Active Directory Certificate Services (ADCS)

A white paper by David Wozny for Oxford Computer Group

Table of Contents

Introduction

PKI Backgrounder

Public Key Infrastructure (PKI) is essentially concerned with harnessing the strength of cryptography and wrapping it up into digital certificates which can be applied to a multitude of security enforcing scenarios and applications. PKI usage generally follows the broad categories of Confidentiality, Integrity and Authentication (CIA). Typical examples of security solutions which leverage PKI are:

- Persistent encryption of files *at rest* – for example with Encrypting File System (EFS) to ensure confidentiality of data
- Digital signing of ActiveX browser controls, Java applets and Office macros to provide assurance of the source and integrity of software code
- Smart card authentication to Windows, remote access gateways (such as VPNs)

As the 'I in PKI' implies, it is not a single entity, but a range of complimentary services such as Certification Authorities (CAs), Registration Authorities (RAs), Validation Authorities (VAs), relying parties, revocation providers and Hardware Security Modules (HSMs). This paper is not intended to explore all of these services in detail, but rather to give a 'useful general background' for a rounded understanding of the basics.

Although almost universally recognised as a comprehensive platform for delivering trust services, PKI has suffered from bad press for being complex, unwieldy and process bound... and expensive. It's fair to say this has generally been the case; however, Microsoft's entry into the market turned this around in a lot of circumstances. Microsoft has made PKI a far more accessible and practical proposition for many enterprises which may have been considering implementing PKI but were put off by the aforementioned *challenges*.

Microsoft's PKI History

Microsoft first dabbled in PKI on the Windows Server platform by providing a Certification Authority in the Windows NT 4 Server option pack. Since then, Windows 2000 Server introduced a vastly improved CA capability as a standard application service, much like adding DNS or IIS, although in terms of flexibility, scalability and feature richness it still lagged significantly behind third party solutions. With Windows Server 2003 Microsoft introduced the notion of the Enterprise CA and truly laid claim to providing an enterprise class CA for all but the most demanding of scenarios. Windows Server 2008 sees further enhancements to 'Microsoft PKI', such as providing clustering capability, support for advanced algorithms and better revocation services.

Support for enrolment of smart cards and general certificate management has largely been the domain of third party specialist products. Windows Server does provide rudimentary web enrolment pages for smart card issuance as well as scripted / auto-enrolment / MMC snap-in based certificate enrolment for low assurance scenarios, however, these don't tend to scale very well and lack sophistication and extensibility. In 2005 matters changed when Microsoft purchased Canadian digital identity specialist Alacris and their flagship idNexus product, a smart card and certificate lifecycle management system tightly integrated with the Windows Server CA and Active Directory (AD). This product was initially branded Certificate Lifecycle Manager (CLM), then Microsoft consumed it into Identity Lifecycle Manager (ILM) in 2007. In time, it became ForeFront Identity Manager Certificate Management (FIM-CM) and in the opinion of the author it has strong merits. Indeed the author has implemented a FIM-CM solution which was used for smart card issuance in sixty countries – but it has never really ‘taken off’ and is not considered in the remainder of this paper.

Objective

This paper explores many of the business drivers for Microsoft centric PKI services. It explores the challenges and opportunities of implementing a PKI solution from the perspectives of design, implementation, operations, etc.

Various scenarios are explored and particular emphasis laid upon practical, real world applications and the supporting technologies which leverage PKI, rather than getting bogged down in the theoretical and policy based aspects which often *obstruct* a rapid grasp of the subject.

A number of terms are used in this document which may not be understood by ‘non-PKI aware’ readers, it is suggested you have a brief review of the glossary before proceeding. Given the sometimes complex and arcane nature of PKI, some poetic liberty has been taken to simplify readability, hopefully without distorting the facts. It is fair to say that some topics have been treated with brevity and rather simplistically; this is deliberate, as I have attempted to make this paper a relatively easy read and hopefully conducive to better understanding.

Public Key Infrastructure

Microsoft Certificate Services for Active Directory

The Certification Authority

The central component of the Windows Server PKI platform is undoubtedly the Certification Authority (CA) – Active Directory Certificate Services (ADCS). A CA loosely performs the following roles:

- Receives certificate requests from subscribers
- Identifies and validates aforementioned requests
- Issues certificates according to the PKI's security policy,
- Publishes certificates to specified locations
- Archive a subscriber's private key if suitably configured
- Revoke certificates deemed untrustworthy
- Create and publish Certificate Revocation Lists (CRLs)
- Logs certificate and CRL transactions to its database

The Case for the Offline Root CA

At the heart of PKI is the concept of a *chain of trust*, that is, an organisation puts a *stake in the ground* and explicitly trusts a CA, this is referred to as the root CA. By explicit, what is meant is that the Root CA's certificate (which it issued to itself, i.e. self-signed), is installed into a particular certificate store on every computer in an organisation and consequently every certificate issued by it or CAs which it has subordinated (more of them later) is trusted. Or, it makes more sense inversely, any *end entity* certificate presented to relying party must chain to a trusted Root CA. The concept of the chain of trust is illustrated in Figure 1.

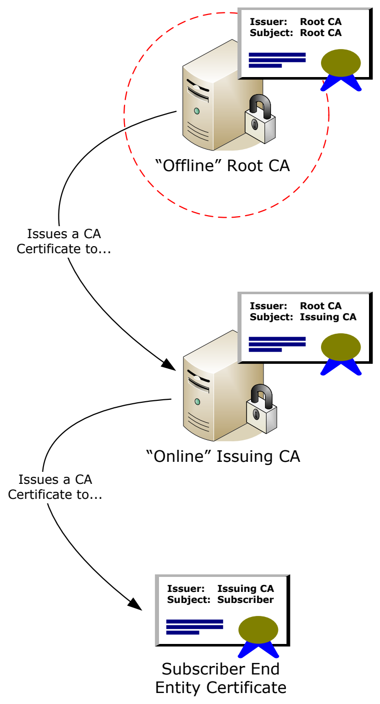


Figure 1: PKI Trust Chain

Note: It is perhaps worth mentioning the concept of explicitly trusted Root CAs is not only relevant to private CAs (ones deployed in an enterprise) but also to commercial CAs which we *kind of* explicitly trust; actually Microsoft made the trust decision on our behalf. You can see this by running the `certmgr.msc`[1] or `certlm.msc`[2] MMC snap-in and browsing the Trusted Root CA container. In fact, whenever you access an SSL protected web site with Internet Explorer, such as the payment pages on www.amazon.com, the web server certificate used as part of the SSL session establishment will have been issued by a CA in the list (or a CA subordinate to one of the CAs in the list). If it wasn't, your browser would complain loudly!

Because of the trust asserted by use of digital certificates (as we have just seen for example, we trust it to protect payment credentials on the internet), it is necessary to ensure the trust chain is suitably protected. And, as we all know, the most secure *box in the network* is the box that isn't on the network! Essentially, Root CAs are deployed in an entirely disconnected state, in the context of an enterprise this generally means on a server built from 'scratch' with Windows source media, into a workgroup and with Network

[1] For 'user' based certificates
[2] For 'computer' based certificates

Interface Cards (NICs) disabled, or preferably removed. Once commissioned, Root CAs have very little operational role to play other than occasionally produce a CRL and their CA certificate generally has a long validity period such as twenty years. So, apart from being a trust anchor, a Root CA is really quite lazy and unsuited to churning out tens / hundreds / thousands pf certificates at subscribers' behest, so like all 'big cheeses', it delegates!

The Issuing CA

The process whereby a Root CA delegates certificate issuance to another CA is known as subordination; the subordinated CA is generally referred to as an 'Issuing CA', in the context that it will be performing the majority of certificate issuance in the enterprise - it is the 'workhorse' of the PKI.

The commissioning of an Issuing CA follows a sequence of operations whereby the Issuing CA generates a CA certificate request (which contains its subject information and public key); this is generally transferred onto removable media and taken to the Root CA. The Root CA certifies the Issuing CA by taking the information in the certificate request and 'wrapping it up' into an x.509 v3 certificate which it digitally signs using its own private key. It is the signature that 'authorises' the Issuing CA to issue certificates which chain up to the Root CA.

Note: The Root CA's private key is a security sensitive object and as we shall see, so is the Issuing CA's private key likewise. Compromise of a CA's private key makes it relatively easy for a hostile party to impersonate the CA and fraudulently issue certificates as that CA. It is the signature on a certificate that imparts the magic, *unbreakable* binding in PKI of a subject to its public key, and it is the CA's private key which performs the signing operation. CA private keys consequently need extra special protection in the form of HSMs which will be described later.

On the Windows Server platform, a CA can be either a 'standalone CA' or an 'enterprise CA', the former providing rudimentary CA services whereas the latter leverages AD to provide richer capabilities. This section on Issuing CAs will concentrate on the enterprise CA capability (whereas a Root CA will almost certainly be deployed in standalone CA mode).

Note: It is feasible to combine both the Root CA role and the Issuing CA role - typically referred to as an Enterprise Root CA. An Enterprise Root CA is perhaps appropriate for a tactical solution or where high assurance in the PKI is not required. Conversely, a PKI hierarchy is not limited to a simple two-tier approach; circumstances may dictate two offline tiers whereby the 'middle CA' is known as an Intermediate (or Policy) CA. This approach is generally more suitable to extremely large and sophisticated PKI implementations; this paper concentrates on the more common two-tier approach.

Figure 2 illustrates how an enterprise CA relies upon AD domain controllers to store objects; and domain joined computers can leverage AD group policy to 'automatically' enrol for certificates.

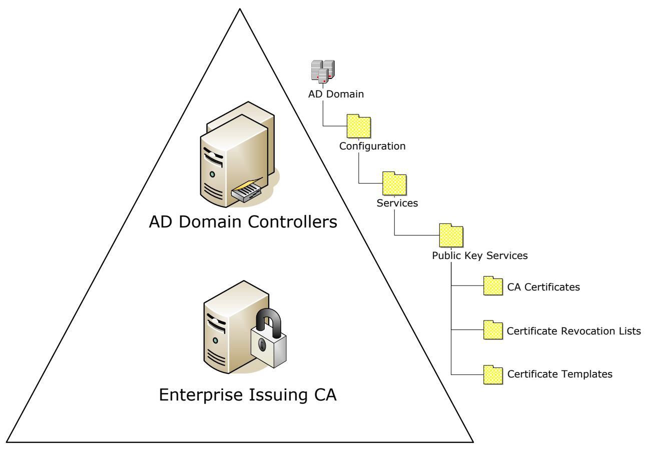
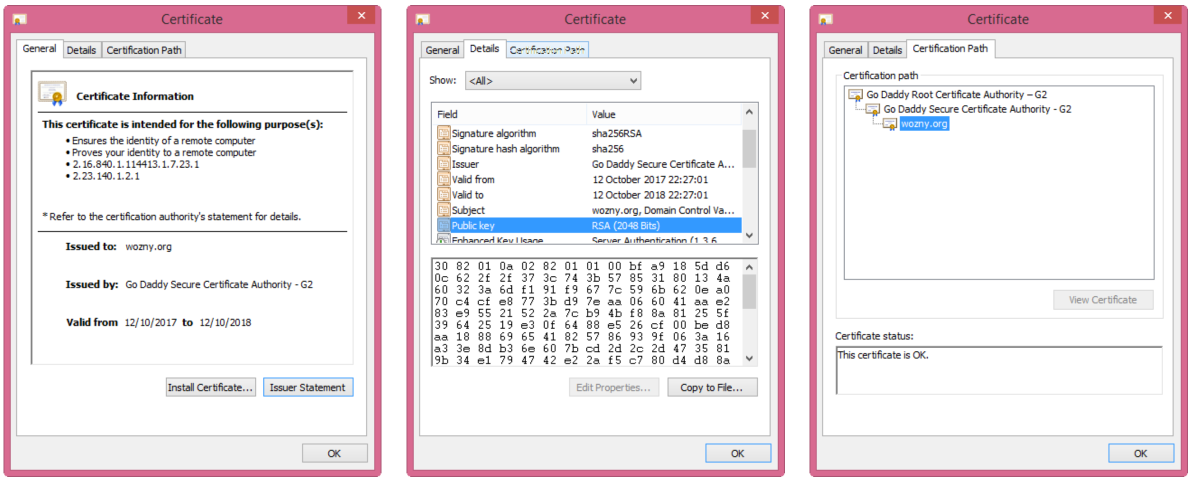


Figure 2: Enterprise CA

One of the most important capabilities of an enterprise CA is the use of certificate templates as the basic definition for certificates which an enterprise CA issues. A certificate template has an Access Control List (ACL) which defines what entities can enrol for certificates; it also contains extensions which effectively determine to what purposes a certificate can be used. Certificate templates also describe how subject information is retrieved and entered into a certificate, for a web server certificate this may well be inserted manually during enrolment (`www.my-domain-name.com`, etc.); or it might be automatically retrieved from AD, an example might be a subject's distinguished name, email name or User Principal Name (UPN). It shouldn't be forgotten that the same principal could be used when enrolling a computer for a certificate, in this circumstance the computer's DNS name might be the appropriate subject of the certificate.

Note: An enterprise CA can be deployed on either Windows Server Standard or Enterprise Edition. If deployed on Standard Edition, a set of pre-defined certificate templates are introduced which cannot be modified, only their ACL can be changed. On Windows Server Enterprise Edition, additional customisable certificate templates can be established and this is often a very important capability. There are other distinctions when deploying an enterprise CA on the different Windows Server editions, such as whether private key archival can be performed; data encryption use of PKI and the corresponding requirement to archive private keys is discussed in the next chapter.

You might be a little interested in looking at one of these digital certificates, and I have included a 'collage' in Figure 3.



Certificate Information

This certificate is intended for the following purpose(s):
- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- 2.16.840.1.114413.1.7.23.1
- 2.23.140.1.2.1

* Refer to the certification authority's statement for details.

Issued to: wozny.org

Issued by: Go Daddy Secure Certificate Authority - G2

Valid from 12/10/2017 **to** 12/10/2018

Field	Value
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	Go Daddy Secure Certificate A...
Valid from	12 October 2017 22:27:01
Valid to	12 October 2018 22:27:01
Subject	wozny.org, Domain Control Va...
Public key	RSA (2048 Bits)
Enhanced Key Usage	Server Authentication (1.3.6

```
30 82 01 0a 02 82 01 01 00 bf a9 18 5d d6
0c 62 2f 2f 37 3c 74 3b 57 85 31 80 13 4a
60 32 3a 6d f1 91 f9 67 7c 59 6b 62 0e a0
70 c4 cf e8 77 3b d9 7e aa 06 60 41 aa e2
83 e9 55 21 52 2a 7c b9 4b f8 8a 81 25 5f
39 64 25 19 e3 0f 64 88 e5 26 cf 00 be d8
aa 18 88 69 65 41 82 57 86 93 9f 06 3a 16
a3 3e 8d b3 6e 60 7b cd 2d 2c 2d 47 35 81
9b 34 e1 79 47 42 e2 2a f5 c7 80 d4 d8 8a
```

Certification path
- Go Daddy Root Certificate Authority – G2
  - Go Daddy Secure Certificate Authority - G2
    - wozny.org

Certificate status:
This certificate is OK.

Figure 3: A Digital Certificate

We can see on the General tab some pertinent summary information, including the issuer (Go Daddy) and subject (wozny.org). We can see the purposes to which the certificate can be used (proves your identity to a remote computer). We couldn't use this certificate to digitally sign mobile code or logon to Windows as those purposes are not specified. Certificates are time bounded, and here we can see its validity period expires in October 2018.

The Details tab includes all of the Fields (attributes / extensions) in the certificate; it should be noted that the subscribers' public key exists as an extension (as can be seen).

Finally, on the Certification Path tab we can observe the chain of trust: the bottom leaf represents the author's certificate, which was issued by the Go Daddy Secure Certificate Authority, which in turn was subordinated by the Go Daddy Root Certificate Authority.

Revocation Status and Providers

So, we've established a CA hierarchy and issued a certificate... but what if that certificate becomes compromised; actually it's the private key corresponding to the certificate which we're actually concerned with – but we'll consider the term 'certificate' for simplicity. Perhaps the certificate is on a smart card that has been stolen or eaten by a dog, or maybe it's on a computer where malware has been discovered that could potentially employ the certificate for malicious purposes.

A mechanism is needed to inform a relying party (let's say an AD domain controller during a smart card based login) to consult with an authoritative source of a user's certificate 'status' before deciding whether it considers the certificate *legitimate*. One of the operations of a Certification Authority is to publish a list of time valid certificates which are deemed untrustworthy (revoked). This is typically achieved using an item referred to as a Certificate Revocation List (CRL); this is simply a file that contains a list of serial numbers of revoked certificates and perhaps a reason for revocation. CRLs are published on a regular basis by CAs - which also digitally sign CRLs to ensure their integrity. The precise publication interval is often a very delicate design decision due to conflicting requirements - as will be seen.

Note: CRLs are only concerned with time valid certificates because an out-of-date certificate would fail on a time validity check anyway, and hence listing on a CRL would be redundant. Indeed, certificates which are listed on a CRL which then expire are automatically purged from the CRL.

So how does a relying party know where to retrieve a CRL from when validating a certificate presented to it? The answer is: another one of those extensions we saw in the Details tab of an issued certificate, known as a CRL Distribution Point (CDP) which contains a list of Universal Resource Locations (URLs). Remember, the CRL is simply a file and it makes sense to publish it to a highly available, and possibly distributed location; almost certainly never the CA itself.

Note: Root CAs also must publish CRLs (which are copied onto removable media and transferred to the CDPs), pertaining to the subordinate CA certificate issued to issuing CAs. Root CA CRL publication is performed on an infrequent basis, such as annually, and not considered further in this document.

Figure 4 shows a CRL being published by an Enterprise CA to both HTTP and LDAP CDPs (A). An AD domain controller retrieves a CRL from one of the CDPs (B) after being presented with a certificate during a smart card based logon authentication (C).

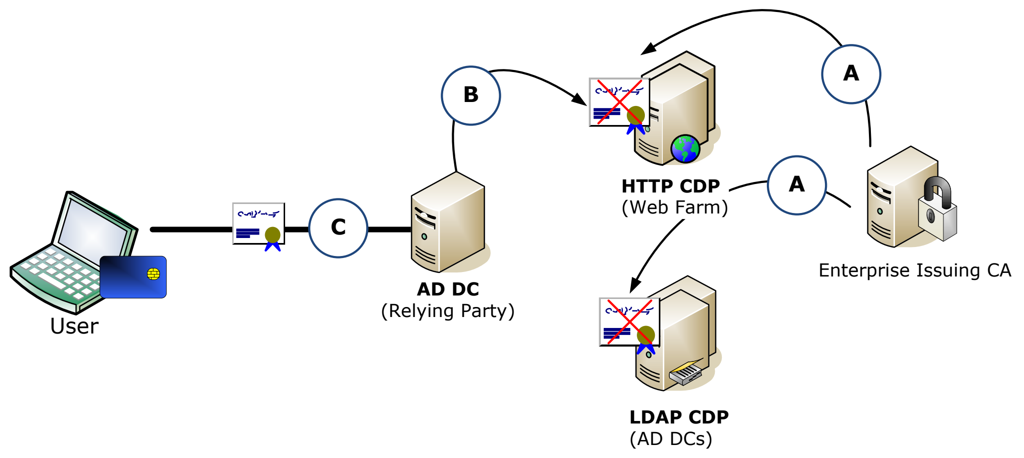


Figure 4: Revocation Providers

When a relying party inspects a certificate presented to it, it will extract the ordered list of CDPs and attempt to retrieve a CRL so that it can determine whether the certificate has been revoked.

Note: If a relying party cannot retrieve a CRL from the list of CDPs, or the CRL is stale, i.e. it has expired, the default behaviour is to reject the certificate (fail safe). Just like a certificate, a CRL has a finite lifetime.

A CDP extension may contain information similar to that shown here (liberty has been taken with the LDAP CDP for the purpose of simplicity):

```
1. http://pki.my-domain-name.local/crl/My Issuing CA.crl
2. ldap:///CN=My Issuing CA,CN=CDP,CN=Public Key Services,DC=MyDomain,DC=local
```

The HTTP CDP is a useful first CDP because it is a 'universally recognised' protocol, often accessible externally in the event that certificates are being used across public networks. It should be noted that publication of CRLs from an Issuing CA to an HTTP CDP will usually involve an 'out-of-band' mechanism such as the RoboCopy file transfer tool. The LDAP CDP is useful because an organisation's AD can be leveraged to provide a highly distributed resource, however, this isn't so accessible externally and the LDAP:/// *syntax* used by AD is not universally recognised.

Note: As mentioned previously, the validity period of a CRL is often a sensitive design matter. Balancing the demand for up-to-date revocation status against the availability concern of a CA failure and non-publication of fresh CRLs causing certificates to be rejected by relying parties is a classic PKI dilemma. Added to the conundrum is the fact that some relying parties cache CRLs during their authoritative period (lifetime) and won't try to retrieve a CRL from a CDP if it has a cached CRL with outstanding validity period. Best practice for CRLs is too wide a subject to consider here, but rest assured they will certainly be a matter of great deliberation, and compromise!

Hardware Security Modules

One of the components of your PKI you won't be buying from Microsoft are HSMs; there are many vendors and form factors of HSMs. Vendors include Thales (nCipher), Gemalto (SafeNet) and AEP; whereas form factors include PCI cards, USB attached modules and network attached modules.

HSMs are employed to secure a CA's private key material (be it Root or Issuing) by implementing physical and logical controls. CA private keys are only ever accessed (generally for signing operations) within the confines of the HSM. The HSM is a 'cryptographically secure' tamper proof and tamper evident device. Logical controls tend to require a *nuclear launch code* approach, whereby multiple people need to present a token (and corresponding PIN) to perform a particular function at the HSM. By way of example, Thales HSM's fundamental logical control (the Security World) and associated tokens are illustrated in Figure 5 - most HSM vendors have a similar approach.

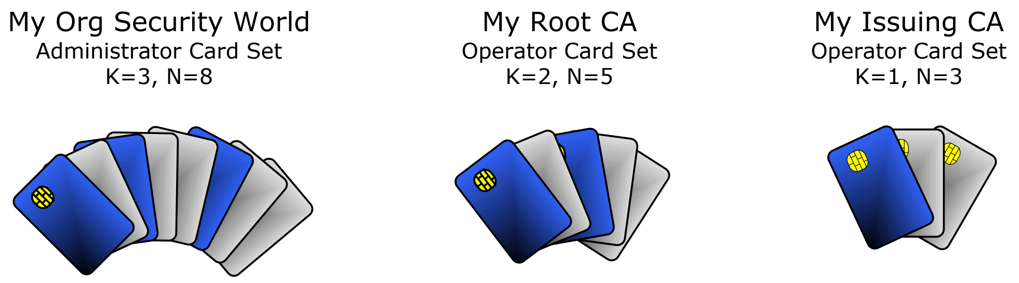


Figure 5: Thales HSM Logical Controls

In the circumstance illustrated above there is an Administrator Card Set (ACS) which typically governs management functions for all of the HSMs in a deployment and have identified a total of eight *individuals* (N) with key management responsibility. Three ACS cards (K) need to 'come together' to effect a task. The term 'K of N' is often used, whereby K is the quorum and N is the total number (of cards). Management tasks might be reloading key material into a replacement HSMs following a failure, or deploying further HSMs for high availability purposes. It should be noted that having a single Security World including all CA servers in a PKI implementation is a design decision, this approach is not mandatory.

The operator card set is used to authorise operational tasks, such as the *activation* of the CA's private key when the CA service is started. For the Root CA we have designated here that a quorum of two card holders must be present to perform this task, whereas for the Issuing CA only one card holder needs to be present. This approach might result in an 'operational constraint' in a situation whereby the Issuing CA is deployed in a lights-out data centre scenario and a power failure causes a reboot of the Issuing CA - approaches to mitigate this risk are not covered in this paper.

Summary

We've covered off the PKI fundamentals and implemented a CA hierarchy and supporting PKI elements, so what...?

Practical Applications of PKI

Introduction

As we have learned, PKI can generally be applied to meet the following security paradigms: Confidentiality, Integrity and Authentication. You'll often also see *non-repudiation* referred to in addition to CIA, though this is generally more about policy to provide assurance of CIA in the context of a legal framework and therefore not addressed separately here.

Some practical applications of PKI include the following:

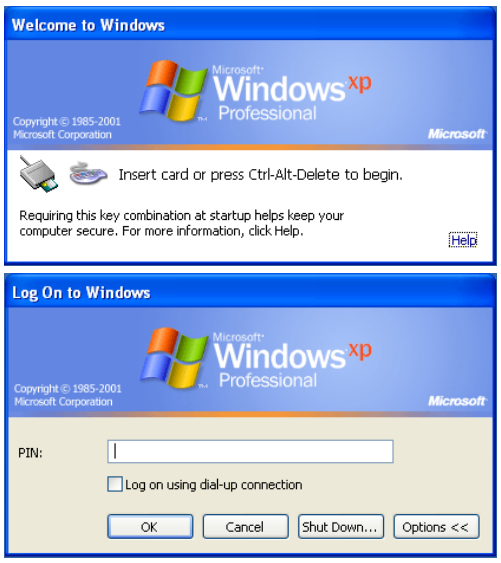
- Signing email (integrity and authentication)
- Encrypting email (confidentiality)
- Code signing (integrity and authentication)
- Web server (authentication and confidentiality)
- Virtual private network (authentication)
- Wireless network (authentication)

This chapter will address typical PKI usage incorporating the authentication and confidentiality paradigms - to give an appreciation of the practical issues and benefits, rather than a purist / theoretical approach.

Two Factor Strong Authentication

Introduction

Probably the most common implementation of ADCS is to support smart card logon to Windows. In this scenario, when your domain joined computer starts up, instead of pressing CTRL+ALT-DEL and entering a user account and password, a smart card is inserted into a smart card reader the smart card PIN is entered when prompted.


Welcome to Windows
Microsoft Windows xp Professional
Copyright © 1985-2001 Microsoft Corporation
Microsoft
Help
Log On to Windows
Microsoft Windows xp Professional
Copyright © 1985-2001 Microsoft Corporation
Microsoft
PIN:
Log on using dial-up connection
OK
Cancel
Shut Down...
Options <<

Insert card or press Ctrl-Alt-Delete to begin.

Requiring this key combination at startup helps keep your computer secure. For more information, click Help.

Figure 6: Smart Card Logon Prompt

It can be seen in Figure 6 (top), that when a smart card reader is attached to a workstation the GINA (logon prompt) changes to include a smart card graphic and new text. In Figure 6 (bottom) the result of having inserted a smart card inserted into a smart card reader is displayed.

Note: There is no requirement to enter a login ID when performing a smart card based login to Windows, the UPN is read from an extension in the digital certificate on the smart card and this is what is presented to the AD domain controller authenticating a user.

Perhaps it is worthwhile to expand on just what is going on, and what the two factors are. The smart card is not a factor, neither is the certificate. The *something you have* is a private key (associated with the certificate) stored on a smart card; the *something you know* is the PIN, this is required for your computer to be able to unlock the container on the smart card where the private key is securely stored.

Note: It's not mandatory to implement a 'Microsoft CA' for issuing smart card logon certificates, third party CA solutions and managed PKI services can achieve the same end.

Native Support

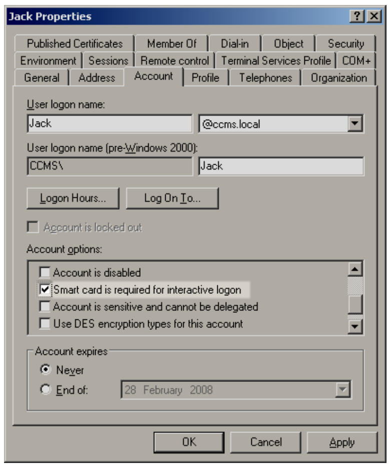
One of the most important aspects of the smart card based two-factor authentication to Windows solution is that it is natively supported and integrated with Kerberos. The process of a Kerberos Key Distribution Centre (KDC) running on an authenticating AD domain controller granting a Kerberos Ticket Granting Ticket (TGT) still applies. The only change is the matter of the password based authentication being substituted by use of cryptographic principals (public and private keys).

Third party 2FA solutions do exist for logging on to Windows but they generally exhibit the following deficiencies:

- The authentication is not Kerberos integrated, often resulting in AD DCs having to *consult* RADIUS servers upon which the third party solution integrates; this can be a scalability, performance and availability constraint
- A requirement to often implement GINA replacements (shims)

Enforcement

It is useful to know that implementing smart card based authentication doesn't automatically mean that it is enforced, that is a separate matter. Enforcement can be applied to computers by AD group policy, such that all <u>computers</u> governed by a GPO require a smart card based logon. More likely, smart card logon will be enforced for a <u>user</u> (this cannot be done by group policy) by a setting on their user account, see Figure 7.



User logon name: Jack @ccms.local

User logon name (pre-Windows 2000): CCMS\ Jack

☐ Account is locked out

Account options:

☐ Account is disabled
☑ Smart card is required for interactive logon
☐ Account is sensitive and cannot be delegated
☐ Use DES encryption types for this account

Account expires

● Never
○ End of: 28 February 2008

Figure 7: Smart Card Logon Enforcement

It should be noted that enforcing smart card based logon randomises the password associated with the user account, hence, username and password type logins are no longer possible.

Some of the challenges presented by enforcing smart card logon are for authenticating Remote Desktop Protocol (RDP) sessions, executing `runas` commands to run a session with elevated privileges and `net use` commands. It may be expected that these capabilities require a username and password… but they don't. Authentication can be re-directed to a smart card, followed by entry of the corresponding PIN.

Other AD settings applicable to smart card use include removal behaviour. Options available are that removal of the smart card from the smart card reader will do nothing (weak), lock the workstation (good) or logoff the workstation (drastic).

Remote Access Gateway Authentication

Although a smart card might be issued primarily for Windows logon, it might also be leveraged for strong authentication to remote access gateways, such as VPN concentrators. Most VPN solution providers have integrated support for this approach, including Microsoft Forefront TMG, CheckPoint VPN-1 and Cisco ACE, etc. These approaches generally leverage the same smart card and certificate, resulting in a *smoother* experience for the user. Going even further, integration of physical access proximity devices in smart card bodies for building access, etc. makes the one card utopia a possibility.

Encrypting File System

Introduction

EFS provides a method to encrypt files on a local, NTFS formatted file system in Windows 2000 and later operating systems. It is possible to specify files, or more likely a folder into which stored files are earmark to be encrypted. Files saved in the marked folder are automatically encrypted and decrypted on access - the underlying 'mechanics' are entirely transparent to the user.

Note: Even though the folder is marked for encryption, it isn't really; it's just an indicator that any documents placed in the folder will be encrypted, file by file.

You may think you're able to use EFS without any PKI since even a Windows XP computer configured in a workgroup can do EFS, and you'd be right... sort of. There's no disputing that EFS does work in this scenario, where your computer goes ahead and creates a self-signed certificate just for this purpose. However, this leads us to the danger apparent in any solution where data encryption is performed... is there a suitable mechanism to recover decryption keys in the event of a 'disaster'? A disaster in this context could be something as seemingly innocuous as a deleted user profile in Windows. As we will see, implementing EFS without suitable recovery measures is akin to feeding your documents into the corporate shredder.

EFS Principals

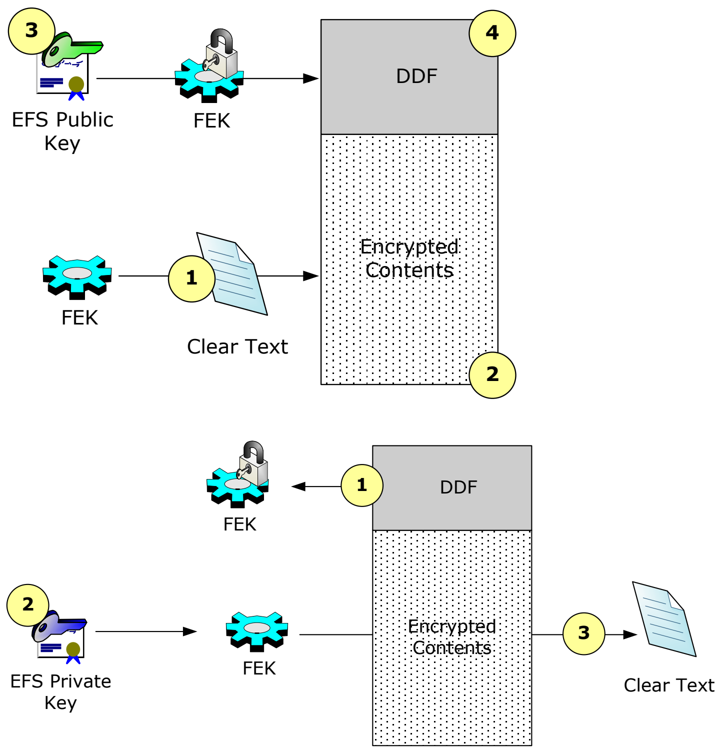


Figure 8: Encrypting File System

Figure 8 (top) illustrates the fundamental principal of EFS encryption. (1) When a file is earmarked for EFS (or placed into an 'EFS marked folder') a symmetric File Encryption Key (FEK) is established and the previously *clear text* contents of the file encrypted (2) with this FEK. (3) The user's EFS certificate is used to encrypt the FEK and the encrypted FEK is placed in the document Data Decryption Field or DDF (4).

Figure 8 (bottom) illustrates the fundamental principal of EFS decryption. On access, the encrypted FEK is retrieved from the document's DDF (1). The user's EFS private key (2) is used to decrypt the FEK and finally the FEK is used to decrypt the body of the document. It should be remembered that all of this is happening *under the covers*, transparent to the user.

Note: Windows XP SP1 and above use a FEK based upon the AES algorithm with a 256-bit key length

EFS Certificate Enrolment

EFS certificates will generally be enrolled using auto-enrolment capability, in this way enrolment is seamless to the user – they just get an EFS certificate. The benefits of this approach are clear, though it might be problematic for users who roam to different machines as they could potentially get auto-enrolled for new EFS certificates at each machine they roam to. One possible means to overcome this problem is credential roaming – by using this technology the user's certificate and key material is stored in their AD account and roams with them wherever they go and available wherever they need them.

Note: It is important if you intend to use EFS that you automatically enrol users for EFS certificates, or at least make it seamless to the user when they first attempt to encrypt a file that an EFS certificate is transparently enrolled for. If a user attempts to encrypt a file and doesn't have a valid EFS certificate, Windows would typically generate a self-signed certificate to perform the encryption. This is a very dangerous scenario, as this certificate would be 'un-managed' and therefore if lost (actually the decryption private key) would result in the data being 'shredded'. It is sensible that until a roper infrastructure is in place to manage EFS, the ability to encrypt files is disabled by group policy.

Another important consideration is users should be relied upon to select folders for encryption (a big risk) or whether it is preferable to use group policy to restrict the folders to which a user can save data and ensure that these folders are flagged for EFS. This way, there's more assurance that data which users save on their local file system is encrypted.

Private Key Archival

In the majority of scenarios where certificates are used for persistent data encryption (such as EFS), it would be expected that an archive of the private (decryption) key is required. An enterprise CA facilitates key archival effortlessly, by defining within the certificate template that the private key should be archived. Recovery of private keys from the CA is a split-role process, requiring participation of a certificate manager and a key recovery agent.

Additionally, EFS has a mechanism that doesn't rely upon key archival at the CA, called EFS data recovery. In this approach, EFS recovery agents are established, and referenced in AD group policy. If any EFS encrypted data needs to be recovered, it would be taken to a workstation where an EFS recovery agent is installed and can subsequently decrypt the data.

With EFS, it is possible to take either approach to security private key material, or perhaps even both! The pros and cons of each approach would need to be explored in greater detail to make an informed design decision.

Note: Even though PKI certificates are used as part of the establishment of an encrypted channel between a web site and a user's browser, this data is only encrypted transiently and therefore there is not necessarily a requirement to archive private keys associated with a Web server's SSL certificate.

EFS in Context

So, in which circumstances might it be attractive to implement EFS? The 'nutshell answer' is where there is a need for a user to ensure confidentiality of data that he/she works with such that no-one else can access that data. Ordinarily this would be for data that is stored on the user's hard disk drive or possibly USB attached storage device (though this would have to be formatted with NTFS).

Note: It is possible for a user to grant access on a file-by-file basis to another user, providing they possess an EFS certificate of their own.

So, does EFS provide the answers to all your confidentiality requirements? Probably not, EFS would generally form part of a range of measures to meet security requirements.

- Can EFS be used to protect all data on your local system? Not really, full volume encryption solutions such as BitLocker may be more appropriate.
- Can EFS be used to protect files as they distributed to various publication locations or transported around on removable media? Not really, Rights Management Services (RMS) might be a better fit.
- Can EFS be used to protect files on network shares? With some trickery, yes. Though again, the RMS approach might be more suitable here.

Does all this mean that EFS has some limits to its applicability? Absolutely. But, it also has extremely strong capabilities and can be a significant component of your security policy – I just want to ensure you're aware that EFS doesn't solve all confidentiality requirements.

Glossary

CA	A Certification Authority is a service which receives requests for certificates (incorporating a subscriber's subject information and public key) and manufactures a digital certificate (x509 v3) incorporating aforementioned attributes
CRL	Certificate Revocation Lists are monolithic objects published (and digitally signed) by CAs containing the serial numbers of certificates designated as untrustworthy; they are retrieved by a relying party from a revocation provider when a certificate is being validated
CSP	A Cryptographic Service Provider is a software library that implements the Microsoft Cryptographic API, generally, smart cards have a specific CSP. The Smart Card Base CSP model, means that specific manufacturer CSPs are becoming redundant.
HSM	A Hardware Security Module is a device which protects sensitive cryptographic material in a physically and logically secure environment; typically implemented in a PCI form factor device
KSP	The Key Storage Provider, which supplanted the CSP in the 'Vista timeframe'.
OCSP	Online Certificate Status Protocol is a bandwidth effective method for relying parties to establish revocation status, whereby the serial number of a certificate being validated is submitted to an OCSP 'responder' which returns a digitally signed response: either positive or negative. OCSP is also often used to overcome latency issues associated with CRLs
RA	A Registration Authority is a service which brokers requests on behalf of a CA, generally handling workflows such as enrolment, renewal, termination etc.; often combined with a smart card management capability
Relying Party	An entity which makes a decision /action based upon the validity of a digital certificate presented to it (e.g. a domain controller is a relying party in the context of a smart card logon)
Revocation Provider	An entity which provides revocation services to a relying party, typically this could be an LDAP or HTTP based service hosting certificate revocation status data
Subscriber	An entity (could be a user, service or a computer) which enrols for a digital certificate; its identity is attested by the binding of its public key and subject information in a digital certificate issued by a Certification Authority trusted either explicitly or implicitly by any relying parties