human users do. When the computer boots up, it logs onto the domain and downloads its Group Policy Objects. The booted computer will compare the certificate types it already possesses with the templates listed in its GPOs. If it does not possess a certificate for a corresponding template, that type of certificate will automatically be requested from an enterprise CA. (Auto-enrollment does not work with stand-alone CAs.) If the CA keeps requests pending until manually approved by an administrator, auto-enrollment is compatible with this and will retrieve the certificate if and when it is approved.

Expired, deleted or revoked certificates will also cause an auto-enrollment for new certificates. Each reboot, and every eight hours after that, the computer will check its certificates and the templates available. You can also force an immediate check with "CERTUTIL.EXE -pulse".

How To Configure

Configuration of auto-enrollment in Group Policy is relatively simple, once the necessary infrastructure is in place.

Try It Now!

To configure auto-enrollment settings in a GPO, navigate in the GPO to Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies. Next, right-click on "Certificate Services Client - Auto-Enrollment" on the right side > set Configuration to Enabled > check all boxes > OK.

You can adjust auto-enrollment behavior for both computer and user certificates. So, in the Try It Now just above, you can make the same change under User Configuration in the GPO, not just under Computer Configuration.

Windows 2000 Only: Computer Certificates Only

Windows 2000 can only request certificates related to the computer, and cannot auto-enroll for user certificates. The certificate templates available for computer auto-enrollment include:

- Computer
- Domain Controller
- Enrollment Agent (Computer)
- IPSec

Computers need certificates for a variety of reasons, e.g., Kerberos authentication with smart cards (DCs only), IPSec certificate authentication of peers and VPN clients, TLS/SSL encryption support on IIS and SQL Server, Active Directory SMTP replication connectors (DCs only), EAP-TLS certificate authentication to RRAS dial-up or VPN servers, etc.

The Enrollment Agent (Computer) is used with custom applications that require one computer to automatically request certificates on behalf of other computers (not other users). This usage will be rare.

Windows XP and Later: Both Computer and User Certificates

Windows XP and later support auto-enrollment for both computer and user certificates. User certificates include those for S/MIME encryption of e-mail, user authentication certificates for Internet Explorer and Connection Manager (dial-up and VPN connections), code signing, etc.

As with computer auto-enrollment, user certificates that have expired, been deleted or revoked are automatically replaced. Revoked certificates are automatically removed. It is also possible to replace existing certificates with new versions, such as when the new version is based on a customized template or when you want the certificates to come from a different CA (this is called "superseded certificate replacement").

Auto-enrollment of user certificates requires an enterprise CA running on Windows Enterprise Server 2003 or later (you can't use just regular Server), and the domain must be at Windows Server 2003 forest functional level or better, hence, all domain controllers

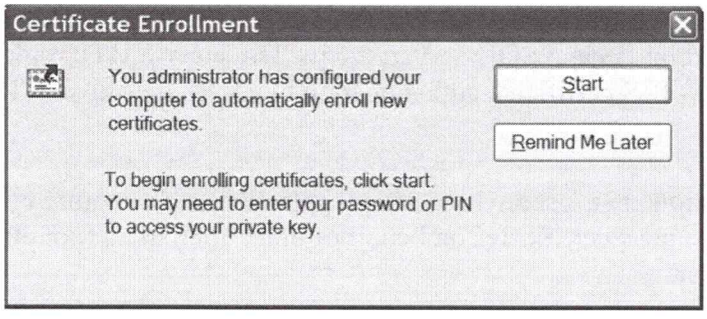
must be running Windows Server 2003 or later. The user's computer can be either Windows XP or later, but not Windows 2000.

Configuration of user certificate auto-enrollment is the same for computer certificates. However, the necessary version 2.0 templates will need to be created using the Certificate Templates snap-in in Windows Enterprise Server. This is done by right-clicking the old version 1.0 template for a user certificate type and selecting Duplicate Template. The version 2.0 duplicate can then be edited to permit auto-enrollment to define other options, such as automatic key archival. Version 3.0 templates are for use with Windows Vista/2008 or later.

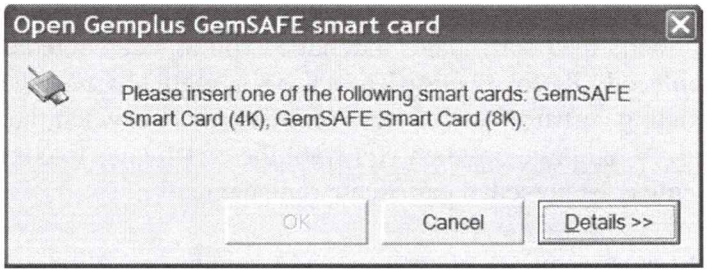
Auto-Enrollment of Smart Card Certificates

Auto-enrollment of certificates is transparent unless the enrollment process has been configured to require user interaction. Enrollment for smart card certificates always requires user interaction because a card must be inserted and the PIN entered.

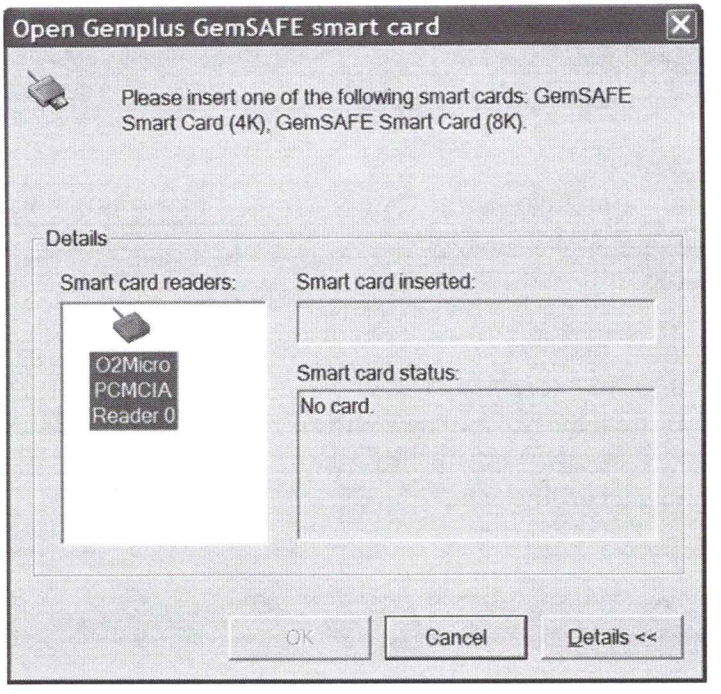
What does auto-enrollment for a smart card certificate look like to the user? After the user logs on, an icon that looks like a tiny certificate appears in the status area of the task bar next to the clock. When the user clicks the icon, a dialog box appears prompting the user to start the enrollment process or to be reminded to do so later.



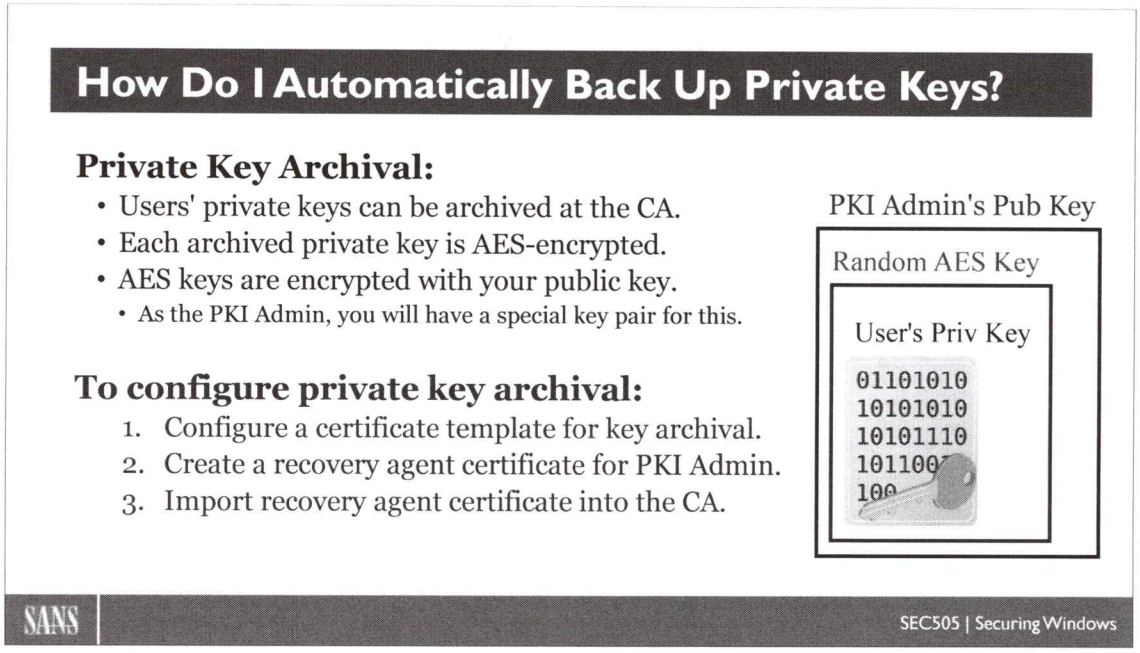
If the user clicks Start and the enrollment is for a smart card, another dialog box appears prompting the user to insert his or her card. If the wrong smart card driver appears first, the user clicks Cancel until the correct driver is listed. A single type of card should be given to users to avoid potential confusion.



If the user clicks the Details button, an expanded dialog box shows information about the card and the reader. This is useful for troubleshooting.



After the PIN is successfully entered, the enrollment process continues normally.



How Do I Automatically Back Up Private Keys?

One way to back up private keys is to enable roaming user profiles and regularly back up the profile server(s). If a private key is lost, you could restore the old profile with the key, but this requires roaming profiles for all users and is likely to run into problems during the recovery procedure. It is also possible, of course, to individually export keys to password-protected files, but this method is labor-intensive, prone to loss (if the passwords are forgotten or stolen), and not all private keys can be exported anyway.

With Windows Server 2003 Enterprise and later, on the other hand, private keys can be automatically backed up and archived on the CA itself. In the event a private key is lost, the key can be recovered from the certificate database. For the sake of non-repudiation, this policy only applies to encryption keys, not keys for authentication.

Requirements And Setup

To support automatic private key archival, you must have or do the following:

- The enterprise CA must be Windows Enterprise Server 2003 or later.
- The forest must be running at Windows Server 2003 forest functional level, hence, all domain controllers must be Windows Server 2003 or later.
- Only version 2.0 certificate templates or later can be used, and the checkbox labeled "Archive subject's encryption private key" must be checked on the Request Handling tab of the template's property sheet.

- One or more trusted administrators must acquire Key Recovery Agent certificates and import these into the configuration of the CA on the Recovery Agents tab of the CA's property sheet.

Archive Security

When a client enrolls for a certificate from a template which has been marked for archival, the resulting private key will be encrypted with a public key obtained from the CA. This public key is embedded in a certificate, which is validated and revocation-checked, then the user's private key encrypted with this public key and returned to the CA over an RPC DCOM channel. The CA will store that private key in its local database (not in AD) in an encrypted format, but not encrypted with the public key just used for transport over the network. Instead, the CA will generate a unique 3DES or AES key, depending on the version and configuration of the CA, to encrypt the user's private key, then that 3DES or AES key will be encrypted with the public key of one or more key recovery agent administrators and stored with the user's private key in the CA database. Only the recovery agent(s) can decrypt the 3DES/AES key associated with the private key, hence, only the recovery agent(s) can get to the private key. Each private key has its own unique 3DES or AES protection key, hence, if there are 1000 backed up user keys, there will be 1000 different 3DES or AES protection keys.

The CA can be configured with one or more recovery agent administrator public keys. This is configured by the CA administrator in the properties of the CA on the Recovery Agents tab. When multiple recovery agent keys are available, one or more will be randomly selected to encrypt the same number of copies of the 3DES/AES protection keys; for example, if 10 recovery agent public keys are available, and the CA is told to use 5 of the 10, then 5 will be selected at random from the available 10, then 5 copies of the 3DES/AES key will be made, each 3DES/AES key encrypted separately with one of the 5 randomly-selected recovery keys, then all the copies of the 3DES/AES keys will be stored. Because each copy of the 3DES/AES protection key is encrypted separately, only one of the 5 randomly-selected recovery agents is required to recover the user's private key. Which agent certificates are available, and how many of these public keys should be randomly selected, is all configurable by the CA administrator.

Try It Now!

To configure private key recovery agent certificates on a CA, first obtain one or more certificates based on the Key Recovery Agent template (not to be confused with the EFS Recovery Agent template) , then go to the Properties of the CA > Recovery Agents tab > select the "Archive the key" radio button > Add button > select a recovery certificate > OK > OK. Repeat as necessary to add the desired number of recover agent certificates to be made available; a minimum of one is required. On that same tab, enter the number of recovery agent keys that will be randomly selected out of the available keys that were added. You must select a number between one and the total number of agent certificates currently available. If in doubt, set it to the current number of agent certificates added. Click OK, then restart the CA service when prompted.

Make sure to audit all key recoveries on the Auditing tab of the CA's property sheet.

Keep in mind that the certificate database on the CA now contains copies of private keys. This server should be secured along with its backup media.

How To Recover A Key

A private key is recovered with CERTUTIL.EXE, a command-line tool that is installed with Certificate Services. The following are the minimal steps, but there are alternative ways of doing it, and CERTUTIL.EXE has other command-line switches you may wish to investigate too.

1. In the Certificate Services snap-in, identify the certificate in the list of issued certificates whose private key needs to be recovered. In the properties of that certificate, go to the Details tab, and copy the thumbprint of the certificate to the clipboard (Ctrl-C).

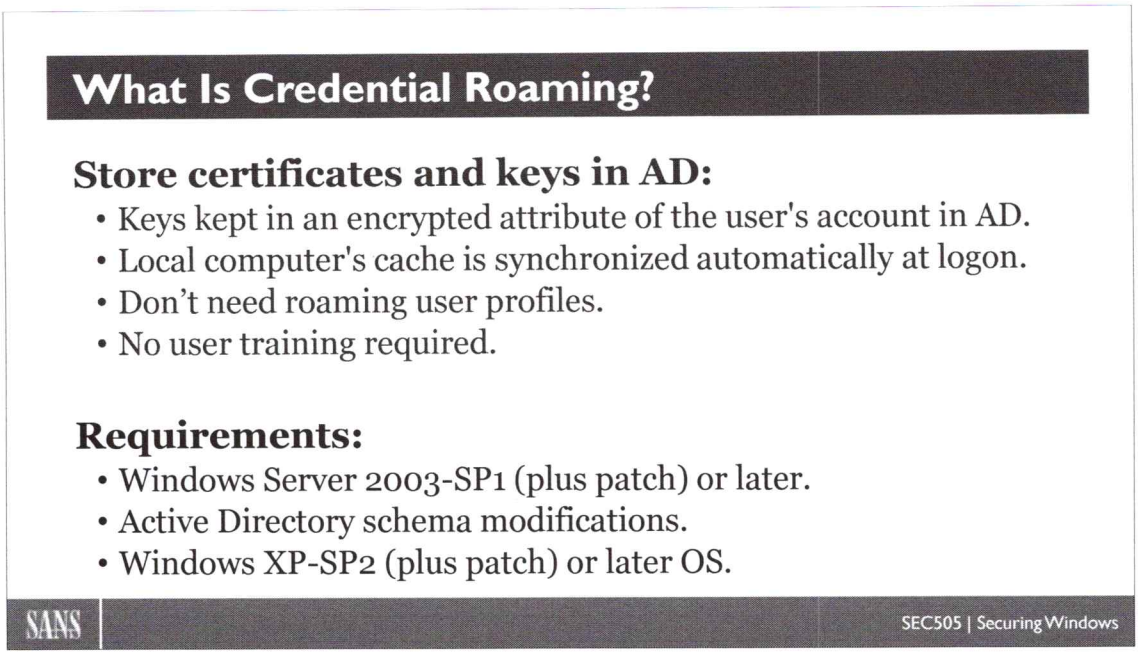
2. In a command-prompt window, a CA administrator should execute: `certutil.exe -getkey "79 9e 65 49 57 86 b5 b2 f1 97 66 e9 b5 f3 d3 45 93 8a e3 1a" outputfile.txt`, where the long number is the thumbprint of the certificate whose private key needs recovery. The extracted data will be stored in outputfile.txt in PKCS#7 format, which includes the encrypted key and a list of exactly which randomly-selected recovery agent certificates were used to encrypt the copies of the user's private key, which is in that output.txt file as well.

3. One of the recover agent administrators listed in the above output.txt file must log on and import their agent certificate and private key. In a command shell, that administrator should execute: `certutil.exe -recoverkey outputfile.txt keyfile.pfx`. The admin will be prompted for a password to encrypt the resulting keyfile.pfx. Use a long and complex password.

4. Delete the recovery agent certificate and private key imported into the computer. Do this before logging off. Because file system traces will be left behind, it's best to use a dedicated computer for this purpose which is physically secured, only connected to the network when necessary, and powered down when not in use.

5. While logged on at the user's computer as the user, delete the certificate whose private key had been lost, if present, then import the certificate and private key from the above keyfile.pfx file. Permanently delete the keyfile.pfx, make sure it did not go into the Recycle Bin, and consider "wiping" the file's clusters from the hard drive. Log off to save the profile and trigger credential roaming synchronization. The key is now recovered.

What is "credential roaming synchronization"? Could this be used to recover the user's key instead?



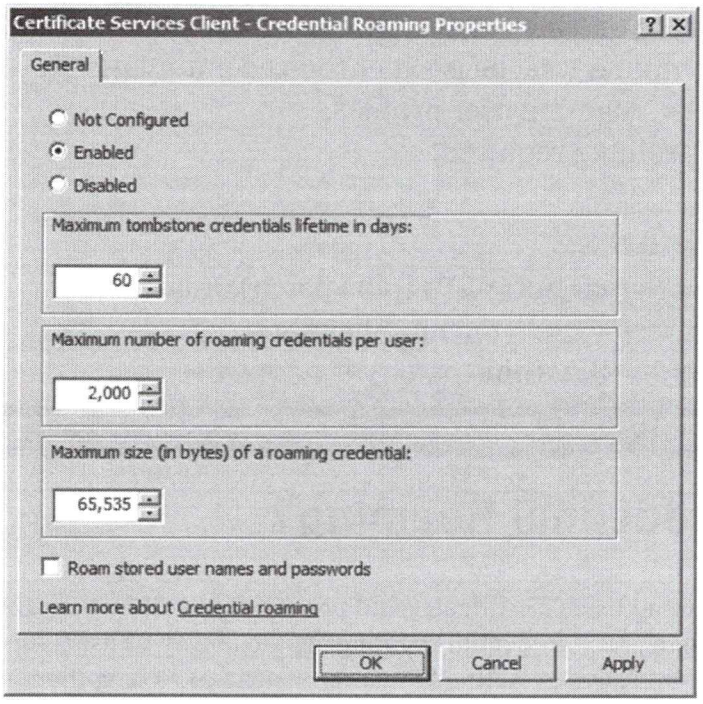
What Is Credential Roaming?

For the sake of making users' certificates and private keys follow their owners around from machine to machine, "credential roaming" is the ability of users to store their certificates and private keys in Active Directory instead of storing them in their roaming user profiles.

The certificates and keys stored in AD are kept synchronized with the local certificates and keys at every computer where the user logs on locally. If there is a difference between the items in AD and the local items, the newer item is uploaded/downloaded to overwrite the older one. This synchronization check is triggered every time a user logs on, locks or unlocks the computer, Group Policy is refreshed, or anytime a change occurs to the local collection of certificates and private keys. If a user deletes a certificate or private key locally, this change is updated in AD, and copies of the deleted item(s) can be automatically deleted at the other computers where the user subsequently logs on. Synchronization is performed before autoenrollment to ensure that unnecessary duplicates are not created. Renewed certificates are automatically synchronized. Expired certificates are archived indefinitely both in AD and in the local profile.

Domain accounts using credential roaming will have a multi-valued attribute named "ms-PKI-DPAPIMasterKey" which will contain all of the user's master key files (used to encrypt private keys), the name of the preferred master key to use, an update timestamp (ms-PKI-RoamingTimeStamp), and another multi-valued attribute (ms-PKI-AccountCredentials) to hold one or more encrypted copies of private keys, certificates, certificate requests, and, on Vista and later, even stored usernames and passwords. *How* are they encrypted exactly? It's not published.

Credential roaming options are configured on clients through Group Policy, which requires a new ADM template on Windows Server 2003, but the template is built into Server 2008 and later. In Server 2008 and later, you'll find the options under User Configuration > Policies > Windows Settings > Security Settings > Public Key Policies.



Currently, the best documentation for troubleshooting credential roaming is the following: http://technet.microsoft.com/en-us/library/cc700848.aspx

If your domain controllers are Server 2003, you'll need to contact Microsoft Customer Support Services (MCSS) to get the scripts necessary to configure credential roaming. On Server 2008 and later domain controllers, the feature is built in, but not enabled by default.

Requirements

To enable credential roaming, you must have or perform the following:

- All controllers running at least Windows Server 2003-SP1 with the update described in KB907247, or Windows Server 2003-SP2, or later, plus the schema modifications. Server 2008 and later are already capable.

- The necessary schema modifications on pre-Server 2008 domains are described in an article entitled "Configuring and Troubleshooting Certificate Services Client-Credential Roaming" last seen at http://technet.microsoft.com/en-us/library/cc700848.aspx. They are also mentioned in KB907247. (You can also search on the names of the files necessary to try to find more documentation:

credroam.ldf, permupdate.ldf, credroam.adm, credroamperms.bat, and krdeploy.cmd.)

- Clients running Windows XP-SP2 or Windows Server 2003-SP1 or later with the update described in KB907247 applied. Windows XP-SP3/2003-SP2 and later versions of Windows already have the update.

- Forest functionality level must be at Windows 2000 native mode or better.

- If domain controllers are Server 2008 or later, the client must be able to communicate with at least one controller which is not a Read-Only Domain Controller (RODC).

- Users who are migrated from one domain to another must first export all of their certificates and private keys, migrate to the new domain, then import. Migration and credential roaming cannot be used simultaneously.

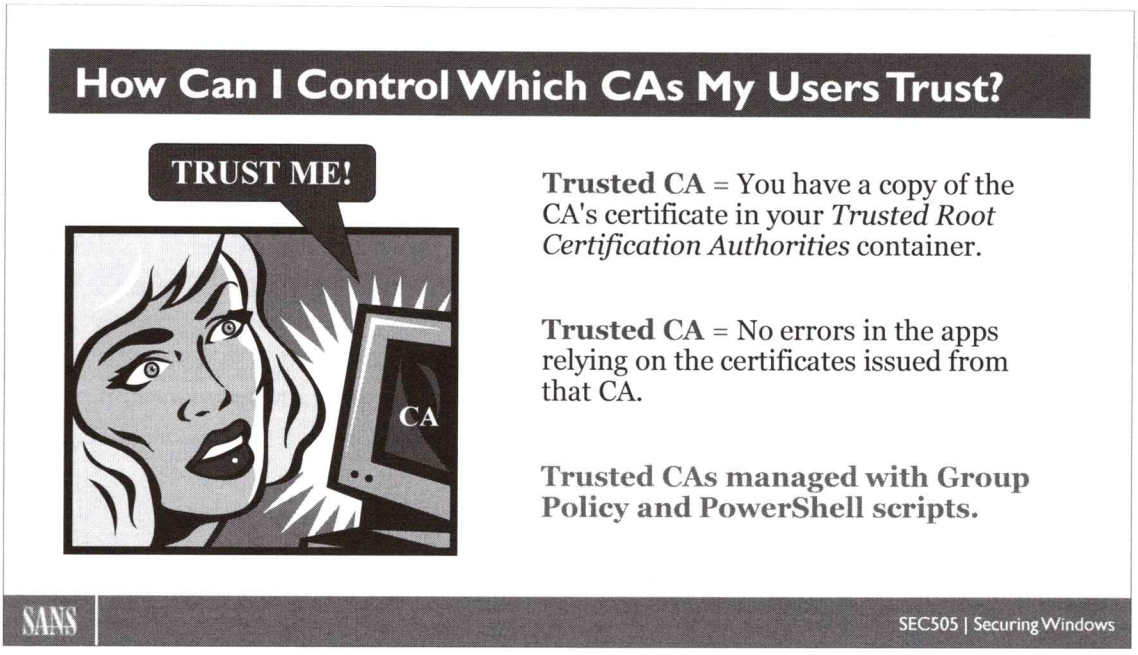
- Roaming user profiles either disabled entirely or with the necessary directories excluded from roaming (see the articles mentioned above).

- Cannot currently be used for user accounts used as service accounts, e.g., SQL Server service accounts. A future patch or Service Pack has been promised to correct this shortcoming (don't hold your breath).

Does It Really Work?

In this author's experience, amazingly, *Yes!* The user sees nothing different on their desktops, there's no user training required, and the schema modifications caused no problems. This author has seen the feature deployed on a network of over 20 thousand users using Windows XP workstations and Server 2003 domain controllers.

One tip, though, is to remember an option found in the properties of a certificate template named "Do not automatically reenroll if a duplicate certificate already exists in Active Directory" (it's found on the General tab). You'll almost certainly want this box checked on templates intended for users. This option goes with credential roaming very nicely to help ensure that each user gets only one certificate of that type and that that certificate (and private key) follows the user around from computer to computer. This is very important, for example, for S/MIME e-mail certificates and private keys.



How Can I Control Which CAs My Users Trust?

If a client trusts a CA, then the client will trust the certificates issued by the CA. Trusting a certificate means believing the credentials in it are correct and that the subject is the sole owner of the certificate's corresponding private key. Controlling which CAs your users and computers trust is important because attackers will attempt to trick users/computers into trusting CAs controlled by the attackers, because malware might install new CA certificates, and because you might not trust some CAs.

If an application does not use CryptoAPI/CNG, then it will most likely have its own list of trusted CAs or be configured to use third-party PKI services on the network. Mozilla Firefox, for example, comes with its own list of trusted CAs.

If an application uses CryptoAPI/CNG, then that application will trust any CA listed in the "Trusted Root Certification Authorities" container of the Certificate Store of the user or computer running the application. The application will also trust the CA certificates found in any Certificate Trust List (CTL) in the Enterprise Trust container.

> **Note:** The "Intermediate Certification Authorities" container does not imply trust. It exists only to cache CA certificates to optimize path building.

The "Trusted Root Certification Authorities" Container

The "Trusted Root Certification Authorities" container can be seen in the Certificates snap-in. By default, the container will already have many certificates from popular CAs such as VeriSign, Thawte, Equifax, etc. These CA certificates come bundled with the operating system so that Internet Explorer and other applications will trust them even if no PKI is deployed on the network or at home.

Additional CA certificates can be added to the Trusted Root container using Group Policy. Each site, domain and OU could have different trusted CAs added depending on security requirements. Computers can have additional root CAs added, but not users, when using Group Policy. CTLs should be enabled for per-user configuration of trusted CAs instead.

> ***Try It Now!***
>
> To add CA certificates to the "Trusted Root Certification Authorities" container, open the GPO for the site, domain or OU desired > Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > right-click on Trusted Root Certification Authorities > All Tasks > Import. You must have an exported copy of the CA's certificate.

Deleting trusted CA certificates is more difficult. There is no option to delete CA certificates with Group Policy except to write custom scripts. CA certificates can be deleted, i.e., no longer trusted, with the Certificates snap-in, but this manual method does not scale beyond a handful of systems. However, you can script the removal of unwanted certificates and push such scripts out through Group Policy. See KB293781 for a list of root CA certificates that should not be removed though.

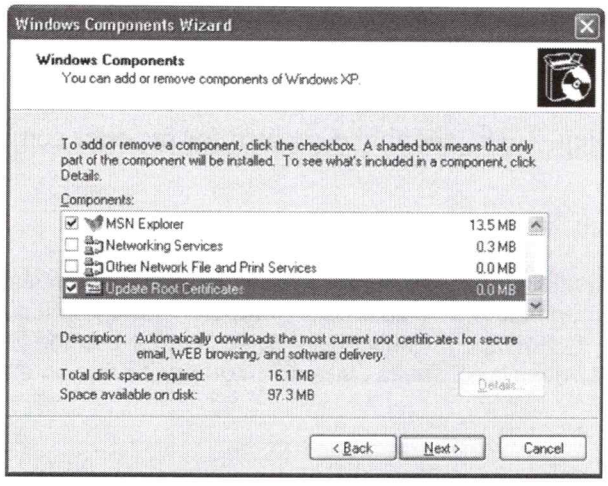
> **Important!** It *is* possible to import a CA certificate directly into the registry with REGEDIT.EXE and add that certificate to the Trusted Root CA Store. It is easy for malware running with administrative privileges to install fake root CAs.

Windows Updates Your List of Trusted Roots Automatically

If a Vista or later user receives a certificate that was issued by a CA which is not currently trusted, such as when the user is browsing a website using SSL, then, by default, the computer will connect to Microsoft's Windows Update website and check to see if that CA has been added to Microsoft's list of trusted CAs (KB931125). If it has been added, then the user's computer will automatically download that CA's root certificate using HTTP on TCP/80 and add it to the local Trusted Root Certification Authorities store. This will make the computer --and the user's applications-- trust the certificate received and its CA. The users sees nothing, there are no prompts, the process is invisible.

When new trusted root CA certificates are added or removed, a variety of events are added to the Application event log with a source property of CAPI2 (filter on Source = CAPI2), such as event ID numbers 4097, 4100, 4107, 4108, 4109, 4111 and 4112 for both successful and failed update actions. Sometimes certificates are added to the disallowed list too, such as when the private key of the CA has been compromised. At the time of this writing, there are hundreds of trusted CAs on Microsoft's list.

For Windows XP, there is an optional "Update Root Certificates" component which can be downloaded (KB931125) to update root CAs too. To check if it is currently installed, or to remove it if you don't want it, open the Control Panel in Windows XP > Add/Remove Programs > Windows Components > verify or uncheck the "Update Root Certificates" component.



Do you really want to trust hundreds of Microsoft-managed CAs? The good thing about letting Microsoft manage the list is that bad CAs will be disallowed quickly without requiring user training or intervention. Also, as popular CAs fade over time and new issuers grow, then new CA certificates can be added without each company having to do all of its own validation and research into each new proposed CA since Microsoft has its own validation program. Microsoft is motivated to validate new CAs before adding them to the list because of the embarrassment which would come from making a mistake. In general, the benefits are similar to the "walled garden" of the Apple App Store and the Microsoft Windows Store, except that you can always explicitly disallow any non-Microsoft root CA you wish even if it's on Microsoft's list of trustworthy CAs. On the other hand, we are allowing Microsoft to manage extremely an important security setting on our machines, and Microsoft could get hacked or make a mistake about who to trust.

In the end, your organization will have to decide whether to outsource most of the root CA list management to Microsoft (the default) or to do it all yourself. As a practical matter, 99% of organizations will simply not devote the time to doing root CA management properly, resulting in mistakes and user complaints about SSL and S/MIME error messages. If you're not sure, leave it at the default, but be prepared to push out additional CA certificates through Group Policy and to disallow CAs in an emergency using a CTL (see below).

To turn off root CA updates through Group Policy, open the GPO > Computer Configuration > Policies > Administrative Templates > System > Internet Communication Management > Internet Communication Settings > Turn off Automatic Certificate Root Update.

The "Enterprise Trust" Container and CTLs

A user's or computer's Certificate Store also contains an Enterprise Trust container which can hold zero or more Certificate Trust Lists (CTLs). A CTL specifies a CA certificate that should be trusted *and the purposes* for which its issued certificates may be used. By default, the Enterprise Trust container is empty. New CTLs can be added to the

Enterprise Trust container for users or computers with Group Policy. Edit the CTL to remove previously CTL-trusted CAs.

In order to create a CTL, an administrator must have a certificate that supports the CTL Signing purpose. The Administrator certificate template includes this purpose.

> ***Try It Now!***
>
> To trust a CA using a CTL, open the desired site, domain or OU Group Policy Object. The CTL can be added either the user's or the computer's configuration. In either case, open Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > right-click Enterprise Trust > New > Certificate Trust List. This will launch a Wizard to complete the operation. (You can also import a CTL by selecting All Tasks > Import instead.) The Wizard will ask you how long the CTL should remain valid and the permitted purpose(s) of the CA certificates added.
>
> **Note:** The CTLs assigned to users are not shared with services or the computer account. CTLs assigned to the computer are shared with users, services and the computer account. A user's applications can trust additional CAs than the user's computer or services, but any CA trusted by the computer will also be trusted by the user.

CA Certificate Purposes

As mentioned just above, a CTL is also used to control the purposes for which the certificates issued by the CA can be used. For example, you might trust the certificates from a certain CA only for the sake of IPSec and time-stamping, but nothing else. This gives the PKI administrator a more fine-grained control over how foreign certificates will be trusted.

These acceptable purposes are assigned when the CTL is first created. They can be edited in the CTL in Group Policy later if desired by right-clicking the CTL in the Group Policy Object > All Tasks > Edit.

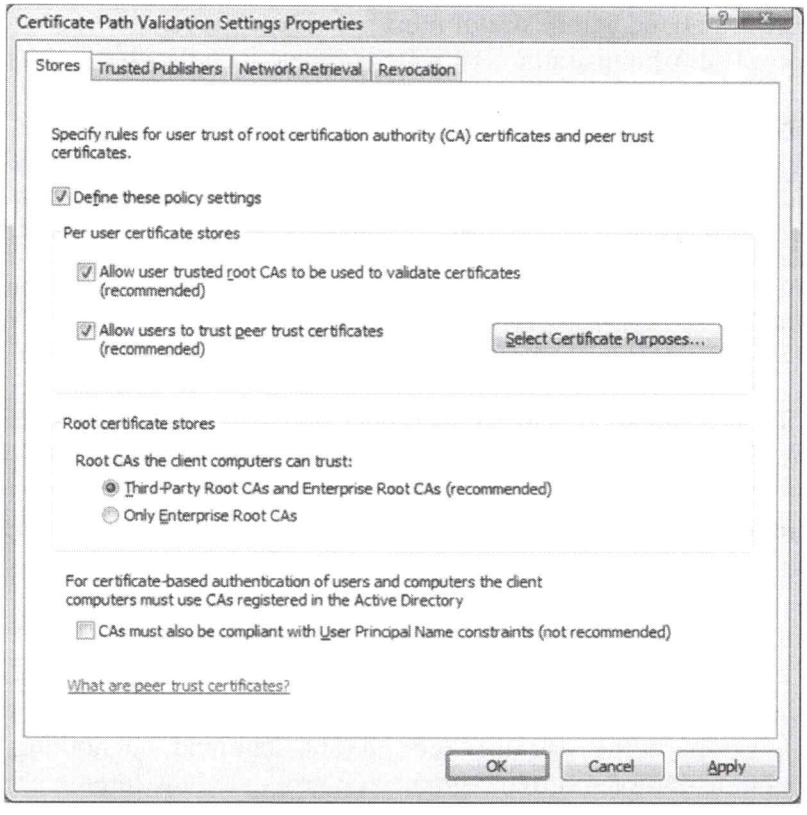
A user can also modify acceptable purposes of certificates issued from a CA on their own systems.

Certificate Path Validation Settings (2008/Vista and Later)

With the domain functionality level at Server 2008 or better, and with Vista or later clients, you can specify additional settings to more tightly and flexibly control certificate validation behavior in the client. On a Server 2008 or later domain controller, you can find these settings in a GPO by navigating to Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > properties of "Certificate Path Validation Settings". Here you can control such things as:

- Whether users can choose which new root CAs to trust.
- Whether users can trust other user certificates for e-mail or user auth.
- Whether computers will trust third-party CAs at all.
- Which user groups can choose to trust code signing certificates.

- Whether Microsoft can update the list of third-party trusted CAs.
- CRL vs. OCSP preferences and time-outs.



These settings allow you to more precisely control under what circumstances users may choose to trust a certificate or CA. If you allow users to do anything, it's too much. If users cannot make any choices, there will be complaints and problems. Hopefully a middle ground can be reached which satisfies security requirements, user preferences, and the needs of the over-worked help desk.

Danger

Because a computer's list of trusted root CAs is so important, it also makes it an attractive target for hackers and malware. If a victim machine can be tricked into trusting a fake Microsoft or VeriSign code-signing or SSL certificate, it opens up the door for man-in-the-middle and other spoofing attacks. If a computer is infected with malware running with elevated privileges, the malware can inject a fake root CA certificate. Hence, it's important to periodically audit the root CAs trusted by computers in the organization, especially all the client devices. Fortunately, this can scripted in PowerShell.

On Your Computer

This exercise has two parts.

Audit Trusted Root CA List

Please switch to the C:\SANS\Day3-PKI\AuditRootCAs folder:

```
cd C:\SANS\Day3-PKI\AuditRootCAs
```

See the README.txt file, which provides some suggestions on where to obtain hash lists of trustworthy root CA certificates, such as from Microsoft, Google or Apple:

```
notepad .\README.txt
```

Use a script to search for root CA certificates trusted by the user or computer, but which are not on the reference list of trustworthy certificates (the X's in the example represent the time stamp in the file name, which will be different on your computer, not X's):

Note: The following is one command that wraps to two lines.

```
.\Audit-TrustedRootCA.ps1 -PathToReferenceList
  .\Microsoft-Trusted-Root-CA-Certs-XX.XXX.XXXX.txt -OutputPath .
```

In real life, the output path would be a shared folder available to all hosts on the network, but this example just uses the present working directory (a single dot, ".", is an alias for

the present working directory). The output is a file with a name similar to "ComputerName+Administrator+635574443418152132.csv", which includes the name of the local computer, the name of the user which ran the script, and a timestamp number to indicate when the file was created (useful for sorting too).

Look for a file named after your computer in the present directory and open it:

```
dir

ise computername+username+ticksnumber.csv
```

If the file is nearly empty, it means all of the root CAs trusted by the computer are on the list of trustworthy CAs. If the file contains any root CA names and hashes, it means these trusted root CAs are not on the list of trustworthy CAs.

In real life, the script would be configured as a logon script or scheduled job through Group Policy, with the output saved to a Distributed File System (DFS) shared folder. When you review the folder, look for files with a size greater than zero.

Remove Unwanted Certificates

Please go back up to the C:\SANS\Day3-PKI folder:

```
cd C:\SANS\Day3-PKI
```

Run the following VBScript script to add the Belgacom certificate as a trusted CA:

```
cscript.exe .\Inject_Root_CA_Cert.vbs
```

In your MMC console for today's course, go to the "Certificates - Current User" snap-in and navigate down to the "Trusted Root Certification Authorities" certificates container under Local Computer > Certificates. Notice that you have a trusted root CA named "Belgacom E-Trust Primary CA".

Double-click the Belgacom certificate > Details tab > see the Thumbprint field at the bottom. This field shows a hash of the certificate. OK to close.

In PowerShell, run the following script to delete the Belgacom root CA certificate:

```
.\Remove-TrustedRootCA.ps1
```

Back in the MMC console, right-click the "Trusted Root Certification Authorities" certificates container > Refresh. Notice that the Belgacom certificate is gone.

View the source code of the script and see the $BadCerts variable which contains a list of all the hashes of the CA certificates the script should delete:

```
ise .\Remove-TrustedRootCA.ps1
```

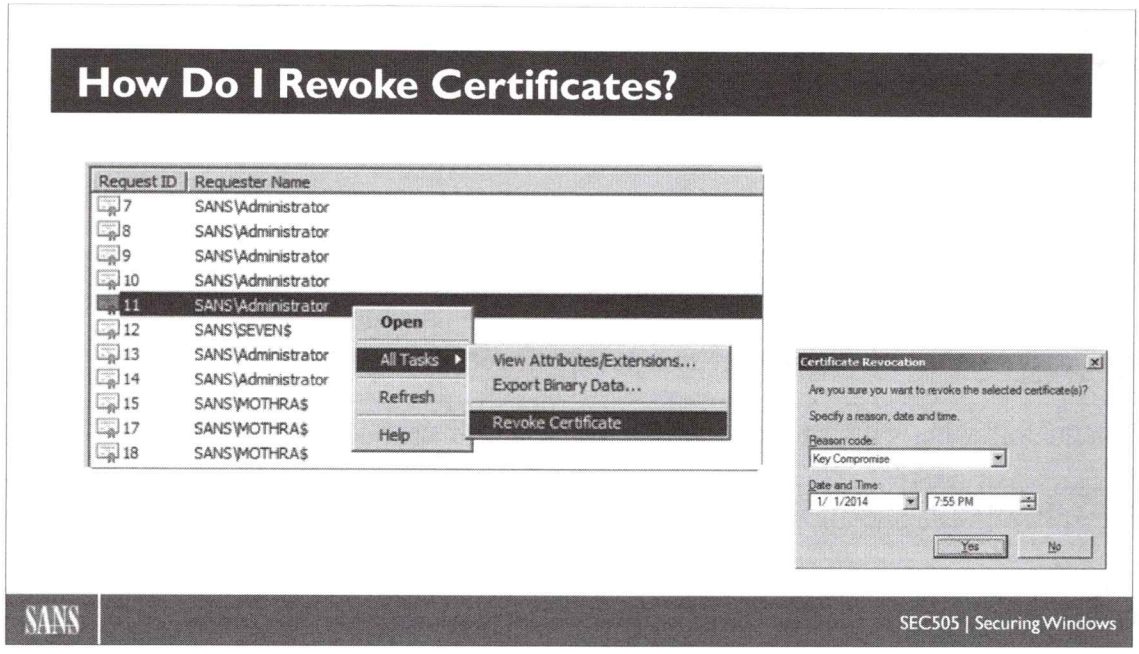
In real life, the $BadCerts variable would be edited, perhaps after discovering fake CA certificates injected by malware in your environment, and the script would be pushed out through Group Policy as a startup script or scheduled task.

How easy would it be for malware to inject a fake root CA certificate?

Run the following VBScript script to add the Belgacom certificate back again:

```
cscript.exe .\Inject_Root_CA_Cert.vbs
```

It's very common for document or browser malware to include VBScript, JavaScript, PowerShell or Office Macros, all of which could inject fake root CA certificates.



How Do I Revoke Certificates?

When the private key of a certificate has been compromised, or when you otherwise no longer wish a certificate to be trusted, that certificate should be revoked. When a certificate is revoked it is added to a Certificate Revocation List (CRL). Clients download an updated CRL periodically and check their locally cached copy of the CRL whenever Windows is used to verify a certificate's validity.

Certificates are revoked at the CA using the Certification Authority snap-in. Regular users cannot revoke certificates, including their own.

> ***Try It Now!***
>
> To revoke a certificate, open the Certification Authority snap-in and attach to the CA which issued the certificate > Issued Certificates > right-click the desired certificate > All Tasks > Revoke Certificate. You will be prompted to given an optional explanation for the revocation. Once revoked, the certificate will be moved to the Revoked Certificates container.
>
> **Note:** Once a certificate has been revoked, it is not possible to un-revoke it unless you select "Certificate Hold" as the revocation reason.

To open the Certification Authority MMC snap-in so that it will show extra information about your CRLs, open the console with the "/e" switch:

```
certsrv.msc /e
```

Where Are CRLs Obtained?

The certificate of a CA will contain a field called the "CRL Distribution Point" (CDP). The CDP field lists the location(s) where the CA's CRLs are published and made available to users. The locations are usually defined with URLs for access using HTTP or LDAP. Windows domain clients, for example, can use LDAP to download a CRL from Active Directory or HTTP to download from your web servers.

You'll want to make sure your CDP information is correct before issuing certificates to your users and computers. The CDP information defined in the properties of the CA will be placed into each certificate the CA issues.

> ***Try It Now!***
>
> To change the CDP of an enterprise CA, go to the properties of your CA in the Certification Authority snap-in > Extensions tab > add/remove CDP locations from the list. (Search "CDP" in Windows Help for the syntax of CDP URL locations.)
>
> **Note:** The above describes how Enterprise CAs publish CRLs. Stand-alone CAs simply "publish" their CRLs by creating the CRL file in a local folder (%SystemRoot%\System32\Certsrv\Certenroll). These stand-alone CRLs are available through the Certificate Services Website. It is up to the administrator to distribute the CRL files to other locations if desired; for example, the CRL could be manually imported into Active Directory with the CERTUTIL.EXE tool.

If you have multiple domain controllers in each of your sites, then LDAP availability of CRLs is automatically fault tolerant and load-balanced for your internal and VPN clients. If you wish to use HTTP to obtain CRLs, such as for roaming clients outside the LAN, then it is up to you to 1) choose a FQDN in the URL that DNS can map to the appropriate farm of web servers, 2) configure DNS to do the desired geo-aware or load-balanced mapping as desired, and 3) set up two or more web servers in each farm.

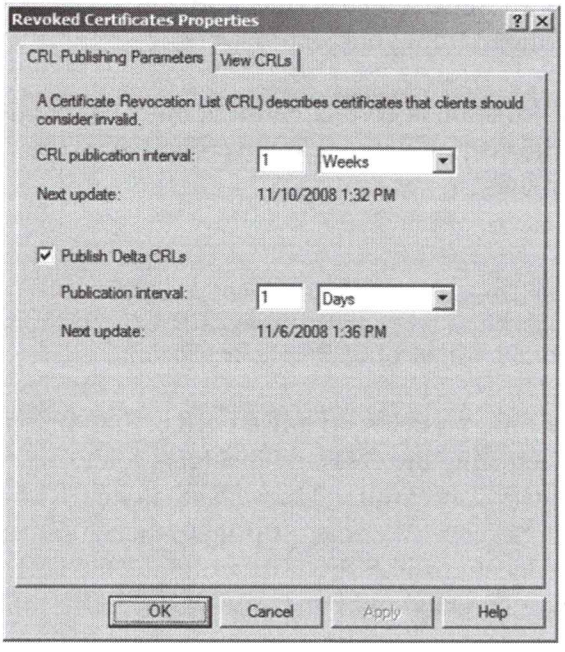
Also, if you wish to change the CDP field in the certificate of the CA itself (not just in the certificates the CA issues), then you'll need to create/modify the CAPOLICY.INF file described earlier in this manual, then renew your CA certificate afterwards. The CAPOLICY.INF file was discussed in the section on how to install certificate services. A good reference for the syntax of that file is Brian Komar's book, *Windows Sever PKI And Certificate Security* (Microsoft Press).

Can I Change How Often An Updated CRL Is Published?

By default, an updated list of revoked certificates is published to the CA's CDP locations once per week. This can be changed. When a new CRL is published, all new revoked certificates are added to the CRL and all revoked certificates on the list whose validity period have expired will be removed from the CRL. Time-expired certificates are removed from the CRL to keep the CRL small and because clients should automatically not trust time-expired certificates (being on the CRL is redundant).

Try It Now!

To change the default publication period for updated CRLs, open the Certification Authority snap-in > right-click on Revoked Certificates > Properties > CRL Publishing Parameters. Periods are measured in hours, days, weeks, months or years.



If desired, you can force an immediate publication of an updated CRL to the CA's CDP locations.

Try It Now!

To force an immediate publication of a CRL, open the Certification Authority snap-in > right-click on Revoked Certificates > All Tasks > Publish.

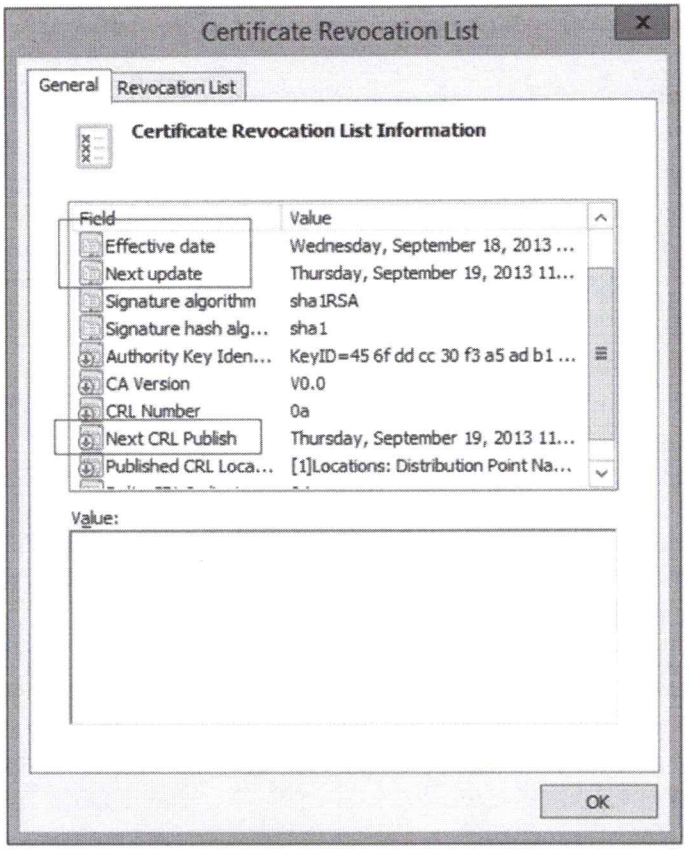
A CRL will include the date and time of its next scheduled update. A client computer will use this to determine when it should download an updated CRL. A client computer will continue to use a locally-cached CRL until the next scheduled update as defined in the CRL itself. This is true even if a more up-to-date CRL has been published.

> **Important!** Changing the CRL publication schedule or forcing a CRL publication does not cause clients to download the latest CRL.

When you examine the properties of a CRL, there are three important dates:

- **Effective Date:** The earliest date at which the CRL may be used.
- **Next Update:** The latest date at which the CRL may be used (expiration date).
- **Next CRL Publish:** A hint to the client about when the CA expects to publish a new CRL so that the client may pre-fetch and cache that CRL before the current

CRL expires. This provides a buffer or measure of safety in case all CRL distribution points are unavailable when a CRL expires.



These CRL dates can be customized by editing the registry on the CA, but be aware that there are constraints on the relationships between these values, the time skew in the clocks of clients and CAs, and the expiration dates on the certificates of the root and subordinate CAs involved (search on "CRLOverlapPeriod" to find articles describing these values and guidance). Also, be aware that non-Windows clients might not behave the same as Windows even when both are RFC-compliant.

Client CRL Caching

Be aware that publishing an updated CRL does not update the locally cached CRL on client computers. Scheduling or forcing a CRL publication only writes the updated CRL to the CDP locations, it does not cause any clients on the network to immediately download the updated CRL. A client will continue to use a locally-cached CRL until the next scheduled update of that CRL or until that CRL has been manually marked as invalid, whichever comes first.

To see the CRLs currently being cached by the client (it shows their URLs):

```
certutil.exe -URLCache CRL
```

To manually invalidate all locally-cached CRLs, which triggers fresh CRL downloads:

```
certutil.exe -setreg chain\ChainCacheResyncFiletime '@now'
```

Delete locally-cached CRLs after invalidating them, just for good measure:

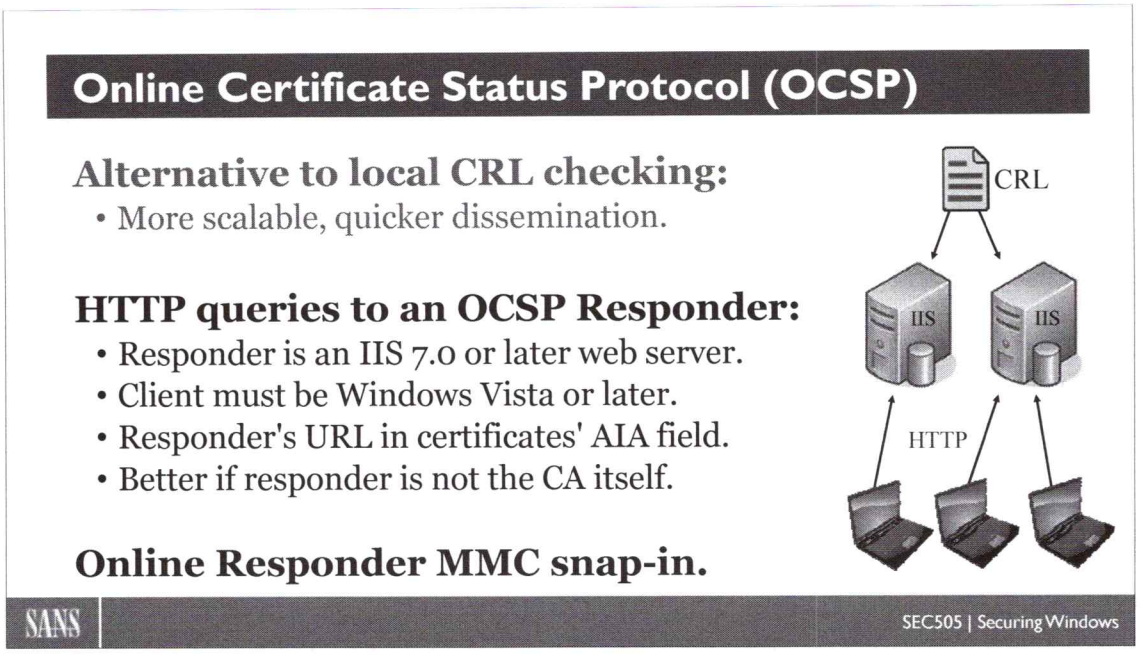
```
certutil.exe -URLcache CRL delete
```

In an emergency, such as when important certificates have been revoked and you wish for clients to know about this as soon as possible, the above command to immediately invalidate all cached CRLs might be remotely executed through PowerShell remoting or a Group Policy immediate task.

What Are Delta CRLs?

A full Certificate Revocation List (CRL) contains the serial numbers of all revoked certificates from a CA which have not yet time-expired. CRLs can potentially get very large. To alleviate this problem, Windows Server 2003 and later supports Delta CRLs. A "Delta CRL" is similar to a differential tape backup: it only lists the certificates which have been revoked since the last time the client obtained the full CRL from the CA, i.e., it only lists the new additions. Hence, a client will first obtain the full CRL the first time it connects to the CA, but thereafter it will download only the Delta CRL until its next scheduled "full download" time. Only Windows XP and later can use Delta CRLs as a client. Unless your CRLs are getting very large (> 5MB) using Delta CRLs is probably not necessary and adds more complexity.

However, Delta CRLs still allow a delay between the revocation of a certificate and when relying parties could be made aware of that revocation. What would be better is a real-time lightweight protocol that could be used by clients to perform a quick query of a particular certificate at the moment the client needs to choose whether to trust that certificate. Fortunately, starting with Server 2008 and later, we have the Online Certificate Status Protocol (OCSP) for just this purpose!



Online Certificate Status Protocol (OCSP)

Online Certificate Status Protocol (OCSP) is a lightweight and fast HTTP-based protocol to perform certificate revocation checking (RFC 2560). It is intended to replace or complement traditional CRL status checking. The advantages of OCSP over CRLs is greater scalability and quicker availability of revocation change information to relying clients.

A Windows OCSP server, called an "OCSP responder", is an IIS 7.0 or later web server. On the client side, Windows Vista and later have built-in support for OCSP, but there are third-party extensions available to XP/2003 to make these older operating systems at least partially support it too. Keep in mind, though, that there is no guarantee a particular application will use the CNG interface, hence, there's no guarantee that any arbitrary application running on Windows Vista/2008 or later will actually use OCSP. Your browser, for example, will most likely support OCSP, but what about your third-party VPN client or e-mail application?

OCSP Is HTTP-Based

OCSP as a protocol is quite simple. The client sends an HTTP GET request to the OCSP responder with the serial number of the certificate to be checked, and the responder replies with a digitally-signed response indicating either good, revoked, unknown or error. The client checks the signature on the response to prevent spoofing. The client must trust the issuer of the certificate the responder uses to sign its responses.

> **Note:** While it is possible to configure OCSP to use HTTPS, the server's responses are already digitally signed, so the use of SSL would impose an enormous performance penalty without much additional benefit.

Some clients can include a nonce (a random number challenge) in their requests which the responder must incorporate into its response in order to prevent replay attacks or other mischief. At the time of this writing, however, none of the OCSP clients from Microsoft support the use of challenge nonces even though the OCSP responder itself does.

A drawback of OCSP to be aware of, though, is that it reveals to the responder which certificates the client is checking at what times. If the OCSP server is not your own, this is a privacy risk.

There is also a configuration change which is required on the CA to allow OCSP responders to work correctly even after the certificate of the CA itself is renewed or replaced. Run the following command on the CA, not on the OCSP web server, to set the necessary registry value and then restart Certificate Services on the CA:

```
certutil.exe -setreg ca\UseDefinedCACertInRequest 1
```

What does this registry value do? It allows OCSP responders to renew their OCSP signing certificate using the older (but not yet expired) certificate on the CA. By default, the signing certificate on the OCSP responder will be issued or renewed using the latest certificate on the CA, but sometimes the OCSP responder will need to sign one of its responses with a certificate that was originally issued by the previous certificate on the CA. Setting this registry value on the CA changes this default behavior.

For each signing certificate the OCSP responder will need to use, which is typically only the current certificate and the just-previous certificate, the OCSP responder will need a separate revocation configuration (as described below in Configuration Steps), hence, the OCSP responder will typically require two revocation configurations.

Requirements

The requirements to use OCSP with Windows are the following:

- OCSP responder must be Windows Server 2008 or later, but it must be Enterprise Edition, not Standard Edition.
- If it is Windows, the CA must be Windows Server 2003 or later, but note that even non-Microsoft CAs are potentially supported.
- The OCSP client must be Windows Vista or later.

The CA does not have to be the OCSP responder; in fact, it is better for performance and security if the CA and OCSP responder are hosted on two different machines.

You do not need to reissue certificates if the current certificates lack OCSP information in their AIA fields. You can configure the OCSP responder with the necessary CDP URLs afterwards; that is to say, neither the OCSP responder nor the OCSP client is

prevented from using OCSP checking if the certificate to be checked lacks OCSP information inside it.

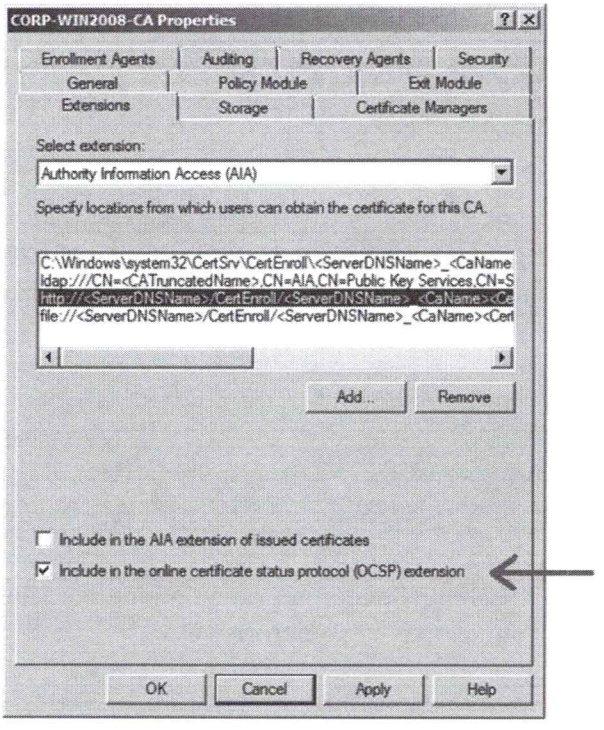
Configuration Steps

To configure Windows Server 2008 or later as an OCSP responder for your PKI, follow these steps.

1) Using the Certificate Templates snap-in on the CA, change the permissions on the "OCSP Response Signing" template to allow Read, Enroll and Autoenroll for the computer accounts of the IIS servers which will act as OCSP responders. These computers will use Group Policy autoenrollment to obtain and renew their certificates.

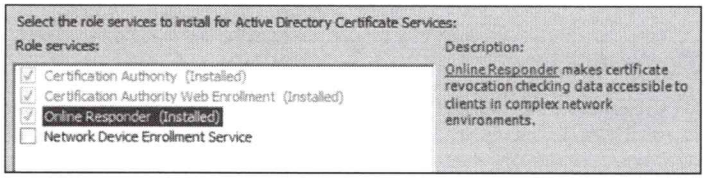
2) Using the Certification Authority snap-in on the CA, add the "OCSP Response Signing" template to the Certificate Templates container (right-click that container > New > Certificate Template To Issue).

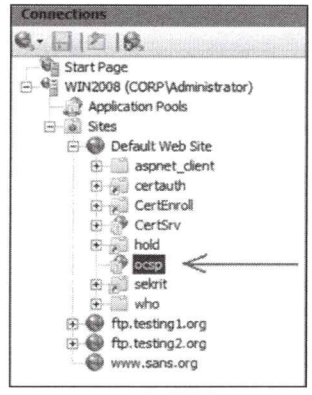
3) Using the Certification Authority snap-in on the CA, right-click the CA > Properties > Extensions tab > select Authority Information Access (AIA) > Add the HTTP URL to your intended responder, e.g., "http://www.sans.org/ocsp/" > check the box to "Include in the online certificate status protocol (OCSP) extension" > OK.



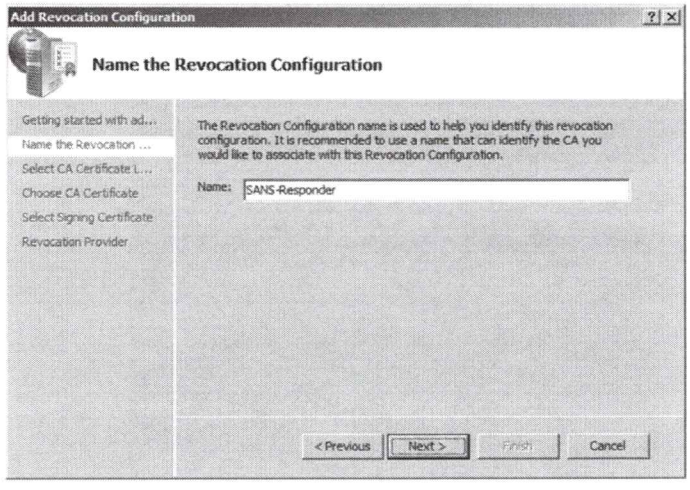
4) Ensure that the IIS computer which will be the OCSP responder has been configured through Group Policy to perform certificate autoenrollment, including the option to automatically renew expiring certificates. This IIS server may or may not be the CA.

5) Using Server Manager on the IIS server which will be the OCSP responder, confirm that IIS is installed, then add the "Online Responder" role service from the "Active Directory Certificate Services" top-level role. This will add a new application directory named "ocsp" to your default web site.

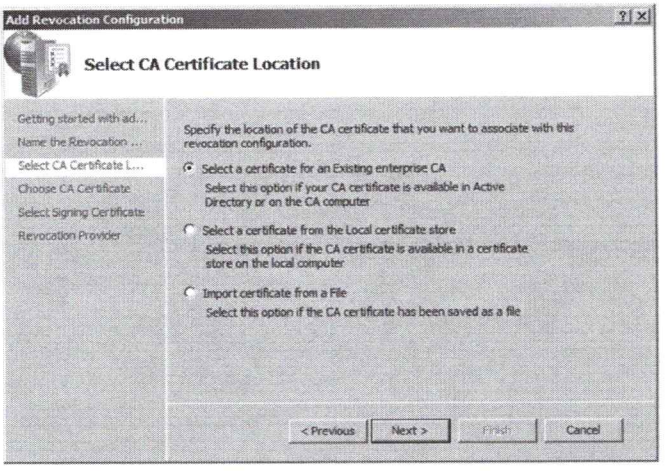




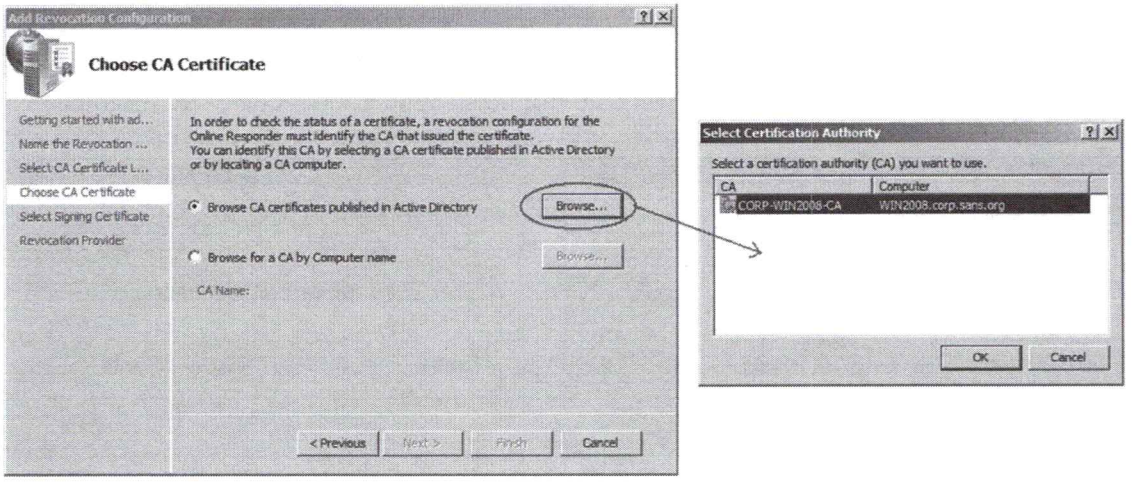
6) Using the Online Responder snap-in on the OCSP responder, right-click the Revocation Configuration container > Add Revocation Configuration. This launches a wizard which will ask you a few questions. Firstly, enter the name of CA, PKI or organization whose revocation information will be published through this OCSP responder.



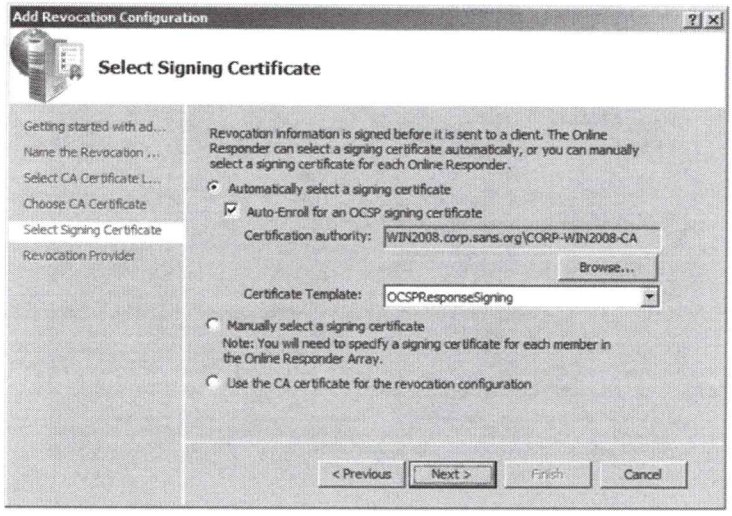
Next, assuming your OCSP responder is a domain member, choose the option to "Select a certificate from an existing enterprise CA".



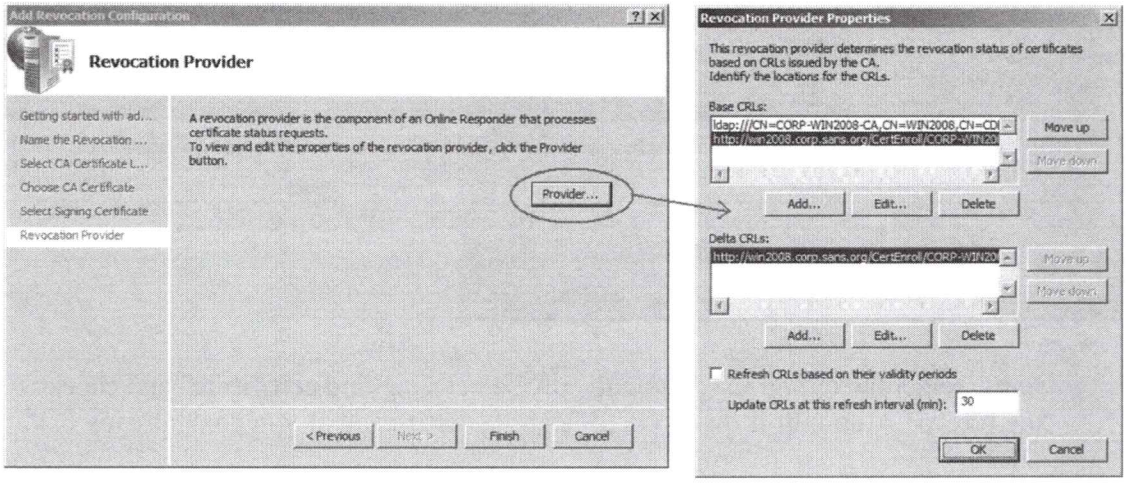
Next, select the option to "Browse CA certificates published in Active Directory" > click Browse > select the desired CA certificate > OK.



Next, select the option to "Automatically select a signing certificate" and check the box to "Auto-enroll for an OCSP signing certificate" if you don't already have one.



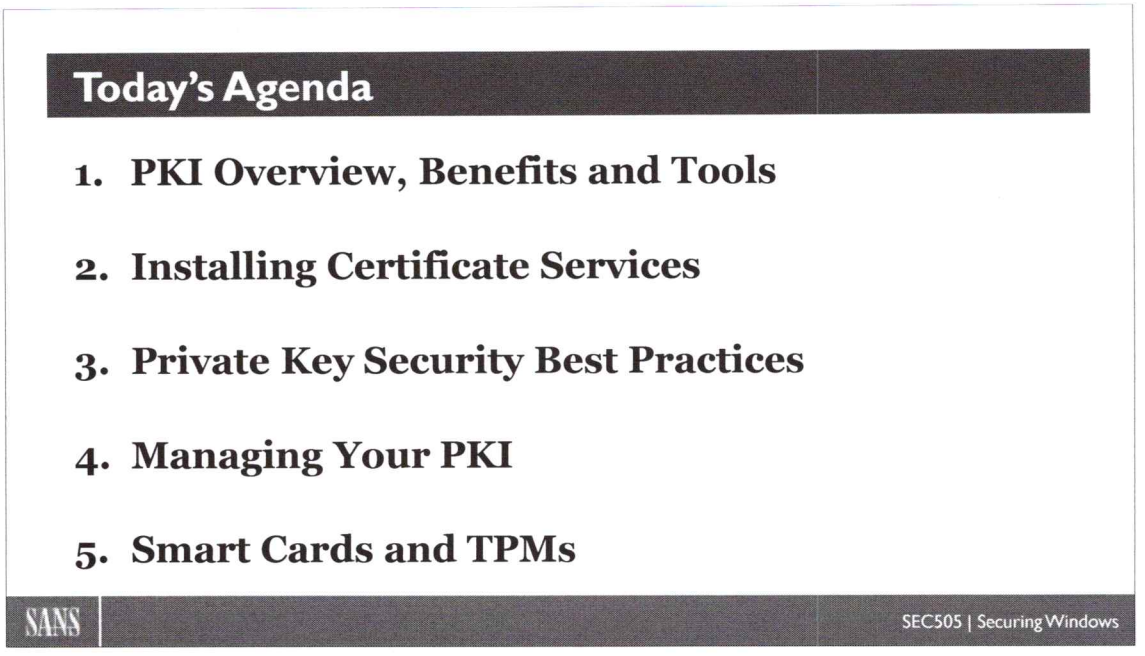
Next, click the Provider button > confirm that you have the correct URLs to your base and delta CRLs. You may need to add the URL for the delta CRL yourself. You can use the information on the Extensions tab in the properties of your CA in the Certification Authority snap-in as a guide. You will also likely wish to make the OCSP responder check for new delta CRLs more quickly than those CRLs are typically published, perhaps every 30 minutes. When done, click OK > Finish.



Auditing and Delegation

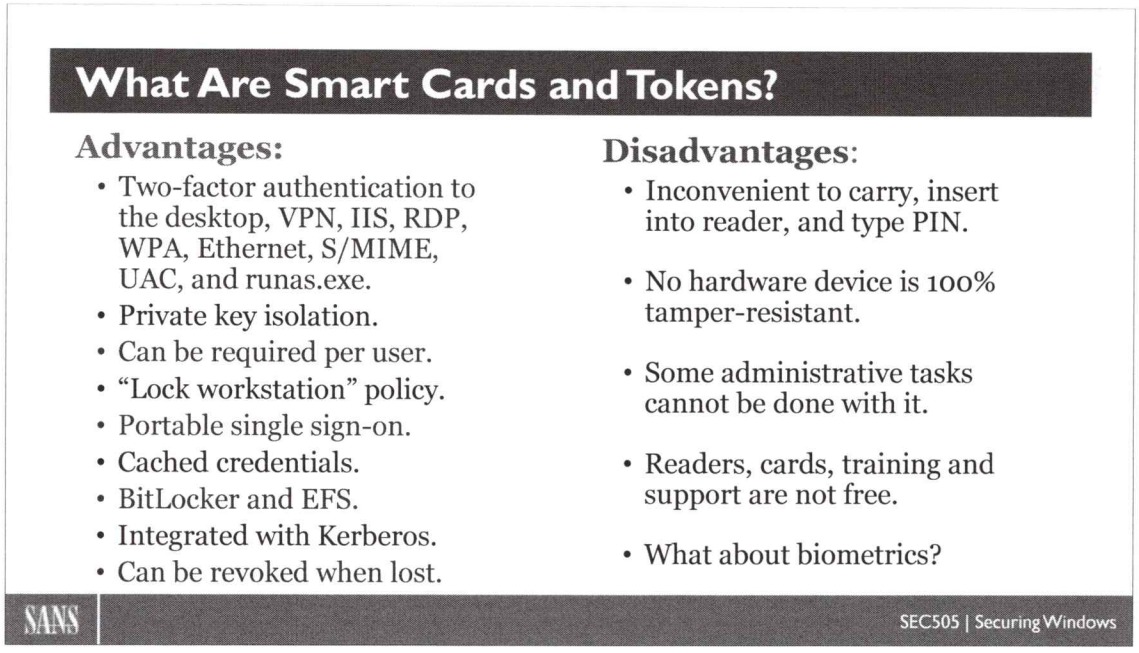
To delegate authority over the management of the OCSP responder and to log changes, right-click the Online Responder snap-in > Responder Properties > Security tab (to delegate) and Audit tab (to configure logging).

The "Manage Online Responder" permission allows one to reconfigure the service. The "Proxy Requests" permission is needed only by the Network Service account to submit requests to the Online Responder service (which can be seen in the Services applet in Administrative Tools). The logging options are self-explanatory.



Today's Agenda

Now that we have a functioning PKI, what can we do with it? In the next section we will see how to deploy smart cards. A smart card isn't always in the shape of a card, we can also have smart USB tokens and Trusted Platform Module (TPM) chips in our motherboards. If you have a Windows tablet or phone, you may have a built-in TPM and not even know it!



What Are Smart Cards and Tokens?

A "smart card" is a piece of plastic the size and thickness of a credit card that contains an embedded microprocessor and a small amount of EEPROM memory (generally 32k or less). The surface of the card exposes tiny electrical contact plates to communicate with a PCMCIA, RS-232 or USB reader, from which it also draws its power.

A "smart token" is a USB device the size of one's pinky finger. It is only slightly larger than the male USB plug itself. Smart tokens also have EEPROM memory and an on-board processor. The idea is the same as a smart card, but with a different physical design. And USB smart tokens are often more rugged than smart cards.

The card or token contains its own miniature operating system. In short, smart cards and tokens are simple computers.

Smart cards and tokens have a wide variety of applications beyond network security. Smart cards are already in widespread use in Europe and parts of Asia, and will soon be common in the United States. They are an integral part of a fully-deployed PKI.

A smart card/token can be a Cryptographic Service Provider (CSP). It can provide all the storage features and cryptographic operations required by (and accessible through) CryptoAPI. A smart card/token can store one's certificates and private keys, as well as perform cryptographic operations using its own CPU and memory. For example, when digitally signing e-mail, the signing occurs on the card/token itself so that the private key never leaves the smart card.

Windows supports PC/SC-compliant plug-and-play smart cards and readers which conform to the ISO 7816-1,2 and 3 specifications. Smart tokens usually use USB ports, and Windows supports those as well.

What Are The Security Advantages Of Using Smart Cards/Tokens?

The security advantages of using smart cards include the following:

- Smart cards/tokens can be used to log onto a local or remote computer. Smart card/token two-factor authentication is more secure than password-only authentication because the user has to both *know something* (the card's PIN number) and has to *have something* (the card/token itself) in order to log on. Users are less likely to write down a short PIN number than a long and complex password, and the PIN number does not have to be changed as often. Generally, users will grasp the concept of the card/token as a "key to unlock their computers" better than the role of passwords in security.

- Smart card/token logon is integrated with Kerberos authentication following the PKINIT standard for replacing a user's Kerberos password with their certificate instead (see below).

- Smart cards can replace your current employee ID cards that have pictures and magnetic strips. They can also be your company debit card, contain your medical records, hold "digital cash" and be your personal credit card too.

- If a thief attempts to guess the PIN number, the smart card/token will disable itself after a small number of incorrect guesses (usually three). An administrative PIN can then re-enable the card or token for user-mode access.

- Smart card/token logon can be required for any user account.

- Using Group Policy, computers can be made to "Lock Workstation" or force a logoff of the user when the smart card/token is removed.

- Smart cards and tokens are designed to be tamper-resistant to prevent the physical extraction of information from them. Though the physical security features of the devices can be overcome, it is still far better to store them in the device than on the hard drive. Moreover, smart card/token technology is still evolving.

- Smart cards and tokens perform all private key operations themselves, hence, the private key is never exposed to the operating system or applications. This includes local log on, e-mail signatures, TLS/SSL key exchanges, and remote smart card/token authentication for dial-up and VPN users.

- Smart cards and tokens are portable, so it is easy for users to carry their key pairs around with them and safely use them on other computers, such as at home or at an airport terminal. This is an alternative to roaming user profiles or flash disks.

- A smart card/token has its own miniature operating system, CPU and writeable memory; hence, additional security features can be programmed into the card. For example, a smart card could be programmed to look for another particular smart card in a second reader on the computer: without the second card being present, the first card will remain disabled, and vice versa.

- A smart card or token can be compatible with multiple types of hardware and different operating systems. It could provide *true* "single sign-on" across platforms and security systems since the hardware is not limited to computers. Cellular phones will have integrated smart card readers to encrypt voice data and pay for long-distance charges. Soon, your new Volkswagen may require a smart card to start.

- If the domain's functional level is Server 2008-R2 or higher, when a user logs on with a smart card, you can modify that user's group membership on-the-fly as they log on such that the user will temporarily become a member of a custom group which indicates the use of a smart card for logon. Critical network resources can then have permissions which only allow access to that custom group. (For details and the necessary PowerShell configuration scripts, Google on "authentication mechanism assurance active directory".)

What Are The Disadvantages of Smart Cards and Tokens?

The drawbacks to using smart cards include the following:

- Some smart card/token CSPs do not permit the export of private keys from the device. This is good for security, but bad for fault tolerance. If a smart card/token is lost or damaged, and it contained the only copy of a critical private key, then that key is simply lost forever. Because of this issue, though, usually smart cards are only for authentication, not data encryption (BitLocker has a recovery mechanism that does not require the original smart card).

- If a user forgets to bring their smart card/token to work or otherwise temporarily cannot get to it, that user will be inconvenienced. In this case, change the user's account to permit regular Ctrl-Alt-Del logon, assign a difficult password, then require a smart card again the next day. If a certificate is absolutely required, issue a certificate with a 48-hour validity period, or, revoke the first card and issue a new one.

- No physical device can be 100% tamper-proof, hence, smart cards and tokens should still be physically protected against theft.

- Currently, private key operations on smart cards and tokens are noticeably slower than the same operations performed on the mainboard CPU, but then again bulk encryption is usually performed with other keys secured *by* the smart card, not *on* the smart card.

- Certain administrative tasks cannot be performed with smart card/token logons, e.g., joining a computer to a domain, promoting a server to become a domain controller, etc. These administrative accounts will have to have the option to log on either with a smart card/token or the old-fashioned way.

- Smart cards and their readers are not free. On the other hand, what is the price of not having strong security? Besides, a card purchased in bulk costs less than $10, and a card reader, if it's not built into your laptop, less than $30.

Note: The following pages will refer only to smart cards, not tokens, in order to avoid cluttering the text. But virtually everything discussed about smart cards will also apply to smart tokens. It remains to be seen which format will win market dominance.

How Are Smart Cards And Kerberos Combined?

The user's certificate on the smart card is used during the Kerberos 5 authentication protocol in accordance with the IETF draft "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)".

PKINIT authentication provides the following benefits:

- Transparent integration of smart card and Kerberos authentication to user accounts in Active Directory without administrative overhead to enable it.
- User private key signatures to authenticate users to domain controllers.
- Domain controller private key signatures to authenticate to users.
- Public key encryption of random keys used to encrypt Ticket-Granting Tickets (TGTs) for secure key exchange across the network or Internet.
- Use of time-stamps to prevent replay attacks.

The following is a condensed summary of PKINIT authentication. It assumes the reader is already familiar with Kerberos. Please see the IETF draft for PKINIT and RFC 1510 for a complete description of the protocol. Microsoft also has a number of whitepapers on Windows Kerberos.

1. When the user's computer sends an authentication service request to the KDC service running on a Windows domain controller, the request will include the user's entire certificate as well as some data encrypted with the user's private key on the smart card.

2. The KDC domain controller will verify that the certificate has a valid path to a trusted CA. The domain controller will then verify that the data encrypted with the user's private key can be decrypted with the corresponding public key in the

certificate. The domain controller will also check the time-stamp in the authentication request to make sure it is not being replayed in an attempt to impersonate the user.

3. If everything checks correctly, the domain controller will use the User Principal Name (UPN) in the certificate to locate the user's account in Active Directory. The user's SIDs for her account and group memberships are put into a Ticket-Granting Ticket (TGT). The TGT is encrypted with a random key, then that key is encrypted with the public key from the user's certificate. Both the TGT and the encrypted key are signed with the domain controller's private key and returned to the user's computer.

4. The user will decrypt the encrypted random key with her private key (on the smart card), then use the random key to decrypt the TGT. The user's computer will verify the domain controller's signature to a trusted CA to ensure that the TGT has not been altered and to authenticate the domain controller.

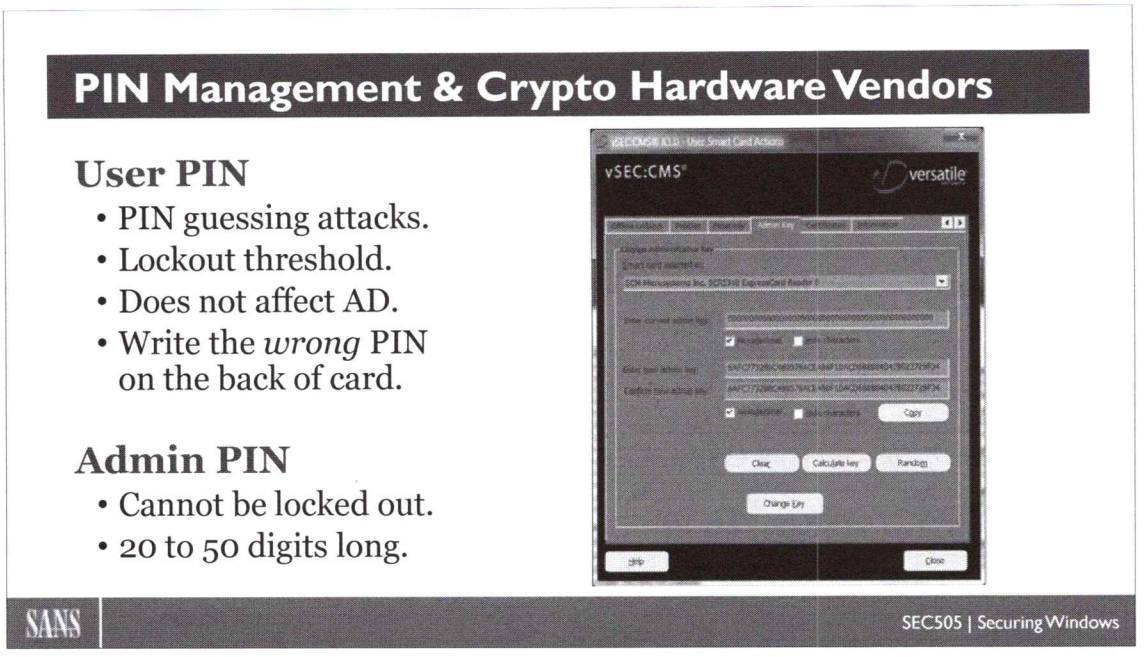
5. The user will use the TGT to continue with the Kerberos protocol in the standard way to request other tickets from the Ticket Granting Service (TGS) on domain controllers.

If no domain controller is available, smart card logon will still succeed because the public key-encrypted TGT from prior authentication sessions is cached on the user's computer. This is secure because the TGT can only be decrypted with the user's private key on their smart card.

There are some conditions which must be met in order for smart card PKINIT authentication to work:

- The Windows enterprise CA must issue the certificate using either the Smart Card User or Smart Card Logon template.
- All certificates in the CA chain from root to issuing CA must be accessible to both client and domain controller. The root CA must be trusted of course.
- Every subordinate CA and the root CA must provide a CRL and these CRLs must be available to both client and domain controller. The time limits on these CRLs must all still be valid.

If any CA certificate is unobtainable, or has been revoked, or does not have a time-valid and obtainable CRL, then authentication will fail.



PIN Management & Crypto Hardware Vendors

To access a private key on a smart card, the user must enter a PIN or password. If the wrong PIN is entered too many times, such as by an attacker who is trying to guess it, the card locks itself and now must be unblocked. A typical card locks itself after 3-7 failed PIN attempts, but these features vary by vendor.

To unblock a locked smart card, the user must either enter an Administrator PIN or contact the help desk to go through a challenge-response procedure which proves knowledge of the Administrator PIN *to the card* without revealing the Administrator PIN itself to the user (it's just a five-minute procedure when done over the phone). The Administrator PIN is different than the User PIN. The Administrator PIN is sometimes called the "Admin Key" or the "Unlock PIN", depending on the vendor. If an attacker tries to guess the Administrator PIN, this PIN either has no card lockout policy (in which case the PIN will be very long) or the card will disable itself completely (in which case the card must either have its memory scrubbed or be shredded and then thrown away).

It's important to understand that anyone who knows the Administrator PIN can reset the User PIN. An attacker who knows the User PIN can then perform actions as the original owner of card, such as VPN into the network or log onto web applications. It is not enough to change the default User PIN in order to secure a smart card, the default Administrator PIN must be changed too.

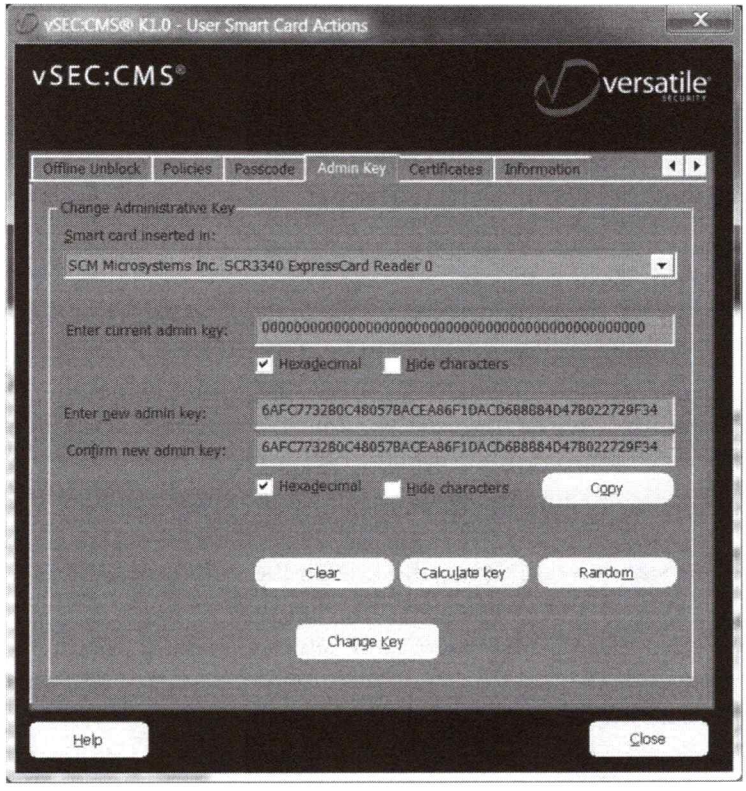
If a smart card is stolen, we want the thief to trigger lockout of the User PIN. Make sure this lockout policy is configured in the card (not in Active Directory), that the card's lockout threshold is only 3-7 attempts, and that the lockout duration is at least 24 hours. A simple but effective trick is to write *incorrect* PIN numbers on the back of the card and

train the user to never use any of those PINs. A thief will probably try all of the incorrect PINs, thus socially engineering the thief into triggering the lockout.

Beware, however, because some smart card vendors use devious marketing tricks to get you to purchase their PKI management systems in addition to the cards themselves; for example, while the smart cards you just purchased might come with a tool for changing the default User PIN, you might not be able to change the card's Administrator PIN unless you also purchase special additional management tools -- but this means that anyone who steals the card can simply use the default Administrator PIN to unlock the card! Any smart card whose default Administrator PIN cannot be changed is worthless. Strongly consider not using any smart card vendor who stoops to such tricks at your expense, especially when there is little or no warning about these dangers on the vendor's web site when you are researching the cards for purchase.

Free Smart Card Management Tool

If you need to change the Administrator PIN on smart cards you've purchased, but the vendor won't help you at a price you can afford, then you might look for third-party tools instead; for example, Versatile Security (http://www.versasec.com) has a free tool which can change the Administrator PIN on a variety of smart cards and smart tokens from other vendors (vSEC:CMS-U).



Managing the lifecycle of smart cards for a large number of users might be easier with the vendor's enterprise-level solution, but this additional software is not necessary to get started or to do a smaller deployment for just the high-value targets. Microsoft has their

Identity Manager product, but you might find other solutions less expensive and less difficult to deploy.

Hardware Vendors: Smart Cards and Smart USB Tokens

Here are a few of the leading manufacturers of smart cards and smart USB tokens which are compatible with Microsoft Windows:

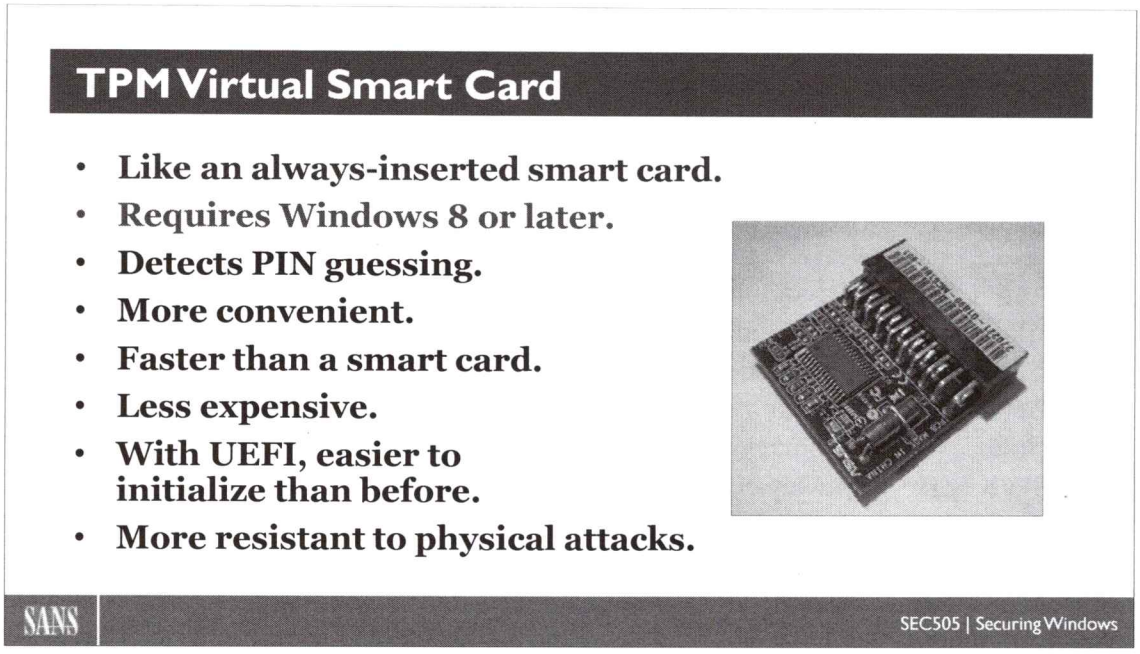
- ActivCard (www.hidglobal.com)
- Aloha Software (www.aloaha.com)
- Athena Smartcard Solutions (www.athena-scs.com)
- Gemalto (www.gemalto.com) **-- the .NET cards work well in particular**
- GoldKey (www.goldkey.com)
- PIVKey (www.pivkey.com) -- **tiny USB tokens also**
- Spyrus (www.spyrus.com)
- United Access (www.united-access.com)

The last time this author checked, Athena and Gemalto were the most friendly for those who only want to purchase a small number of tokens or cards, such as for testing. As always, confirm that you can change both the User PIN and the Admin PIN for any smart cards you consider purchasing.

Hardware Vendors: Private Key Storage and Accelerators

The following is a sampling of manufacturers of cryptographic hardware besides smart cards. Dedicated storage devices provide industrial-strength tamperproof storage of private keys that may meet various Federal Information Processing Standards (FIPS 140). Cryptographic accelerators offload cryptographic processing from the operating system for the sake of security and efficiency, e.g., on high-volume commercial webservers.

- Atalla (www.atalla.com)
- IBM (www-03.ibm.com/security/cryptocards/)
- nCipher (www.thales-esecurity.com) -- very popular for Windows CA HSMs.
- SafeNet (www.safenet-inc.com) -- very popular for Windows CA HSMs.
- Spyrus (www.spyrus.com)



TPM Virtual Smart Card

A Trusted Platform Module (TPM) is a chip built into the motherboard of a computer which can perform on-board random number generation, encryption, hashing, and other cryptographic operations. The TPM is also a secure storage location for keys, passwords, hashes and other secret data. For more information about TPMs, see the document "TCG Architecture Overview" (https://www.trustedcomputinggroup.org/specs/TPM).

A TPM chip can be found in all Windows Phones, all Microsoft Surface tablets, most business-class laptops, and some high-end PC motherboards. You'll have to read the fine print from the manufacturer to know for certain that a device comes with a TPM (or at least an empty TPM socket).

Turning On and Initializing The TPM

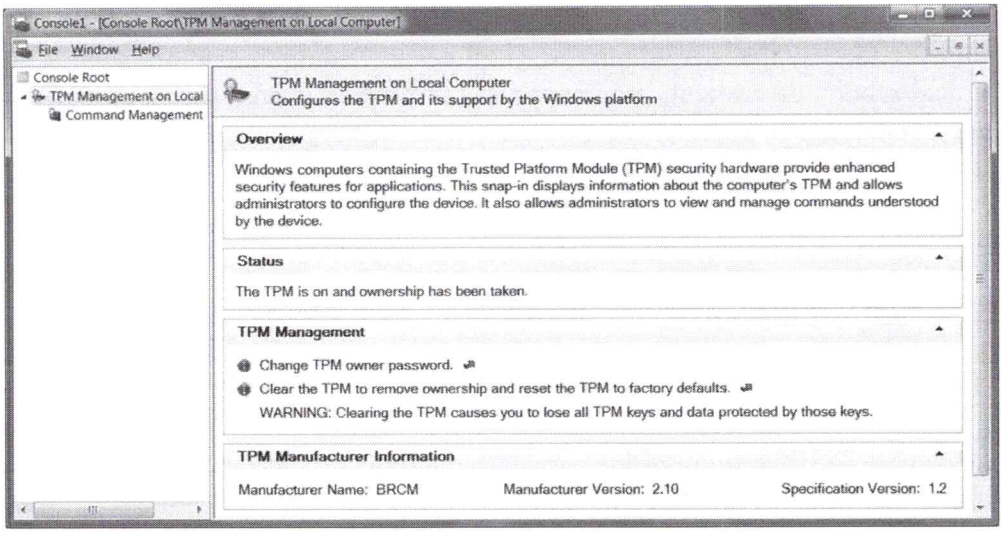
Before the TPM can be used, the TPM must be:

1. Enabled in the computer's firmware,
2. Turned on in Windows, and
3. Initialized in Windows with an owner's password.

How the TPM is enabled in firmware is determined by the computer's manufacturer (usually by pressing F2 or F12 during boot-up, etc.), but it might already be enabled at the factory. With tablets and phones, the TPM will almost certainly come enabled.

The TPM is turned on and initialized with an owner's password by using an MMC console snap-in named "TPM Management" (`TPM.MSC`) or in PowerShell. Turning on the TPM simply makes it available for use, while "initializing" the TPM means setting an

owner's password. This password is required whenever the TPM is significantly modified, e.g., turned on/off, memory cleared, password changed, etc.



The owner's password can be backed up to a file or printed out for recovery purposes.

The TPM is accessed through a programming API named "TPM Base Services (TBS)". This is the API which BitLocker and TPM Management snap-in go through, but it's available for third-party applications too. The TBS interface is also accessible through the Windows Management Instrumentation (WMI) API, and Microsoft provides a WMI tool to manage both the TPM chip and BitLocker generally: MANAGE-BDE.EXE.

Windows 8 and later can also use PowerShell cmdlets for TPMs (get-help *tpm*).

The TPM chip is not just for Microsoft virtual smart cards and BitLocker, other vendors can use it too; for example, SafeBoot and Utimaco SafeGuard both support the use of a TPM for data encryption.

Physical Presence Requirements

Turning on and initializing the TPM with an owner's password requires being physically present at the computer. For the sake of security, there is no way to 100% script these actions. The MANAGE-BDE tool and the TPM PowerShell cmdlets can request that a change be made, but the change will not actually be made until after the computer reboots and someone at the console approves the change (this is enforced by the firmware, not Windows).

On a Dell Latitude D820, for example, turning on the TPM via MANAGE-BDE.EXE, locally or remotely, prompts the user of the script to reboot. At reboot, the firmware displays a text message right after POST:

```
A request to change the configuration of this
computer's TPM is pending.  If you did not expect this
```

```
message, use the arrow keys to select IGNORE and the
TPM configuration will not be changed.  Otherwise
select MODIFY to allow this change.
```

If nothing is done by the user within three minutes, the computer simply powers down. At the next reboot, the same message appears. If multiple changes were made in a row without rebooting that each require physical presence, the machine must be rebooted multiple times and each change approved by selecting MODIFY. Again, this is all handled by the firmware, not Windows, in accordance with TPM specification requirements.

TPM Virtual Smart Card

Windows 8, Windows RT and later support smart cards and smart USB tokens, but they also support the use of "virtual smart cards" implemented by the TPM in the motherboard (TPM version 1.2 and later). This is effectively an always-inserted smart card. The TPM virtual smart card still requires a PIN or alpha-numeric passphrase, and the TPM can detect guessing attacks, just like traditional smart cards.

Virtual Smart Card Advantages

TPM virtual smart cards can be used for desktop logon, DirectAccess, wireless, User Account Control prompts, signing e-mail and other purposes because the TPM is exposed through the Key Storage Provider (KSP) cryptographic service provider interface just like a "real" smart card. As far as Windows is concerned, there isn't a programmatic difference between a smart card, a smart USB token, or a virtualized TPM smart card.

A user's private keys are never decrypted on the hard drive or in system memory, they are only decrypted in the memory of the TPM chip itself.

Up to ten virtual smart cards can be created on a single computer, and each virtual card will have a capacity of 30 private keys and their corresponding certificates. Hence, even on a shared computer, each user can have their own separate virtual smart card; however, with five or more virtual cards, there will be performance penalty.

If a user with a stand-alone BYOD computer uses Remote Desktop Protocol (RDP) to connect to another computer joined to the domain, then certificate enrollment tools can be used with the local TPM even though the local computer is a stand-alone. This is not Group Policy auto-enrollment, but at least it is possible for a BYOD computer to get a company certificate, perhaps for the sake of S/MIME, web applications, VPNs, etc.

Windows 8, Windows RT and later also include enhanced Group Policy control over how the TPM is initialized to make the process easier, plus a new TPM wizard for when manual initialization is required (such as on older TPM-enabled systems that lack UEFI firmware).

The TPM smart card is faster, more convenient and more secure than a regular smart card against physical-layer attacks. Because the TPM is a chip integrated into the

motherboard, it has much faster throughput than a regular smart card. A user must be trained to carry around a regular smart card, insert it properly, avoid breaking or losing it, and it usually sticks out the side of the laptop or tablet (unless you slice off the end). And if an adversary is going to mount a physical attack, presumably the flimsy and thin chip in a regular smart card is easier to penetrate with probes or micro-drills than a plastic-encased TPM chip soldered into the motherboard (time will tell).

Virtual Smart Card Disadvantages

The main disadvantage is that the TPM is not portable like a smart card, it is soldered into the motherboard, hence, a user cannot carry the TPM around from one machine to the next for single sign-on, and the user cannot physically secure the TPM separate from its motherboard.

Another disadvantage is that, unlike an external smart card, when there is a PIN-guessing attack against the TPM, the TPM will not permanently lock like a smart card, the TPM will only delay user-mode access for a period of some seconds and then increase this period with each subsequent failed PIN attempt. The time delay and time increase algorithm is set by the TPM manufacturer, not by Windows (this may change with future TPM versions). Nonetheless, with a random 8-character alpha-numeric PIN and an increasing delay after each PIN failure, it should take years for an adversary to brute-force the PIN. Certificates in virtual smart cards can be revoked like other certificate too.

Finally, the TPM encrypts users' private keys and stores them on the hard drive. If the hard drive is formatted, wiped, damaged or replaced, then, even though the TPM is functioning normally, the user's private keys will be gone. Reinstalling the operating system, for example, will often reformat the C:\ drive, hence, this will also destroy any existing private key blobs on that volume.

Virtual Smart Card Management

Virtual smart cards can also be managed locally or remotely from the command line.

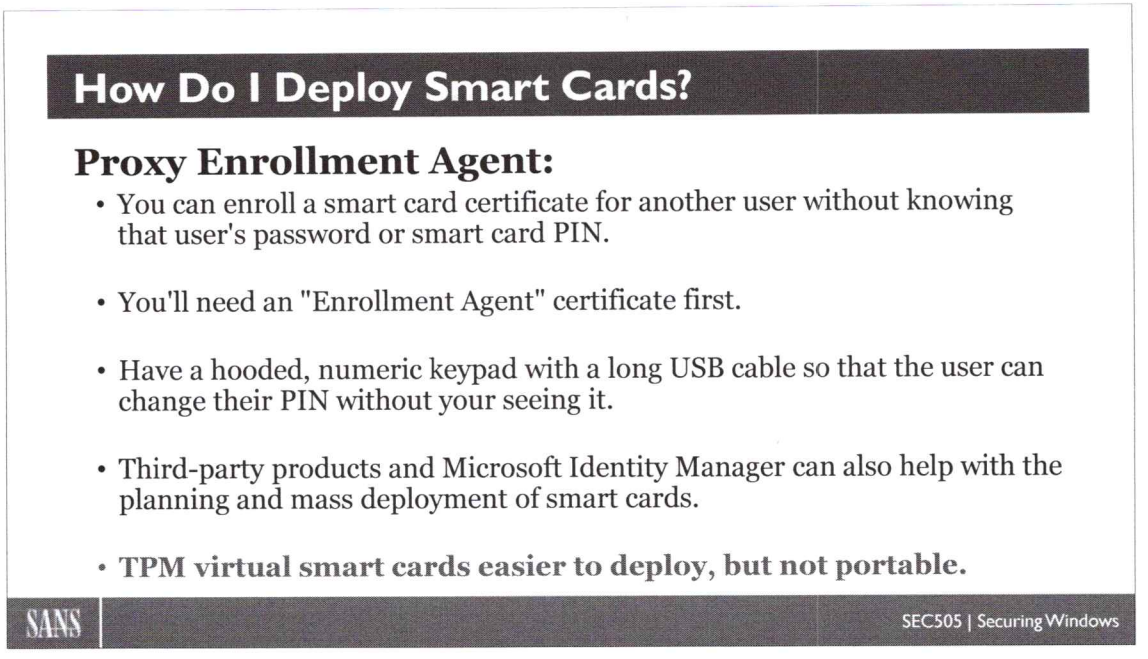
> ***Try It Now!***
> In PowerShell, run tpmvscmgr.exe /?

To read the documentation of the TpmVscMgr.exe tool's command-line options and, more importantly, the life-cycle management tasks for virtual smart cards, search Microsoft's web site for the technical paper entitled "Understanding and Evaluating Virtual Smart Cards."

The user PIN must be at least 8 alpha-numeric characters. The admin PIN must be 48 hexadecimal characters. There is also an optional PIN Unlock Key (PUK), but if a PUK is created, then the admin PIN cannot be used, and in general it is better to have an admin PIN for the sake of other management tasks besides just locally unlocking a forgotten user PIN. If all PINs are forgotten, the virtual smart cards can (and will have to be) deleted and new ones created with new certificates.

Can I Get The Convenience of a TPM with a Standard Smart Card?

Can you get some of the convenience benefits of a TPM virtual smart card with a traditional smart card? Yes, write the wrong PIN on the back of the physical smart card, get a removable but internal smart card reader, insert the card into the reader, mark with a pen where the card is flush with the side of the reader, then slice off the end of the card. Under normal circumstances, just leave the half-sized card in the reader all the time, but, if you're worried about the safety of the device in a particular circumstance, such as when flying through an unfriendly country, then simply pop out the reader+card combo. This trick is still far better than using a passphrase, you also get user mode lockout when there are PIN-guessing attacks, and you can use the reader+card combo on other computers with a compatible slot, like a 54mm ExpressCard.



How Do I Deploy Smart Cards?

To log on with a smart card, a user must obtain a certificate based on either the Smart Card User or Smart Card Logon templates (note that version 3.0 templates cannot be used for smart card logon). The only difference between the two is that Smart Card User certificates can also be used to sign and encrypt e-mail. The desired template must be available for use on the enterprise CA (it must be found in the Policy Settings container) and the permissions on the template must be changed to permit the necessary users to Enroll for those certificates.

Only enterprise CAs can issue Smart Card User or Smart Card Logon certificates. Stand-alone CAs cannot issue smart card certificates that can be used to authenticate to the domain. Stand-alone CAs can issue other types of smart card certificates, just not for AD authentication.

The template permissions force you to make a security decision: will users be permitted (or expected) to enroll for smart card certificates themselves, or will designated administrators issue smart cards with certificates for them by proxy enrollment?

Option One: User Self-Enrollment

If users will enroll themselves, the permissions on the relevant smart card template must be changed to permit this.

A pitfall of self-enrollment is that users must be capable of requesting and installing a certificate into their own smart cards. Even with step-by-step instructions, many users will fail to do this successfully. This will increase help desk support costs.

There are also security risks. Users will be tempted to create multiple smart cards for themselves "just in case", but where will these extra cards be kept? Also, if a user leaves his computer unattended, another user can "station hop" to the first's unlocked desktop and make a smart card for himself with the first's credentials; this will permit the station hopper to log on as the first user, even if the first's password is changed. Finally, users cannot be trusted to change the default PIN numbers on their cards.

However, many of these security risks can be somewhat mitigated (not eliminated) by implementing the following policies:

- All new smart cards should be distributed from a central office. Each card should be printed with a difficult-to-reproduce emblem of the company, the user's employee ID number, the user's name, picture and other identifying information. Many organizations already have such an office in place for employee ID cards, so this change is simply to make these cards smart.

- The distribution office will set a random PIN number for each card and store the PIN with the (empty) card until it is distributed to the user (a Post-It note or envelope with the PIN on it will suffice). The user is encouraged to change the PIN, but it will not compromise security greatly if he or she fails to do so. A random user account password should also be assigned at this time if the user's account won't be marked as requiring a smart card for interactive logon.

- The stated policy of the company is that each user is issued just one card, and each user is permitted to only use that one card with their employee ID number and picture on it. Cards not issued by the distribution office are prohibited and their use is cause for termination. Tell them that use of duplicate cards can be detected even if you can't reliably or automatically do so.

- Use Group Policy to "lock workstation" whenever a smart card is removed, and to require password-protected screensavers. This will help to prevent "station hopping".

- If a user claims to have lost their smart card in an attempt to keep multiple cards, the old smart card certificate should be revoked and a new one enrolled.

- You cannot use Group Policy to require smart card logon until after the user has enrolled themselves. A grace period of a week may be permitted, during which time the random assigned password is used.

Option Two: Proxy Enrollment

Instead of permitting users to enroll themselves, designated administrators can create smart cards on behalf of users. In this case, many of the above policies may still be used, but the difficulties of allowing (and expecting) users to enroll themselves disappears.

The trusted administrator who can issue smart card certificates to others is called the "proxy enrollment agent" or just "enrollment agent". The enrollment agent, in order to request a certificate on behalf of another person, must have a special certificate first. This certificate is from the template named "Enrollment Agent". This template is not loaded into the CA by default.

Once a trusted administrator has obtained a certificate from the Enrollment Agent template, that administrator can create a smart card for another person using the Certificates MMC snap-in.

To request a smart card certificate for another user, insert the card into the reader and have the PIN ready, then, in the Certificates snap-in (User), right-click the Personal container > All Tasks > Advanced Operations > Enroll On Behalf Of > Next > select your enrollment agent certificate > Next > select the Smartcard User template (or equivalent, if you've created a custom template) and then follow the prompts of the wizard to complete the enrollment with the PIN.

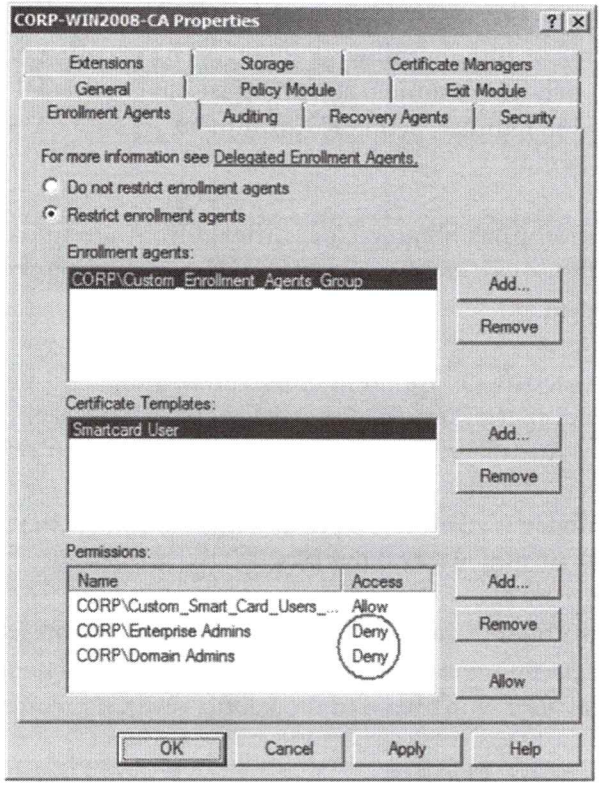
> **Tip:** If your enrollment agent certificate is itself on a smart card, then install two smart card readers on the computer which will be used to generate cards for others. This way you won't have to repeatedly swap out cards.

Restricting Enrollment Agents

A danger of proxy enrollment is the ability to issue a certificate for a user other than oneself, such as for the CEO or for an R&D scientist. With Server 2008 Enterprise CAs and later, you can more tightly regulate the activities of enrollment agent administrators. It is possible to restrict which certificates are proxy issued to which users in order to prevent mischief.

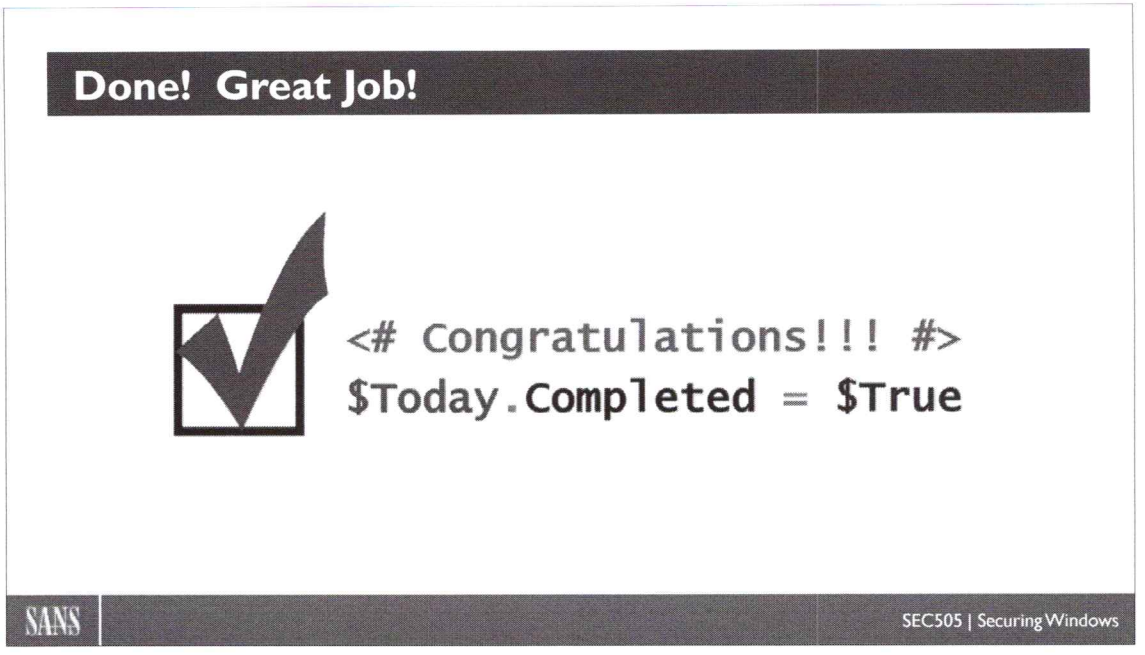
The recommendation is to create one or more custom groups for enrollment agents, then limit the certificate templates which these agents can use for proxy enrollment. You should also explicitly allow and/or deny proxy enrollment permissions on various users' groups; for example, allow proxy enrollment for the Sales group, but explicitly deny proxy enrollment for Enterprise Admins, Domain Admins, Executives, etc.

To restrict which users can be enrolled by proxy for which certificate types, open the Certification Authority snap-in > right-click the CA > Properties > Enrollment Agents tab.



Enrollment agent certificates are some of the most powerful in the PKI because their owners can potentially create certificates for *any* user account, *including the administrator's*. Hence, make sure to tightly regulate who has this power.

> **Warning!** The owners of Enrollment Agent certificates can potentially create smart cards for *any* user account, including a Domain Admin!



Congratulations!

You Have Finished The Course!

Thank you for attending this seminar!

Please complete the evaluation form and return it to the room monitor or place it in the "Course Evals" box. Your evaluations, especially your written comments, play an important role in how SANS seminars are updated.