5

Kerberos

This chapter focuses on the Kerberos authentication protocol, the default authentication protocol of Windows Server 2003. We will look at how the protocol is works, how it has been implemented in Windows Server 2003, and some advanced Kerberos topics.

5.1 Introducing Kerberos

In Greek mythology Kerberos is a three-headed dog guarding the entrance to the underworld. In the context of this book Kerberos refers to the authentication protocol developed as part of the MIT Athena project.[1]

Microsoft introduced Kerberos as the new default authentication protocol for enterprise environments in Windows 2000. Every Windows 2000, Windows XP, and Windows Server 2003 OS platform includes a client Kerberos authentication provider. Neither Windows 2000 nor Windows Server 2003 includes Kerberos support for other legacy Microsoft platforms. Your NT4, Windows 95 or 98 clients will not be able to authenticate using Kerberos—you'll need to upgrade these workstations to either Windows 2000 Professional or Windows XP. In the early days of Windows 2000, Microsoft promised to include Kerberos support for Windows 95 and 98 in the "Directory Services Client" (dsclient.exe), an add-on for Windows 95 and 98 that can be found on the Windows 2000 Server CD.

A little more about the dog's three heads: They stand for authentication, authorization, and auditing. The basic Kerberos protocol (Version 5, as defined in RFC 1510) only deals with authentication. Microsoft's implementation of the protocol also includes extensions for authorization. So far,

1. More historical information on the MIT Athena project is available from the following URL: http://www-tech.mit.edu/V119/N19/history_of_athe.19f.html.

no Kerberos implementation covers auditing. Kerberos can also offer more than the three A's: Later in this chapter we will explain how one of the secret keys exchanged during the Kerberos authentication sequence can be used for packet authentication, integrity, and confidentiality services.

Another analogy to the dog's three heads is the number of basic entities the Kerberos protocol is dealing with. There are always three: two entities that want to authenticate to one another (e.g., a user and a resource server) and an entity that mediates between the two, a trusted third party, or, in Kerberos terminology, the key distribution center (KDC).

5.1.1 Kerberos advantages

In this section, we will explain the key differences between the NTLM and the Kerberos authentication protocols and the advantages that Kerberos brings to the Windows 2000, Windows XP, and Windows Server 2003 operating systems and their users. Many of the terms used in this section will be explained in greater detail later on in this chapter.

Faster authentication

The Kerberos protocol uses a unique ticketing system that provides faster authentication:

- Every authenticated domain entity can request tickets from its local Kerberos KDC to access other domain resources.
- The tickets are considered as access permits by the resource servers.
- The ticket can be used more then once and can be cached on the client side.

When a resource server or the KDC gets a Kerberos ticket and a Kerberos authenticator from the client, the server has enough information to authenticate the client. The NTLM authentication protocol requires resource servers that are not domain controllers, to contact a domain contoller in order to validate a user's authentication request (this process is known as pass-through authentication). Thanks to its ticketing system, Kerberos does not need pass-through authentication. This is why Kerberos accelerates the authentication process. A downside to the ticketing system is that it puts a greater workload on the client. On the other hand, it offloads the resource servers; they must not bother about pass-through authentication anymore.

Mutual authentication

Kerberos supports mutual authentication. This means that the client authenticates to the service that is responsible for the resource and that the service also authenticates to the client. This is a big difference from NTLM. The NTLM challenge-response provides only client authentication: The server challenges the client, the client calculates a response, and the server validates that response. Using NTLM, users might provide their credentials to a bogus server.

Kerberos is an open standard

Microsoft based its Kerberos implementation on the standard defined in RFC 1510 (this is Kerberos Version 5). This is why Kerberos can provide single sign-on (SSO) between Windows Server 2003 and other OSs supporting an RFC 1510-based Kerberos implementation. RFC 1510 can be dowloaded from the Internet Engineering Task Force (IETF) at http://www.ietf.org.

Over the past years, Microsoft has been actively involved in the Kerberos standardization process. Microsoft software engineers participated in the creation of several Kerberos-related Internet drafts (see also http://www.ietf.org).

Support for authentication delegation

Authentication delegation can be looked at as the next step after impersonation: Thanks to impersonation, a service can access local resources on behalf of a user; thanks to delegation, a service can access remote resources on behalf of a user. What delegation really means is that user A can give rights to an intermediary machine B to authenticate to an application server C as if machine B was user A. This means that application server C will base its authorization decisions on user A's identity rather than on machine B's account. Delegation is also known as authentication forwarding. In Kerberos terminology this basically means that user A forwards a ticket to intermediary machine B, and that machine B then uses user A's ticket to authenticate to application server C.

You can use delegation for authentication in multitier applications; an example of such an application is database access using a Web front end. In such a setup the browser, the Web server, and the database server are all running on different machines. In a multitier application, authentication happens on different tiers. In such application if you want to set authorization on the database using the user's identity, you should be capable of using the user's identity for authentication both on the web server and the database server. This is impossible if you use NTLM for authentication on every link, simply because NTLM does not support delegation. We will come back to delegation in greater detail later on in this chapter.

Support for the smart card logon feature

Through the Kerberos PKINIT extension, both Windows 2000 and Windows Server 2003 include support for the smart card logon feature. The smart card logon feature provides much stronger authentication than the password logon feature does because it relies on a two-factor authentication: To log on, a user needs to possess a smart card and know its PIN code. The smart card logon feature also offers other security advantages; for example, it can block Trojan horse attacks that try to grab a user's password from the system memory.

Table 5.1 *Kerberos–NTLM Comparison*

	NTLM	**Kerberos**
Cryptographic technology	Symmetric cryptography	Basic Kerberos: symmetric cryptography Kerberos PKINIT: symmetric and asymmetric cryptography
Trusted third party	Domain controller	Basic Kerberos: domain controller with KDC service Kerberos PKINIT: domain controller with KDC service and Enterprise CA
Microsoft-supported platforms	Windows 95, Windows 98, Windows ME, Windows NT4, Windows 2000, Windows XP, and Windows Server 2003	Windows 2000, Windows XP, and Windows Server 2003*
Features	Slower authentication because of pass-through authentication	Faster authentication because of unique ticketing system
	No mutual authentication	Mutual authentication
	No support for delegation of authentication	Support for delegation of authentication
	No support for smart card logon feature	Support for smart card logon feature
	Proprietary Microsoft authentication protocol	Open standard
	No protection for authorization data carried in NTLM messages†	Cryptographic protection for authorization data carried in Kerberos tickets

* Remember from the previous chapter that Kerberos can only be used for domain logon to a Windows 2000 or Windows Server 2003 domain.

† This was the case for NTLM version 1; this problem has been resolved in NTLM version 2.

5.1.2 Comparing Kerberos to NTLM

Table 5.1 compares Kerberos, the default authentication protocol of Windows 2000 and Windows Server 2003, to NTLM, the default authentication protocol of NT4. It also lists the main features of both protocols introduced in the previous sections.

5.2 Kerberos: The basic protocol

The following sections explain the basic Kerberos protocol as it is defined in RFC 1510. Those not familiar with Kerberos may be bewildered by the need for numerous diverse keys to be transmitted around the network. In order to break down the complexity of the protocol, we will approach it in five steps:

- Step 1: Kerberos authentication is based on symmetric key cryptography.
- Step 2: The Kerberos KDC provides scalability.
- Step 3: A Kerberos ticket provides secure transport of a session key.
- Step 4: The Kerberos KDC distributes the session key by sending it to the client.
- Step 5: The Kerberos Ticket Granting Ticket limits the use of the entities' master keys.

Before starting to explore how Kerberos works, we must explain the notations that will be used in the illustrations:

- The u stands for user, s stands for resource server, and k stands for KDC.
- S stands for session key; Sus means the session key shared between the user and the resource server.
- M stands for master key; Mu is the master key of the user.
- Drawing (1) in Figure 5.1 represents the session key shared between the user and resource server.

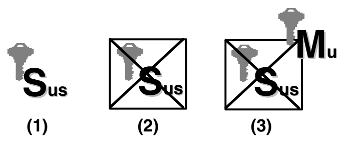


Figure 5.1 *Session keys and encrypted session keys.*

- Drawing (2) represents the same session key, but this time encrypted.
- Drawing (3) represents the same session key, encrypted using the master key of the user.

To ease reading we will talk about a "client," Alice, and a "resource server" that authenticates using Kerberos. The identities used in this Kerberos authentication exchange are Alice's SID and the SID of the service account that is used by the application or the service responsible for the resource. To be fully correct we should talk about the "service account of the service," but this would not promote ease of reading. Also, when we talk about Alice, we really mean the LSA on Alice's machine impersonating Alice and acting on her behalf. From now on, the following words are synonyms: principal and security principal and entity, and domain and realm.

5.2.1 Kerberos design assumptions

Before diving into the nuts and bolts of the protocol, let's have a quick look at some of the design assumptions the Kerberos designers at MIT took. It is very important to keep these assumptions in mind as we run through the Kerberos internals.

- Kerberos always deals with three entities: users, servers, and a set of security servers that mediate between the users and the servers for authentication.
- Time is trusted. This is because Kerberos uses timestamps to protect against replay attacks.
- The user trusts its workstation completely. This is because Kerberos caches authentication tokens on the client side.
- The security server must be online all the time. Kerberos requires the availability of the security server in order to generate new Kerberos security tokens.
- The servers are stateless. Kerberos wants to limit the amount of security principal–related information that is kept on the server side.
- Users' password time on user machine must be minimized. Kerberos looks at a user password as a weak secret—it should be protected the best possible. One of the ways to do this is to limit its time on the user workstation. Another way is to create a key hierarchy.

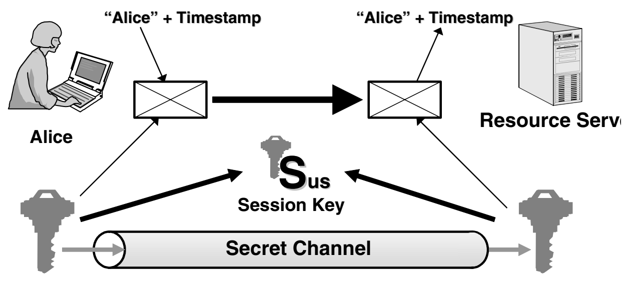


Figure 5.2 *Kerberos authentication is based on symmetric key cryptography.*

5.2.2 Step 1: Kerberos authentication is based on symmetric key cryptography

To authenticate entities Kerberos uses symmetric key cryptography.[2] In symmetric key cryptography the communicating entities use the same key for both encryption and decryption. The basic mathematical formula behind this process is the following:

$D_K(E_K(M)) = M$

If the encryption (E) and decryption (D) processes are both using the same key K, the decryption of the encrypted text (M) results in the readable text (M).

This is what happens when Alice wants to authenticate to a resource server using a symmetric key cipher (illustrated in Figure 5.2):

- Alice encrypts her name and the current timestamp using a symmetric key.
- The encrypted message and Alice's name are sent to the resource server.
- The resource server decrypts the message.
- The resource server checks Alice's name and the timestamp (this is the result of the decryption process). If they are okay, Alice is authenticated to the server.

Why does this process authenticate Alice to the resource server? If the resource server can successfully decrypt the message, this means if the decryption process results in Alice's name and an acceptable timestamp, the resource server knows that only Alice could have encrypted this information,

2. When the Kerberos design was started, public key cryptography was still patented (by RSA). This explains why the default Kerberos protocol (as defined in RFC 1510) relies on symmetric key cryptography.

because she is the only one, besides the resource server, who also knows the symmetric key. In this context acceptable means the following: Upon receipt of Alice's encrypted packet, the resource server will compare the timestamp in Alice's packet against the local time. If the time skew between these two timestamps is too big, the resource server will reject the authentication attempt, because a hacker could have replayed Alice's original authentication packet.

In this explanation you may have noticed the differences and similarities with the NTLM authentication protocol. Both Kerberos and NTLM use symmetric cryptography for authentication: "If you can prove you know your secret key, I believe you are who you say you are." In NTLM the knowledge of the secret key is proven using a challenge-response mechanism. Kerberos uses symmetric encryption of the timestamp and the user's name to do the same thing.

The encrypted packet containing Alice's name and the timestamp is known in Kerberos as the authenticator, and the symmetric key is called a session key. A session key exists between all Kerberos principals that want to authenticate to each other.

A critical element in this exchange is the timestamp: It provides "authenticator uniqueness" and protects against replay attacks. Without the authenticator, a hacker could grab a ticket off the network and use it to impersonate Alice to a resource server. The timestamp explains the time sensitivity of the Kerberos protocol and of Windows 2000 and Windows Server 2003.

Remember from the introduction that Kerberos can provide "mutual" authentication: To provide this the Kerberos protocol includes an additional exchange that authenticates the server to the client. In this example, it means that, in turn, the server will encrypt its name and the current timestamp and send it to Alice.

A big problem when using a symmetric protocol is the secure distribution of the secret key. The secret key is generated at one side of the communication channel and should be sent to the other side of the communication channel in a secure way. Secure means that the confidentiality and integrity of the key should be protected. If anybody could read the secret key when it is sent across the network, the whole authentication system becomes worthless: The secrecy of the secret key is a vital part of a symmetric cipher.

Steps 2, 3, and 4 explain how the Kerberos developers have resolved the problem of secure session key distribution.

5.2.3 Step 2: The Kerberos KDC provides scalability

The Kerberos protocol always deals with three entities: two entities that want to authenticate to one another and one entity that mediates between these two entities for authentication: the key distribution center (KDC). Why do we need a KDC?

Suppose that Alice is part of a workgroup consisting of five entities that all want to authenticate to one another using symmetric key cryptography. Because every entity needs to share a secret key with every other entity, we will need 10 keys. The mathematical formula behind this is n (n – 1)/2. In a 50,000-employee company we would need about 1.5 billion keys. Not only would we have to deal with an enormous amount of keys, but there would also be an enormous amount of small authentication databases: On every client there would be one, containing all the secret keys of the entities with which the client wants to authenticate. This solution is clearly not scalable to the level of a big environment. Imagine having to use such a solution on the Internet.

To make Kerberos more scalable, the Kerberos developers included the concept of a KDC. KDC is a trusted third party with which every entity shares a secret key: This key is called the entity's master key. All entities trust the KDC to mediate in their mutual authentication. The KDC also maintains a centralized authentication database containing a copy of every user's master key.

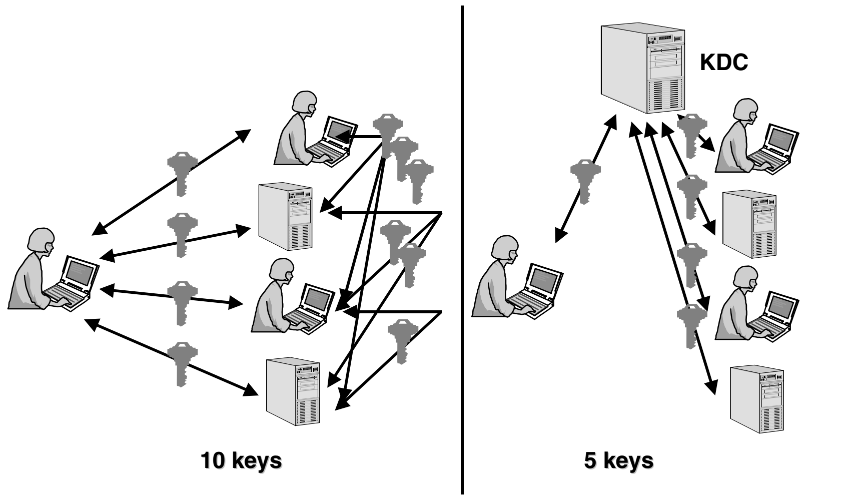


Figure 5.3
A KDC provides scalability.

In Windows Server 2003 the KDC is a service that is installed on every domain controller as part of the dcpromo Active Directory installation process. Every Windows Server 2003 domain controller runs a KDC service[3] and hosts an Active Directory (AD) instance, the central authentication database.[4] As a consequence, a domain with multiple domain controllers provides fault tolerance for the authentication process and the authentication database. If one DC is down, another one can automatically take over. Also, the AD authentication database is replicated between domain controllers.

The concept of a master key is not new to Windows 2000, Windows Server 2003, and Kerberos: It already existed in NT4 and earlier Windows versions. In Windows the master key is derived from a security principal's password.

The password is a secret key that is shared between each individual security principal and the central authentication authority (in the Kerberos case the KDC). Both the entity and the KDC must know the master key before the actual Kerberos authentication process can take place. For obvious security reasons, the AD never stores the plain password but a hashed version (the hash algorithm used is MD4).

An entity's master key is generated as part of the domain enrollment process (e.g, when the administrator enrolls the user and enters a password). A machine's master key is derived from the machine password that is automatically created when an administrator joins the machine into a domain.

5.2.4 Step 3: The ticket provides secure transport of the session key

Figure 5.4 shows the three basic entities with which the Kerberos protocol deals: a client (Alice), a resource server, and a KDC. Figure 5.4 also shows the master keys that are shared between the entities participating in the authentication process and the KDC.

Remember that in the first step we talked about the problem of distributing the secret key (the session key) when dealing with symmetric key ciphers. This section explains how Kerberos resolves this problem. It makes

3. The KDC itself is made of two subservices: the Authentication Service (AS) and the Ticket Granting Service (TGS); in other Kerberos implementations these two subservices can run on different machines, but this is not possible in Windows 2000 and Windows Server 2003.
4. On an interesting note, a standard Kerberos domain is made up of a master KDC and one or more slave KDCs. The master KDC is collocated with a read-write copy of the authentication database (single-master model). In Windows 2000 and Windows Server 2003 every KDC server hosts a read-write copy of the domain portion of the Active Directory (multi-master model).

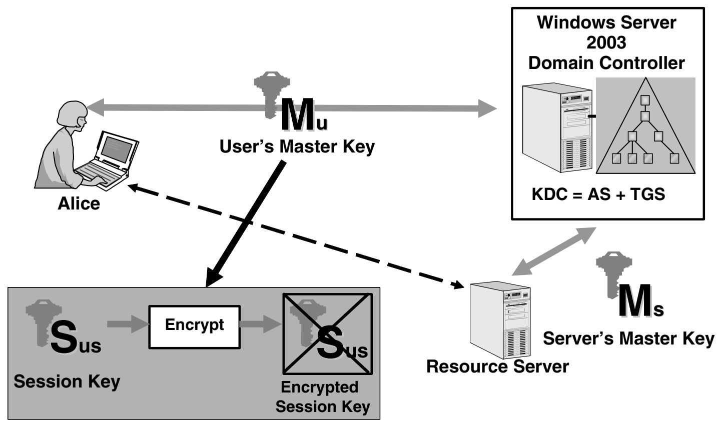


Figure 5.4 *Kerberos entities and master key concept.*

the link between the session key and the master key (introduced in step 2) and explains why we really need a master key in the Kerberos protocol.

In Section 5.2.3, we explained that every entity shares a master key with the KDC. We also said that all entities trust the KDC to mediate in their mutual authentication. Trust in this context also means that every entity trusts the KDC to generate session keys. In the scenario shown in Figure 5.4, the resource server would never trust Alice to generate session keys, because Alice has not authenticated yet to the resource server (the other way around would not work either).

So far, so good. Alice needs to authenticate to the resource server and requests a session key from the KDC. The KDC will generate the session key[5] and distribute it to both entities. After the KDC has generated the session key, it must communicate it to both Alice and the resource server. To secure the transport of the session key to a particular entity, Kerberos encrypts it with the master key of that entity.

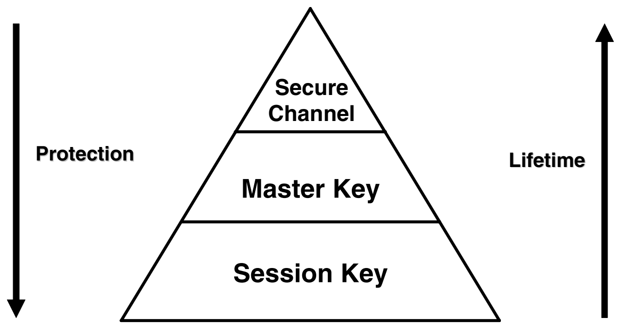
Because there are two entities, Alice and the resource server, two encrypted versions of the session key must be generated:

- One encrypted with Alice's master key
- One encrypted with the master key of the resource server

In Kerberos terminology, the session key encrypted with the resource server's master key is known as a "ticket." A Kerberos ticket provides a way to transport a Kerberos session key securely across the network. Only the

5. The security quality of the session key depends on the quality of the random number generator used to generate the session key.

Figure 5.5
Windows Server 2003 key hierarchy.



destination resource server and a Windows Server 2003 domain controller can decrypt it.

By securing the transport of the session key using the master key, Kerberos creates what is known as a key hierarchy. Figure 5.5 shows the Windows Server 2003 key hierarchy, which consists of:

- *The session key (or short-term key):* A session key is a secret key that is shared between two entities for authentication purposes. The session key is generated by the KDC. Because it is a critical part of the Kerberos authentication protocol, it is never sent in the clear over a communication channel: It is encrypted using the master key.
- *The master key (or long-term key):* The master key is a secret key that is shared between each entity and the KDC. It must be known to both the entity and the KDC before the actual Kerberos protocol communication can take place. The master key is generated as part of the domain enrollment process and is derived from a user, a machine, or a service's password. The transport of the master key over a communication channel is secured using a secure channel.
- *The secure channel:* When Windows is using a secure channel, it is using a master key to secure the transport of another master key. The following example illustrates the secure channel concept. When you create a new user, the user's password will be sent to the domain controller using a secure channel. The secure channel is in this case made up of the master key shared between the workstation you're working on and the domain controller. In other words, the master key is derived from the workstation's machine account password. The concept of a secure channel was also explained in Chapter 2.

In this key hierarchy the following are also true:

- Higher-level keys protect lower-level keys.

- Higher-level keying material has a longer lifetime than lower-level keying material.
- Lower-level keying material is used more frequently for sending encrypted packets across the network. As a consequence, there is a higher risk for brute-force attacks on these packets. In other words, the associated keys should be changed more often.

5.2.5 Step 4: The KDC distributes the session key by sending it to the client

The KDC can distribute the encrypted session keys to Alice and the resource server in two ways:

- Method 1: The KDC could send it directly to both Alice and the resource server (as shown in Figure 5.6).
- Method 2: The KDC could send the two encrypted session keys to Alice. Alice could then send out the resource server's encrypted copy of the session key later on in the Kerberos authentication sequence (as shown in Figure 5.7).

  The first method has the following disadvantages:

- The resource server has to cache all the session keys: one session key for each client that wants to access a resource on the server. This would impose a huge security risk on the server side.
- Synchronization problems could occur: The client could already be using the session key, while the resource server has not even received its copy yet.

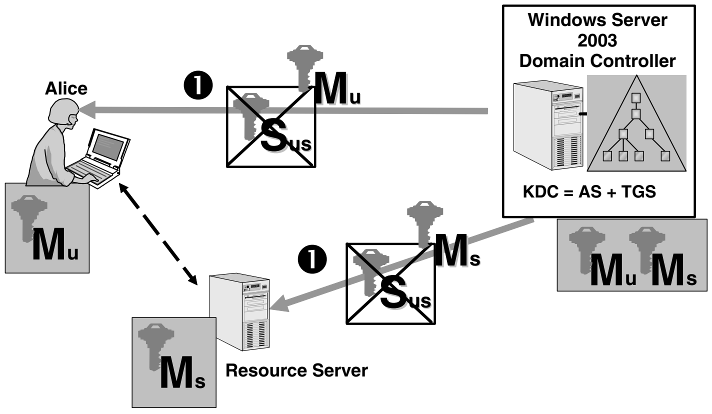


Figure 5.6 *Kerberos ticket distribution Method 1.*

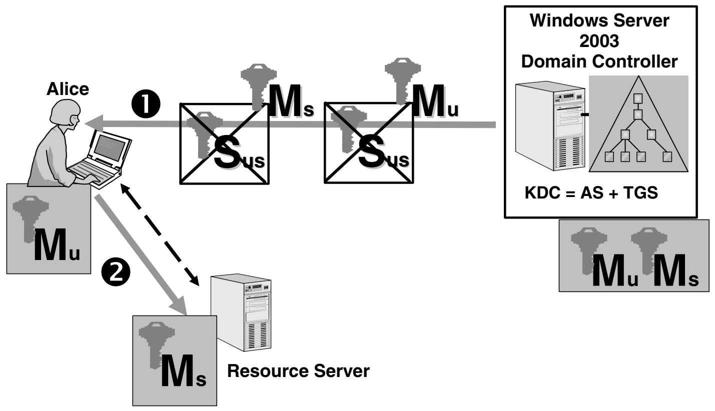


Figure 5.7 *Kerberos ticket distribution Method 2.*

Because of the disadvantages associated with Method 1, Kerberos uses the alternative explained next as Method 2 (see Figure 5.7):

- Both the encrypted session keys (the one for Alice, encrypted with Alice's master key, and the one for the resource server, encrypted with the resource server's master key) are sent to Alice.
- Alice can decrypt the packet encrypted with her master key and get out the session key. Alice's system can now cache both Alice's copy of the session key and the server's copy of the session key (contained in the ticket).
- When Alice needs to authenticate to the resource server, the client will send out the server's copy of the session key.

The key advantage of Method 2 lies in its unique caching architecture: Alice's machine can cache tickets and reuse them. Also, it takes away the need for the server to cache the tickets: It receives them from the clients as needed. This architecture makes the Kerberos protocol stateless on the server side. This has obvious advantages if you want to implement a load balancing or redundancy solution on the server side: There is no need to bother about keeping the session keys synchronized between the different domain controllers.

On the client side, tickets are kept in a special system memory area, which is never paged to disk. The reuse of the cached tickets is limited because of a ticket's limited lifetime and renewal time. Windows 2000, XP, and Windows Server 2003 maintain a ticket cache for every security principal logon session. The ticket cache is purged when the logon session ends. The cache is preserved and written to disk when a system goes into hibernation mode.

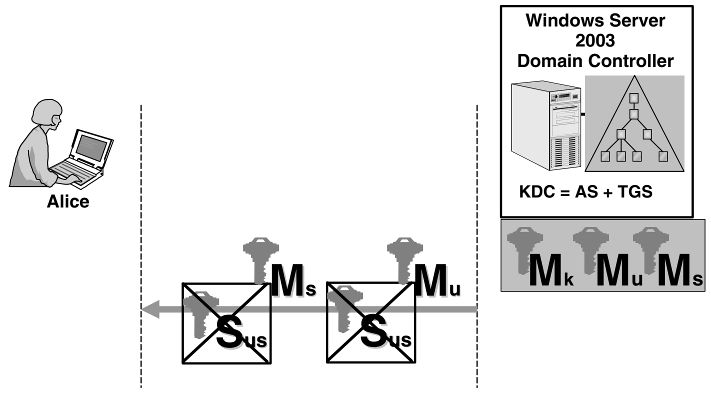


Figure 5.8 *The use of the master key.*

5.2.6 Step 5: The Ticket Granting Ticket limits the use of the master keys

There is yet another important weakness in the protocol that we have not addressed so far: The session key that is sent back from the KDC to Alice is encrypted using Alice's master key (as shown in Figure 5.8). This encrypted packet is sent over the network every time Alice needs a session key to authenticate to a resource server. This means that every time there is an opportunity for hackers to intercept the encrypted packet and to perform—possibly offline—a brute-force attack on the encrypted packet and derive the user's master key.

In a brute-force attack, a hacker tries to guess the key that was used to encrypt a packet by trying out all the possible keys and looking at the result. Such attacks are not unrealistic: Remember some of the tools that were mentioned in Chapter 2 (L0phtcrack, John the Ripper…) to do brute-force attacks on the SAM or AD database or on the authentication packets sent across the network?

There is clearly a need here for a strong[6] secret to replace Alice's master key: This will be the role of the session key that is shared between each entity and the KDC. This session key will replace Alice's password, and it will be used to authenticate Alice to the KDC after the initial authentication.[7]

6. Strong means less susceptible to brute-force attacks. To resist these attacks there are two possibilities: (1) Use longer keys—longer keys create bigger key spaces and make it more difficult to guess the right key; (2) change the keys more often—this limits the chance for brute-force attacks. In other words, limit the lifetime of the keys; this principle is often referred to as perfect forward secrecy (PFS). The Kerberos developers have chosen the latter solution.
7. There are also other reasons why the concept of a session key shared between every Kerberos entity and the KDC is important: it allows for the KDC's AS and TGS services to be hosted on different machines (this cannot be done in Windows, but is often done in UNIX Kerberos implementations), and it enables cross-domain authentication referrals (explained later).

Although it has an identical function (authentication), the session key introduced in this step is *not* the same as the one used in the previous sections. This session key is shared between Alice and the KDC, and the other session key is shared between Alice and the resource server.

Just like Alice's master key, both Alice and the KDC must know this session key. To securely transport this session key, we will use the same mechanism as the one described in steps 3 and 4:

- Step 3: Kerberos uses a ticket to provide secure transport of the session key. The special ticket used here is known as the Ticket Granting Ticket (TGT).
- Step 4: Kerberos distributes the tickets by sending them out via Alice. The KDC sends the TGT to Alice; Alice caches the TGT and can send it to the KDC when needed. Again, there is no need for the KDC to cache the TGTs of every client, and once more this makes the Kerberos protocol stateless on the server side. If you do not consider these arguments, it may sound silly that the KDC generates the session key and sends it out to the client to get it back from the client at a moment later in time.

Figure 5.9 shows how this new session key (S_{ku}) and the associated TGT are used in the basic Kerberos protocol exchange:

- Alice sends a logon message to the domain controller. This message is secured using Alice's master key, derived from Alice's password.[8]
- The KDC will then send out a secured copy of the session key to be used for authentication between Alice and the KDC for the rest of the logon session (this session key will replace the user's master key). The copy of the session key encrypted with the KDC's master key is called the TGT.
- The session key and the TGT will be cached in Alice's local Kerberos ticket cache.
- Later on, when Alice wants to access a resource on the resource server, the security process acting on Alice's behalf will send out a request for a ticket to the KDC using the locally cached TGT. The request for the resource will be secured using the session key S_{ku}.
- Finally, the KDC will send back a ticket and a new session key to Alice, which she can use later on to authenticate to the resource

8. The encryption of this request is not a part of the basic Kerberos protocol as defined in RFC 1510; it is based on a Kerberos extension known as Kerberos preauthentication. It will be explained later on in this chapter.

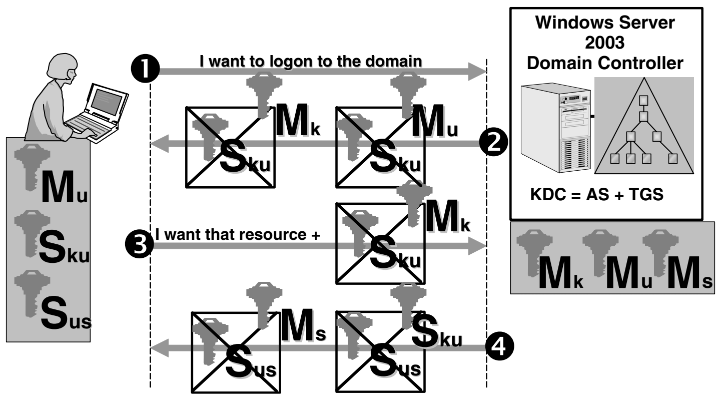

Windows Server 2003 Domain Controller
KDC = AS + TGS
1 I want to logon to the domain
2
3 I want that resource +
4
Mk
Mu
Ms
Sku
Sus

Figure 5.9 *The role of the Kerberos TGT.*

server. Notice that in Figure 5.9 the new session key is not encrypted using Alice's master key, but using the newly created session key S_{ku}.

This sequence shows how in the basic Kerberos exchange:

- The TGT is reused to request tickets for other application or resource servers. The reuse of the TGT is limited by its lifetime. The lifetime of the TGT not only limits the usage of the TGT itself but of all the tickets that were obtained using a particular TGT. For example, if I have a TGT that is about to expire in a half-hour, every new ticket I get will also expire at the same point in time (even though the default lifetime of a ticket may be one hour).
- Ticket requests do not require further use of the client's master key.[9] During the logon session, a weak secret (the master key derived from a client's password) is exchanged for a strong secret (the session key contained within the TGT). In other words, at logon time and at each TGT renewal the user will authenticate to the KDC with his master key; in subsequent ticket requests he will authenticate using the session key, which is contained in the TGT.
- The newly created session key (S_{ku}) doesn't need to be cached on the KDC: The KDC gets it from the client each time the client requests a new service ticket. S_{ku} is encrypted using the KDC's master key (remember: this is what they call the TGT). This feature makes Kerberos stateless on the KDC side, which has, as for resource serv-

9. This also means that once you have a session key, in a standard Kerberos implementation there's no more need to cache the master key on the client, which is very good from a security point of view. Microsoft Windows 2000, XP, and Windows Server 2003 still cache the master key because they need it to perform NTLM authentication to downlevel clients.

ers, obvious advantages if some load balancing or redundancy technology has to be implemented on the KDC side.

5.2.7 Bringing it all together

In this section we will bring together all the elements that were brought up in the previous five steps. Figure 5.10 shows the complete Kerberos protocol: It consists of three subprotocols (or phases), each one made up of two steps. In the following list, the cryptic names between parentheses are the names of the Kerberos protocol messages as they are called in the Kerberos standard documents.

- Phase 1: Authentication Service Exchange (occurs once for every logon session)
  - Step 1: Authentication Server Request (KRB_AS_REQ). Alice logs on the domain from her local machine. A TGT request is sent to a Windows KDC. Ntsecurity.nu makes available two tools (kerbsniff and kerbcrack) to retrieve a user's password from the KRB_AS_REQ network exchange. To capture the KRB_AS_REQ packets, use kerbsniff. Then use kerbcrack to perform a brute-force or dictionary attack on the captured packets. The tools can be downloaded from http://www.ntsecurity.nu/toolbox/kerbcrack/.
  - Step 2: Authentication Server Reply (KRB_AS_REP). The Windows KDC returns a TGT and a session key to Alice.
- Phase 2: Ticket-Granting Service Exchange (occurs once for every resource server)
  - Step 3: TGS Request (KRB_TGS_REQ). Alice wants to access an application on a server. A ticket request for the application server is sent to the Windows KDC. This request consists of Alice's TGT and an authenticator.
  - Step 4: TGS Reply (KRB_TGS_REP). The Windows KDC returns a ticket and a session key to Alice.
- Phase 3: Client-Server Authentication Exchange (occurs once for every server session)
  - Step 5: Application Server Request (KRB_AP_REQ). The ticket is sent to the application server. Upon receiving the ticket and the authenticator, the server can authenticate Alice.

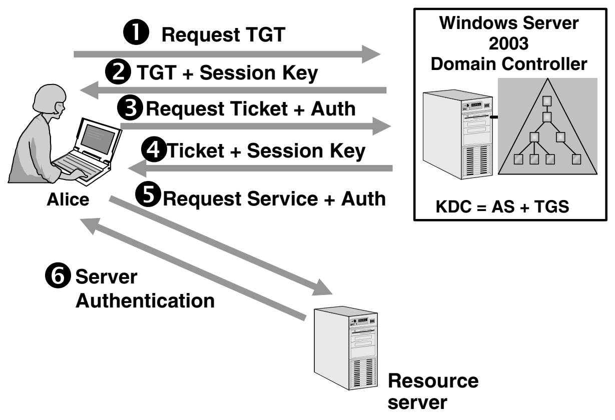


Figure 5.10 *The complete Kerberos protocol.*

- Step 6: Application Server Reply (KRB_AP_REP). The server replies to Alice with another authenticator. On receiving this authenticator, Alice can authenticate the server.

During these exchanges the following keys and tickets are cached on Alice's computer: the TGT, the ticket used to authenticate to the resource server, and two session keys—one to authenticate to the KDC and one to authenticate to the resource server.

5.2.8 Kerberos data confidentiality, authentication, and integrity services

Windows 2000, XP, and Windows Server 2003 all include the Kerberos extensions that can be used to provide data confidentiality, authentication, and integrity for messages that are sent after the initial Kerberos exchange outlined in the previous sections. These extensions are known as the KRB_PRIV (providing data confidentiality) and the KRB_SAFE (providing data authentication and integrity) Kerberos extensions. They are based on the existence of a session key between two entities at the end of a Kerberos authentication protocol exchange:

- The session key can be used to sign a message. A hash, which is the result of applying a hash function to a message, can be encrypted using the session key. A hash encrypted with a session key is also referred to as a message authentication code (MAC).
- The session key can be used to seal a message by encrypting the message using the session key.

5.3 Logging on to windows using Kerberos

Now that we have explained the basic Kerberos protocol, we can discuss some real-world Windows Kerberos logon examples. In this section we will look in detail at both local and network logon features in single and multiple domain environments and in a multiple forest scenario.

5.3.1 Logging on in a single domain environment

Typical examples of logon method in a single domain environment are:

- Alice is logging on from a machine that is a member of the domain where Alice's user account has been defined (this is a local logon method).
- Alice accesses a resource located on a machine that is a member of Alice's logon domain (this is a network logon method).

Local logon process

Figure 5.11 shows what happens during a local logon process in a single domain environment.

Everything starts when Alice presses <CTRL><ALT><DEL> and chooses to log on to the domain.

1. The client Kerberos package acting on behalf of Alice tries to locate a KDC service for the domain; it does so by querying the DNS service.[10] The Kerberos package will retry up to three times to contact a KDC. At first it waits 10 seconds for a reply; on each retry it waits an additional 10 seconds. In most cases a KDC service for the domain is already known. The discovery of a domain controller is also a part of the secure channel setup that occurs before any local logging on.
2. Once the DC is found, Alice sends a Kerberos authentication request to the DC. This request authenticates Alice to the DC and contains a TGT request (KRB_AS_REQ).
3. The Authentication Service authenticates Alice, generates a TGT, and sends it back to the client (KRB_AS_REP).

10. Windows 2000 and Windows Server 2003 publish two Kerberos-specific SRV records to DNS: _kerberos and _kpasswd. The list of all published SRV records can be found on a domain controller in the "%windir%\system32\config\netlogon.dns" file. The SRV DNS records are created automatically during the domain controller setup (as part of the dcpromo process).

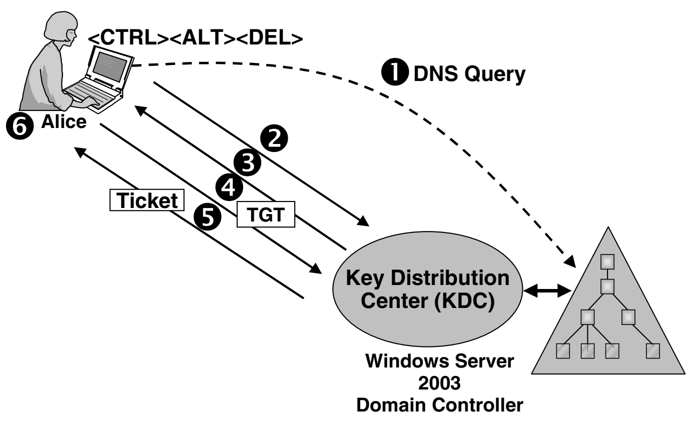

<CTRL><ALT><DEL>
❶ DNS Query
❻ Alice
❷
❸
❹
❺
Ticket
TGT
Key Distribution Center (KDC)
Windows Server 2003 Domain Controller

Figure 5.11 *Local logon process in a single domain environment.*

4. The local machine where Alice logged on is—just like any other resource—a resource for which Alice needs a ticket. Alice sends a ticket request to the DC using her TGT (together with an authenticator) (KRB_TGS_REQ).
5. The TGS of the DC checks the TGT and the authenticator, generates a ticket for the local machine, and sends it back to Alice (KRB_TGS_REP).
6. On Alice's machine, the ticket is presented to the Local Security Authority, which will create an access token for Alice. From then on, any process acting on behalf of Alice can access the local machine's resources.

Network logon process

When Alice is already logged onto a domain and wants to access a resource located on a server within the same domain, a network logon process will take place. In this case, the logon sequence is as follows (see Figure 5.12):

1. Alice sends a server ticket request to the DC using her TGT (together with an authenticator) (KRB_TGS_REQ).
2. The TGS of the DC checks the authenticator, generates a server ticket, and sends it back to Alice (KRB_TGS_REP).
3. Alice sends the ticket (together with an authenticator) to the application server (KRB_AP_REQ).
4. The application verifies the ticket with the authenticator and sends back his or her authenticator to Alice for server authentication (KRB_AP_REP).

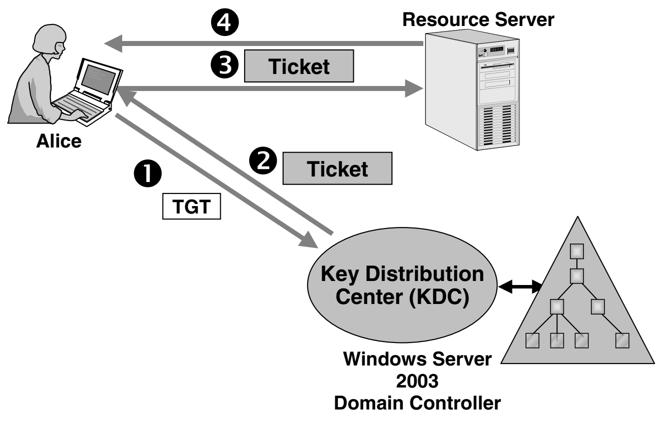

Resource Server
Alice
❶ TGT
❷ Ticket
❸ Ticket
❹
Key Distribution Center (KDC)
Windows Server 2003 Domain Controller

Figure 5.12 *Network logon process in a single domain environment.*

Disabling Kerberos in migration scenarios

In certain migration scenarios it may be necessary to disable the Kerberos authentication protocol on your Windows Server 2003 domain controllers. Remember from Chapter 4 that for Windows 2000 Professional and later clients, the first authentication protocol of choice is always Kerberos—at least if the client is talking to a Windows 2000 or later DC.

Imagine the following migration scenario: You have migrated all your client platforms to Windows XP and you upgraded one of your Windows NT4.0 DCs to Windows Server 2003—all the remaining DCs are still on NT4.0. In this scenario the one and only Windows Server 2003 DC may become overloaded by Kerberos authentication traffic because all Windows XP clients try to authenticate against it. Also, if this DC becomes unavailable, the clients cannot be authenticated anymore. Once a client has been authenticated by a Kerberos KDC it will not fall back to NTLM and NT4 DCs. This is a typical scenario where you may want to temporarily disable the Kerberos authentication protocol on the Windows Server 2003 DC.

To disable Kerberos, Microsoft provides a registry hack (the hack is available from Windows 2000 Service Pack 2 onward). The hack is called NT4Emulator (REG_DWORD) and should be added to the HKEY_LOCAL_MACHINE/System/CurrentControlSet/Services/Netlogon/Parameters registry key of the Windows 2000 SP2 or later DC and set to value 1.

The creation of this key on the Windows 2000 SP2 or later DC creates another problem because it will make it impossible to manage the AD using any of the MMC-based AD management tools. To get around this problem, you must make a registry change on the clients from which you want to use the AD management tools. Add the NeutralizeNT4Emulator registry value (REG_DWORD) in the HKEY_LOCAL_MACHINE/System/CurrentControlSet/Services/Netlogon/Parameters registry key and set it to value 1.

The role of SPNs

One of the great features of Kerberos is its support for mutual authentication. The enabling technologies for mutual authentication are the Kerberos protocol itself, User Principal Names (UPNs), and Service Principal Names (SPNs). The UPN and SPN concepts were introduced in Chapter 2. In the following we will look at how SPNs are used during the Kerberos authentication exchanges.

Let's take the example of a network logon process: A user decides to access a file on another server during his or her logon session. In this example the following SPN-related events will occur during the authentication exchanges:

- The Kerberos software on the client side constructs a Kerberos "KRB_TGS_REQ" message, containing the user's TGT and the SPN of the service that is responsible for the file the user wants to access. This message is sent to the user's domain controller. A Kerberos client can always construct a service's SPN—how this works was explained in Chapter 2.
- The KDC queries the AD[11] to find an account that has a matching SPN (this process is also known as "resolving" the SPN). The service's SPN must be unique in the AD. If more than one account is found with a corresponding SPN, the authentication will fail.
- Given the service account corresponding to the SPN and the associated master key, the KDC can construct a service ticket and send it back to the client. (This is the "KRB_TGS_REP" message.)
- In the next step the client sends a "KRB_AP_REQ" message to the file server (including the service ticket and a Kerberos authenticator).
- Finally, the service will authenticate back to the client (this is the "KRB_AP_REP" message). This is where the real mutual authentication happens.

11. In the first place it will query the local domain Naming Context of the user's authentication DC; after that, it will query the global catalog.

5.3.2 Logging on in a multiple domain environment

Typical examples of scenarios where a multiple domain logon process occurs are the following:

- Alice is logging on from a machine member of a different domain than the one where Alice's account has been defined (this is a local logon process).
- Alice is accessing a resource located on a machine that is a member of a different domain than the one where Alice initially logged on (this is a network logon process).

In the following examples, we will frequently use the concepts "referral ticket" and "inter-realm key." These concepts will be explained in more detail in the section on "multiple domain logon: under the hood."

Local logon process

Figure 5.13 shows a typical multidomain environment, consisting of a parent domain hp.com and two child domains, North America (NA) and Europe. In the local logon example, Alice's account is defined in the Europe domain. Alice logs on from a workstation whose account is defined in the NA domain.

The local logon process can be broken down into the four following steps:

- Step 1: AS exchange (KRB_AS_REQ and KRB_AS_REP):
  - To log on Alice, a TGT request is sent to a KDC in Europe.hp.com.
  - The AS Request is sent to a KDC in Europe.hp.com. The selected KDC will sent back the AS Reply. Only a KDC of Alice's account domain can authenticate Alice (Windows credentials are never replicated between the domain controllers of different domains).
- Steps 2, 3, and 4: TGS exchanges (KRB_TGS_REQ and KRB_TGS_REP):
  - To request a ticket for Alice to work on the NA workstation, a TGS request is sent to the KDC of Europe.hp.com.
  - The KDC of Europe.hp.com cannot issue a ticket that allows Alice to work on a workstation in NA. Only a KDC of NA can return such a ticket.[12] Therefore, the TGS reply contains a referral

12. This is because only a KDC of NA knows the workstation's master key.

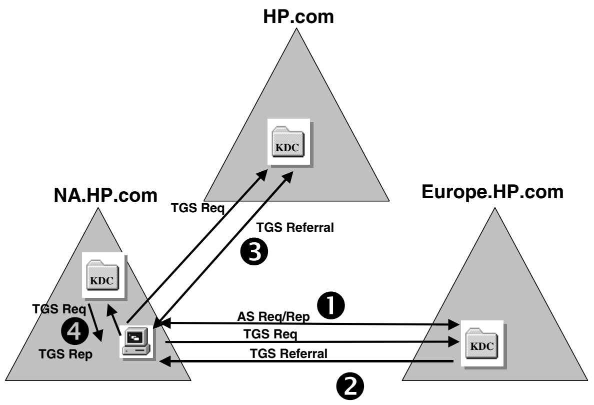


Figure 5.13 *Local logon in a multiple domain environment.*

ticket to the domain closest to NA.hp.com (from a DNS point of view) and with which NA.hp.com has a real (nontransitive) Kerberos trust. In this example, this is hp.com.

- On receiving the referral ticket, Alice locates a KDC of the intermediary domain hp.com and sends a TGS request including the referral ticket to that KDC.
- The KDC in hp.com decrypts the referral ticket using an interdomain key shared between Europe.hp.com and hp.com. The KDC detects that the referral ticket contains a ticket request for a ticket for a workstation in NA. The KDC checks on the domain closest to NA.hp.com from hp.com's point of view and sends Alice a referral ticket to this domain.
- Alice asks a KDC of NA.hp.com for a ticket for the local workstation. Finally, the KDC of NA.hp.com will send Alice a TGS reply with a valid ticket for the workstation.

The amount of interdomain authentication traffic occurring in this scenario should not be overestimated for several reasons:

- The size of Kerberos tickets is relatively small.
- Tickets have a lifetime and are cached.
- The referral traffic does not occur for every resource access.

An interesting side note is to look at what happens if at some point in this exchange, the administrator of the Europe domain decides to disable Alice's account. The answer to this question is pretty straightforward: The KDC of Europe will continue to issue tickets as long as the original TGT is valid. The disabled account will only be detected when Alice tries to get a new TGT.

Network logon process

Let's look at what happens with a local logon process in a multidomain environment. Again, we are using the example of a parent domain and two child domains. In the following network logon example, Alice is logged on to the NA domain (Alice and computer account are defined in the NA domain). Alice wants to access a resource hosted on a server in the Europe domain.

The network logon process can be split into the following four steps (as illustrated in Figure 5.14):

- Steps 1, 2, and 3: TGS exchanges:
  - Before Alice can contact the KDC in realm Europe.hp.com, she must have a valid ticket to talk to the KDC of that domain. Because there is no direct trust between Europe.hp.com and NA.hp.com, Alice must request the ticket via an intermediary domain.
  - Alice first tries to request a ticket for the KDC of the domain closest to the Europe.hp.com domain; this is hp.com. Because there is a direct trust between NA.hp.com and hp.com, Alice can request this ticket to her own KDC. The KDC will return a referral ticket that is encrypted with the interdomain key shared between NA.hp.com and hp.com.
  - Armed with this referral ticket, Alice can send a TGS request to the KDC of the hp.com domain, requesting a ticket for the KDC of the Europe.hp.com domain. Because there is a direct trust between hp.com and Europe.hp.com, the KDC of hp.com can answer this request. The returned referral ticket will be encrypted with the interdomain key shared between hp.com and Europe.hp.com.

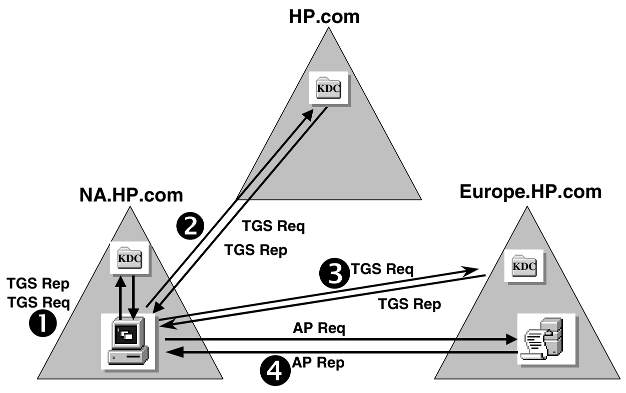


Figure 5.14
Network logon in a multiple domain environment.

- With this referral ticket, Alice finally can send a TGS request to a KDC of the Europe.hp.com domain to request a ticket for the target file server.

- Step 4: Application Server Exchange:
  - With the ticket she received from the target server's KDC, Alice can send an authentication request (consisting of the ticket and an authenticator) to the target server.
  - During the last step, the target server will also authenticate back to Alice.

The effect of shortcut trusts on multiple domain logon traffic

A typical scenario where you would create a shortcut trust is a Windows Server 2003 domain tree where a massive amount of authentication traffic occurs between two domains that are logically linked together using a transitive trust (such as the example shown in Figure 5.15). The shortcut trust example illustrated in Figure 5.15 shows how the number of referrals is reduced and how the trust path used during authentication is shortened. Note that the KDC in the user domain can detect the existence of shortcut trust when querying the AD. It has enough intelligence to refer Alice directly to the KDC in the resource domain.

In the example illustrated in Figure 5.14, Alice would go through the following steps to access the resource located in the Europe domain when a shortcut trust is defined between the Europe and the NA domains:

- Step 1: Alice uses her TGT to obtain a ticket from the KDC in the NA domain for the resource server in the Europe domain. The KDC in the NA domain is not the authoritative KDC for the resource server's Europe domain, so the KDC in the NA domain refers Alice

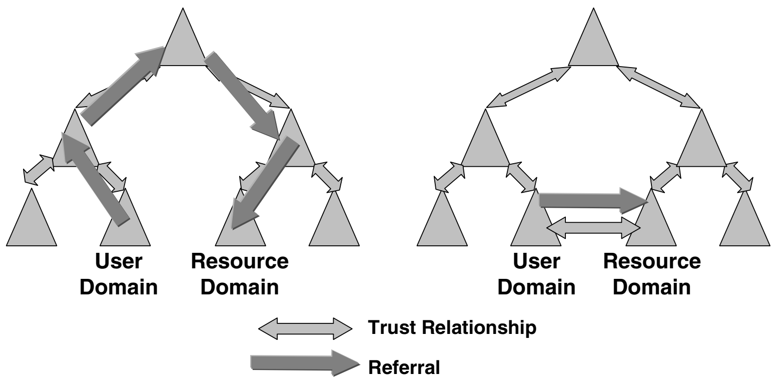


Figure 5.15 *Effect of a shortcut trust on multiple domain logon traffic.*

to the domain closest to the target domain with which the NA domain has a Kerberos trust relationship. This domain is Europe.

- Step 2: The KDC in the Europe domain is the authority for the resource server's Europe domain, so The KDC in the Europe domain can generate a ticket for Alice.
- Step 3: Alice uses the ticket from the KDC in the Europe domain to access the resource server.

Transitive Trusts in Domains Containing Windows 2000, Windows Server 2003, and NT 4.0 DCs Be careful when relying on transitive trust in domains containing both Windows 2000 or Windows Server 2003 and NT 4 DCs. Because NT4 domain controllers do not support Kerberos, transitive trust will only work if a user is authenticated by a Windows 2000 or later DC.

Consider the network logon example illustrated in Figure 5.16. A user defined in NA logged on from a Windows 2000 workstation in NA is accessing a resource in the "Belgium" domain (be.emea.compaq.com). There is a transitive trust between the NA and Europe and between the NA and Be domains. In this scenario, authentication will fail if the DC authenticating the user in the Be domain is an NT4 DC. Because of the NT4 DC, authentication will fall back to NTLM. NTLM does not understand transitive trust and requires a "real" trust.

What does this mean? When the NT4 domain controller receives the authentication request from the user in NA, it cannot create a trust path back to the Be domain because NT4 and NTLM can only deal with "single hop" trusts. NTLM would work in this scenario if an "explicit" trust relationship was defined between the NA and Be domains.

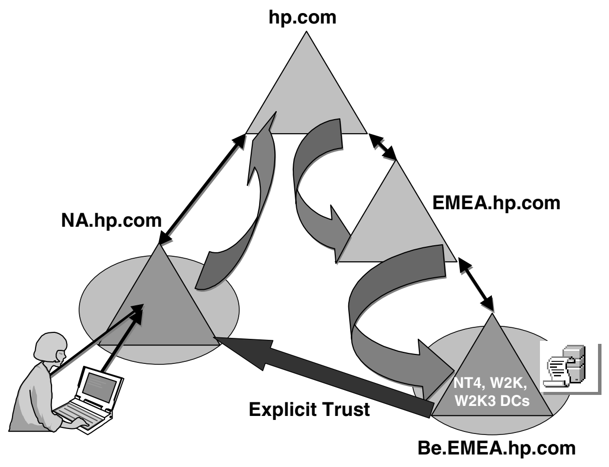


Figure 5.16 *Transitive trusts in mixed-mode domains.*

Multiple domain logon process: Under the hood

In this section we will explain some of the concepts behind the multiple domain logon process: referral tickets and the KDC's Authentication Service (AS) and Ticket Granting Service (TGS). To fully understand the multiple domain logon process, we will also introduce a special Kerberos principal—the krbtgt principal.

We will not come back to the concepts of trusteddomain account object (TDO) and interdomain secret. To enable interdomain authentication, every domain that is trusted by another domain is registered in the domain's AD domain naming context as a security principal. These principals are also known as TDOs. The TDO's master key is often referred to as an interdomain secret. TDOs were also explained in Chapter 3.

Interdomain authentication traffic (the referral tickets mentioned before) are secured using the master key and the session key of the TDO principals. Like for any other account, the domain trust account's master key is derived from the account's password. The creation of the TDOs and their master keys can happen automatically during the dcpromo process or manually when an administrator explicitly defines a trust relationship.

To explain the use of the krbtgt account, we must first explain why the Kerberos KDC is made up of two subservices: the Authentication Service (AS) and the Ticket Granting Service (TGS). The services offered by a Kerberos KDC can be split into two service categories; each subservice has a set of different tasks:

- The Authentication Service authenticates accounts defined in the domain where the AS is running and issues TGTs for these accounts.
- The Ticket Granting Service issues service tickets for resources defined in the domain where the TGS is running.

The AS and TGS share a secret that is derived from the password of the krbtgt principal, which is the security principal used by the KDC. Its master key will be used to encrypt the TGTs that are issued by the KDC. The krbtgt account is created automatically when a Windows 2000 or Windows Server 2003 domain is created. It cannot be deleted and renamed. Like for any other account, its password is changed regularly. In the Windows 2000 and Windows Server 2003 Users and Computers snap-in, this account is always shown as disabled.

Now that we have explained the TDO, interdomain secret, and krbtgt concepts, let's look once more at how the multiple domain logon process works. A basic rule in Kerberos is that to access a resource a user needs a

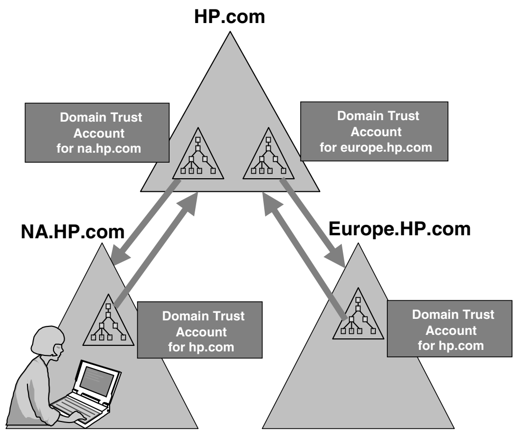

HP.com
Domain Trust Account for na.hp.com
Domain Trust Account for europe.hp.com
NA.HP.com
Europe.HP.com
Domain Trust Account for hp.com
Domain Trust Account for hp.com

Figure 5.17 *Multiple domain logon process revisited.*

ticket. How can Alice get a ticket for a resource contained in a domain different from Alice's definition domain? Let's once more take the example of Alice defined and logged on in domain na.hp.com, who decides to access a resource in europe.hp.com (as illustrated in Figure 5.17).

In this scenario, the KDC of na.hp.com would issue a referral ticket to Alice to access hp.com. What exactly is a referral ticket? A referral ticket is a TGT that Alice can use in domain hp.com to get a ticket for the resource in that domain. The KDC of na.hp.com can issue such a TGT because hp.com is a security principal [it has a TDO, Trusted Domain Object, account (see Chapter 3)] in his or her domain. How can the KDC of hp.com trust a TGT that was issued not by itself, but by the KDC of na.hp.com? The KDC of hp.com will decrypt the TGT with the interdomain secret of its TDO account in na.hp.com to retrieve the session key—it will then use this session key to validate the associated Kerberos authenticator. If the authenticator is valid, the KDC will consider the TGT trustworthy. The same things will happen when hp.com issues a referral ticket for Alice to authenticate to a domain controller in europe.hp.com.

The referral process we just explained relies heavily on the AD and more particularly on the global catalog (GC). First, it uses the GC to find out, given the SPN, in which domain the resource is located, and thus which domain controller can issue a ticket to access the resource. Then it uses the AD to find out the domain closest to the target domain to which Alice should be referred.

Figure 5.18 explains the process outlined in the previous sections a bit more in detail and using UNIX Kerberos terminology: It illustrates an inter-realm (interdomain) Kerberos authentication exchange between the

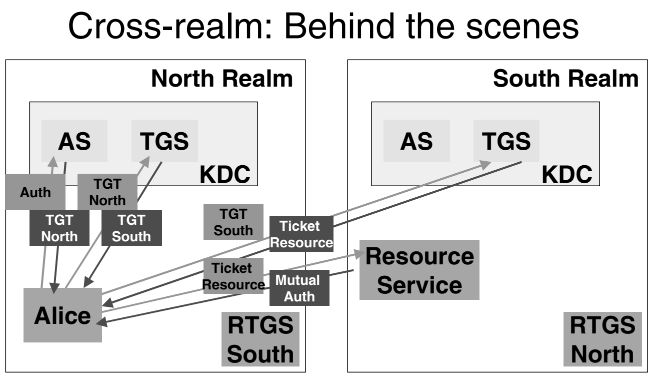


Figure 5.18 *Multiple domain logon process: under the hood.*

North and the South realms. In the example of Figure 5.18, Alice wants to access a resource service in the North realm. In the UNIX Kerberos terminology, they call the Windows TDO accounts from the previous example Remote Ticket Granting Service (RTGS) accounts: Figure 5.18 shows that there is an RTGS account for North in the South realm and vice versa.

5.3.3 Multiple forest logon process

In Windows Server 2003, Microsoft has added additional information in the TDO account objects to enable interforest authentication traffic. Let's look at an example that shows how Windows Server 2003 uses the extra information stored in the TDO to route Kerberos authentication requests during a cross-forest resource access.

In the example (illustrated in Figure 5.19), a user that is logged on to the emea.compaq.com domain (the user and machine accounts are defined in emea.compaq.com) wants to access a resource located on a server in the us.hp.com domain. Both forests are at functionality level 2, and a bidirectional forest trust relationship has been set up between them. From a Kerberos point of view, the user is already logged on to the emea.compaq.com domain and has a valid TGT. The remote resource is identified using an SPN of the following format: <servicename>/us.hp.com.

In this example the authentication requests will be routed as follows:

1. The user's machine contacts the local DC to request a Kerberos service ticket for the resource in the us.hp.com domain. The DC in emea.compaq.com cannot find an entry for the remote service in its local domain database and asks a GC server in the emea.compaq.com for help. The GC suspects (based on the DNS suffix) that the service is located in the hp.com forest, and it sends

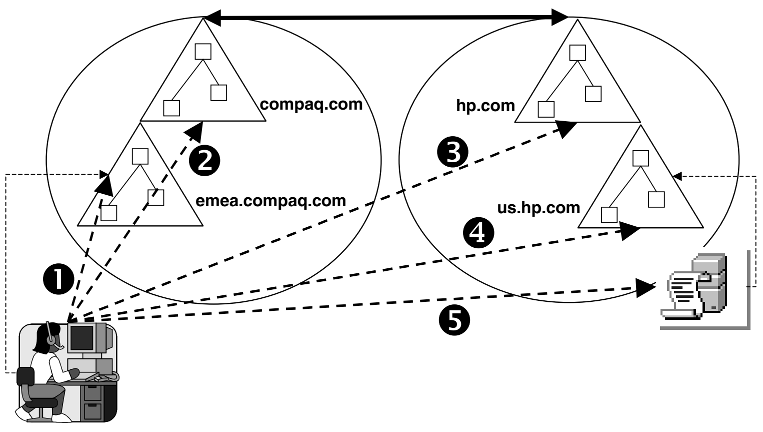


Figure 5.19 *Forest trust authentication flow.*

this routing hint to the DC and tells the DC to refer the user to a DC in the compaq.com root domain.

2. The user's machine contacts a DC in the root domain of the compaq.com forest. This DC refers the user to a DC in the root domain of the hp.com forest.
3. The user's machine contacts a DC in the root domain of the hp.com forest. The DC of the hp.com forest double-checks with the local GC whether the service is in his or her forest. After validation it refers the user to a DC in the us.hp.com domain.
4. The user's machine contacts a DC in the us.hp.com domain. This DC can issue a service ticket to the user for the resource in the us.hp.com domain.
5. The user uses the service ticket to authenticate to the resource server in the us.hp.com domain.

5.4 Advanced Kerberos topics

In this section we will focus on some advanced Kerberos topics: delegation of authentication, the link between authentication and authorization, the content of Kerberos tickets and authenticators, the details behind the smart card logon process, Kerberos transport protocol, and port usage.

5.4.1 Delegation of authentication

Delegation refers to the facility for a service to impersonate an authenticated client in order to relieve the user of the additional burden of authenticating to multiple services. To the latter services it will look as if they are communicating directly with the user, whereas in reality another service will sit between them and the user.

A classical example of where delegation is a very useful feature is when a user asks a print server to print a file that is located on another server. In today's Internet world there are many more examples. Basically, any Web-based multitier application can take advantage of delegation. Examples are Web sites launching user queries against a database located on some back-end server, or a user accessing his or her mailbox from a Web interface [a good example is Microsoft's Outlook Web Access (OWA)]. In the future, when Web services become widespread, the need for authentication delegation support will only become bigger. Web services are rooted on highly distributed architectures that can make data and other resources available to a wide range of users. Web services are typically accessed using open Internet protocols (such as HTTP and SMTP). In such environments it would not be a very smart idea to host the data on the Web servers. Web services require multitier application designs and the ability to reuse the user identity end-to-end.

The ability to refer to a user's identity end-to-end in a multitier application scenario is one of the key advantages of the Kerberos delegation support. It means that administrators can enforce authorization settings at the different tiers using a single user identity. This not only simplifies management but also facilitates user tracking and auditing on the different levels of a multitier application. Finally, because of its ability to transparently authenticate a user on multiple tiers, delegation provides SSO support.

Kerberos' ability to support delegation is a consequence of its unique ticketing mechanism. When sending a ticket to a server, the Kerberos client can add additional information to it so the server can reuse it to request other tickets on the user's behalf to the Kerberos KDC.

Delegation: Behind the scenes

The Kerberos delegation uses specific flags that can be set in a Kerberos ticket. The Kerberos standard (RFC 1510) defines four types of flags, shown in Table 5.2. Windows 2000 and Windows Server 2003 currently only support the "forwardable" and "forwarded" flags. Notice in Table 5.2 that "forwardable" is a much more powerful concept than "proxiable": A forwardable ticket is a TGT, a proxiable ticket is a plain ticket; a ticket can be used for one single application, and a TGT can be used for multiple applications.

Table 5.2 *Kerberos Ticket Delegation Flags*

Flag	Meaning
Proxiable	Tells the TGS that a new service ticket with a different network address may be issued based on this ticket
Proxy	Indicates that the ticket is a proxy ticket
Forwardable	Tells the TGS that a new TGT with a different network address may be issued based on this TGT
Forwarded	Indicates that this ticket has been forwarded or was issued based on an authentication using a forwarded TGT

What's missing in Windows 2000 Kerberos delegation?

One of the reasons why Kerberos delegation in Windows 2000 is only used rarely is because few people really know and understand it. Another reason is that the Windows 2000 implementation lacks some important security-related configuration options.

In Windows 2000, when a computer is "trusted for delegation," it can impersonate a user to any other service on any other computer in the Windows 2000 domain. In other words, when a Windows 2000 administrator trusts a computer for delegation, the delegation is "complete"; there are no configuration options to make the delegation more granular.

Another obstacle is that Kerberos delegation in a Windows 2000 Web scenario only works if the user has authenticated to the Web server using Kerberos or Basic authentication. There is no way to use delegation when you prefer to use the more secure digest authentication protocol to authenticate your users to the Web server. We also have to keep in mind that the use of Kerberos between a browser and a Web server is only possible when the browser supports Kerberos and the Kerberos KDC is accessible from the browser. The latter is clearly a problem in Internet scenarios: Very few companies are willing to expose their KDC to Internet users. Also, on the Internet, not every user has a Kerberos-enabled Web browser. So far, only Microsoft's Internet Explorer (from version 5.0 on) is Kerberos-enabled.

In Windows Server 2003, Microsoft embedded a set of Kerberos protocol extensions to remedy these problems. These extensions are referred to as the "Service-for-User" (S4U) Kerberos extensions. There are two new extensions: the Service-for-User-to-Proxy extension (S4U2Proxy) and the Service-for-User-to-Self extension (S4U2Self). Microsoft is planning to submit the specifications of both Kerberos extensions to the IETF some time in the near future.

The new Kerberos extensions are only available if your Windows Server 2003 domain is in Functionality Level 2 (which is the native Windows Server 2003 functionality level).

The S4U2Proxy Kerberos extension

The way that delegation works in Windows 2000 is by letting a Kerberos client forward a user's TGT to a service. In Kerberos-speak, a TGT is a very powerful security token: It is a digital piece of evidence that proves that a user's identity has been validated by the Kerberos KDC. A service can use a TGT to get many other service tickets on the user's behalf. This is why in Windows 2000 delegation is considered "complete." What makes the possessor of a TGT even more powerful is the fact that in Windows 2000 Microsoft uses the TGT to transport user-related authorization data [embedded in the Privilege Attribute Certificate (PAC) field].

The S4U2Proxy Kerberos Extension allows a service to reuse a user's "service ticket" to request a new service ticket to the KDC. In other words, there is no more need to forward a user's TGT to the service. The simple fact that the service can present a user's ticket to a KDC is enough to prove a user's identity. The Kerberos exchanges occurring in a typical S4U2Proxy scenario are illustrated in Figure 5.20.

In Figure 5.20 steps 1 through 3 illustrate the Kerberos exchanges related to a user authenticating to server 1.

- Step 1: The user requests a service ticket for server 1 to the KDC (Kerberos TGS_REQ message).
- Step 2: The KDC returns a service ticket for server 1 to the user (Kerberos TGS_REP message).
- Step 3: The user presents the service ticket for server 1 to server 1 (Kerberos AP_REQ message).
- Server 1 then needs to access a resource on server 2 on the user's behalf. Server 1 is running Windows Server 2003 and thus supports the S4U2Proxy extension.
- Step 4: The S4U2Proxy extension on server 1 requests a service ticket for server 2 to the KDC on the user's behalf. To prove the user's identity to the KDC, server 1 presents the user's service ticket for server 1 (Kerberos TGS_REQ message).

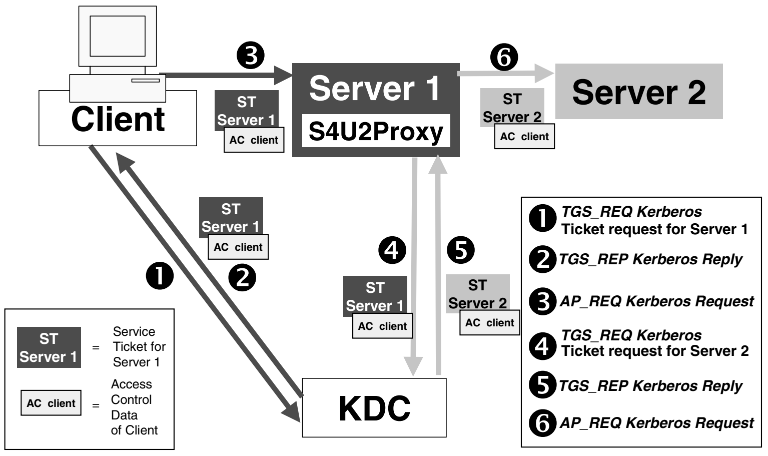

Client
Server 1
S4U2Proxy
Server 2
KDC
ST Server 1
AC client
ST Server 2
❶ ❷ ❸ ❹ ❺ ❻
ST Server 1 = Service Ticket for Server 1
AC client = Access Control Data of Client
❶ TGS_REQ Kerberos Ticket request for Server 1
❷ TGS_REP Kerberos Reply
❸ AP_REQ Kerberos Request
❹ TGS_REQ Kerberos Ticket request for Server 2
❺ TGS_REP Kerberos Reply
❻ AP_REQ Kerberos Request

Figure 5.20 *Basic S4U2Proxy operation.*

- Step 5: The KDC returns a service ticket for server 2 to server 1. Even though this new ticket is passed to server 1, it contains the user's authorization data (Kerberos TGS_REP message).
- Step 6: Server 1 presents the service ticket for server 2 to server 2 (Kerberos AP_REQ message).

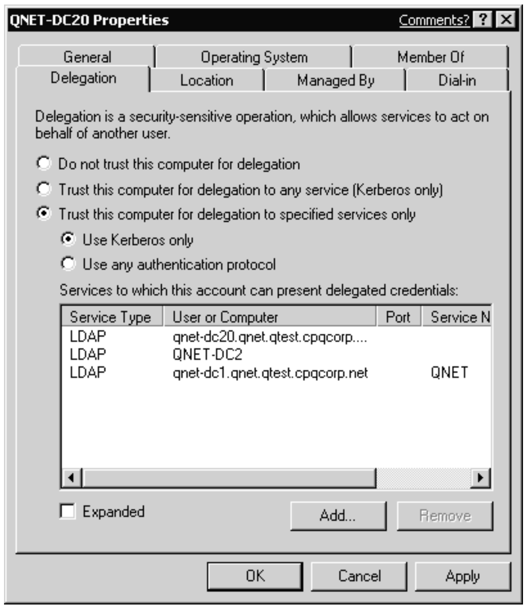
If a service could do this with any user ticket it receives, the S4U2Proxy feature would create a security hole: The service would be allowed to access any other service on the user's behalf. That is why Microsoft has added support for fine-grain delegation configuration. In Windows Server 2003 an administrator can configure which services a machine or service is allowed to access on a user's behalf. How to set this constrained delegation up is explained in the next section.

Configuring constrained delegation

When you open the properties of a machine or a service account in Windows Server 2003 (from the Users and Computers MMC snap-in), you will notice the new "Delegation" tab. This tab is not available in the properties of a plain user account. It only shows up if the account has an associated SPN.

From the Delegation tab you can configure delegation in three different ways (as illustrated in Figure 5.21):

- *Disallowed:* This is done by checking the "Do not trust this computer for delegation" option.
- *Allowed for all services:* This can be done by checking the "Trust this computer for delegation to any service (Kerberos only)" option. This



QNET-DC20 Properties

Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

- ○ Do not trust this computer for delegation
- ○ Trust this computer for delegation to any service (Kerberos only)
- ● Trust this computer for delegation to specified services only
  - ● Use Kerberos only
  - ○ Use any authentication protocol

Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service N
LDAP	qnet-dc20.qnet.qtest.cpqcorp....		
LDAP	QNET-DC2		
LDAP	qnet-dc1.qnet.qtest.cpqcorp.net		QNET

☐ Expanded

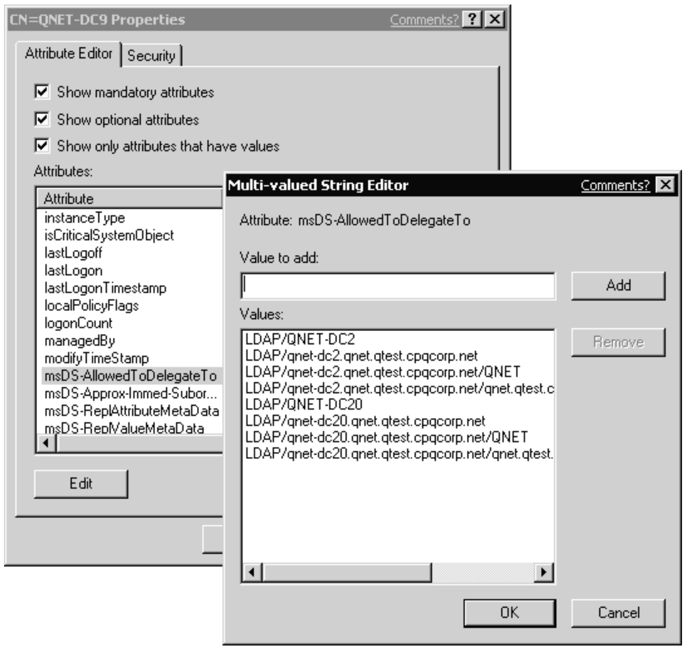
Figure 5.21 *Configuring delegation in Windows Server 2003.*

is the default option and refers to delegation the way it was available in Windows 2000.

- *Only allowed for a limited set of services:* This can be done by checking the "Trust this computer for delegation to specified services only" option. This is the new "constrained" delegation option coming with Windows Server 2003.

When selecting the latter option (constrained delegation), you can select the SPNs of the services for which the delegation is allowed. You can do so using the "Add…" push button. Pushing this button will bring up the "Add Services" dialog box. Initially, the available services list in this dialog box appears empty. To fill the list, press the "User or Computers…" push button. The latter will bring up the Active Directory object picker dialog box, which will allow you to select the appropriate SPNs. You can only select the SPNs of machines that are a member of the machine's domain.

The SPNs for which delegation is allowed are stored in a new AD account object attribute called "msDS-AllowedToDelegateTo." You can examine the content of this multivalued attribute using the Support Tools "AdsiEdit" tool (as illustrated in Figure 5.22). When the Kerberos authentication service receives a delegation request for a ticket for a particular service, it compares the SPNs listed in the Kerberos ticket request with the ones listed in the computer or service account's "msDS-AllowedToDele-


CN=QNET-DC9 Properties

Attribute Editor | Security

☑ Show mandatory attributes
☑ Show optional attributes
☑ Show only attributes that have values

Attributes:

```
Attribute
instanceType
isCriticalSystemObject
lastLogoff
lastLogon
lastLogonTimestamp
localPolicyFlags
logonCount
managedBy
modifyTimeStamp
msDS-AllowedToDelegateTo
msDS-Approx-Immed-Subor...
msDS-ReplAttributeMetaData
msDS-ReplValueMetaData
```

Multi-valued String Editor

Attribute: msDS-AllowedToDelegateTo

Value to add:

Values:

```
LDAP/QNET-DC2
LDAP/qnet-dc2.qnet.qtest.cpqcorp.net
LDAP/qnet-dc2.qnet.qtest.cpqcorp.net/QNET
LDAP/qnet-dc2.qnet.qtest.cpqcorp.net/qnet.qtest.c
LDAP/QNET-DC20
LDAP/qnet-dc20.qnet.qtest.cpqcorp.net
LDAP/qnet-dc20.qnet.qtest.cpqcorp.net/QNET
LDAP/qnet-dc20.qnet.qtest.cpqcorp.net/qnet.qtest.
```

Figure 5.22 *The new "msDS-AllowedToDelegateTo" AD account attribute enabling constrained delegation.*

gateTo" attribute. If there are no matches, the delegation request is denied. Remember that an account's SPNs are stored in the "servicePrincipalName" attribute.

In the scenario illustrated in Figure 5.20, you would set up constrained delegation in the account properties of server 1. You would trust server 1 for delegation to the SPN of server 2.

The S4U2Self Kerberos extension

An important problem when using Kerberos delegation in a Web-based Windows 2000 environment is that it can only be used when the client uses Kerberos or Basic authentication to authenticate to the Web server. The Web server (IIS 6.0) that ships with Windows Server 2003 comes with many other interesting authentication options, such as MS Passport–based, Digest-based, or certificate-based authentication. In Windows Server 2003 you can use the latter authentication options together with Kerberos delegation thanks to the combination of the S4U2Proxy (explained earlier) and another new Windows Server 2003 Kerberos extension called Service-for-User-to-Self (S4U2Self).

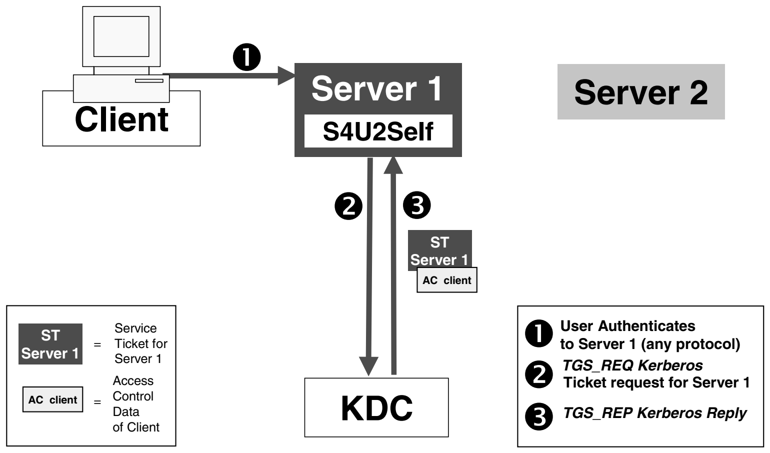
The service provided by the S4U2Self extension provides a service known as "protocol transition." It allows for the combination of any of the

IIS authentication protocols listed earlier on the IIS front end with the Kerberos authentication protocol on the IIS back end. In other words, the Web server can use Kerberos authentication and delegation with a user's identity to a back-end server independently of how the user authenticated to the Web server. Behind this new extension there is nothing else than the ability for an application or service to request to the Kerberos KDC a new service ticket on behalf of a user account defined in the domain or the forest. If this appears to you as a dangerous password-less logon process, keep in mind that before an application or service is allowed to request a service ticket on a user's behalf, it must already have authenticated the user. Also, the application or service itself must hold a valid Kerberos TGT. The basic operation of the S4U2Self Kerberos extension is illustrated in Figure 5.23.

In Figure 5.23, step 1 illustrates the user authenticating to server 1. When server 1 is an IIS6.0 Web server, he or she can do so using Digest-based, Basic-based, MS Passport–based, or client certificate–based authentication. In steps 2 through 3 server 1 requests a Kerberos ticket for server 1 on the user's behalf.

- Step 2: The S4U2Proxy extension on server 1 requests a service ticket for server 1 to the KDC on the user's behalf (Kerberos TGS_REQ message).
- Step 3: The KDC returns a service ticket for server 1 to server 1. Even though this new ticket is passed to server 1, it contains the user's authorization data (Kerberos TGS_REP message).

This clearly illustrates the authentication "transition": A user who was initially authenticated to server 1 using another authentication protocol is



❶
Client
Server 1
S4U2Self
Server 2
❷
❸
ST Server 1
AC client
ST Server 1 = Service Ticket for Server 1
AC client = Access Control Data of Client
KDC
❶ User Authenticates to Server 1 (any protocol)
❷ TGS_REQ Kerberos Ticket request for Server 1
❸ TGS_REP Kerberos Reply

Figure 5.23 *Basic S4U2Self operation.*

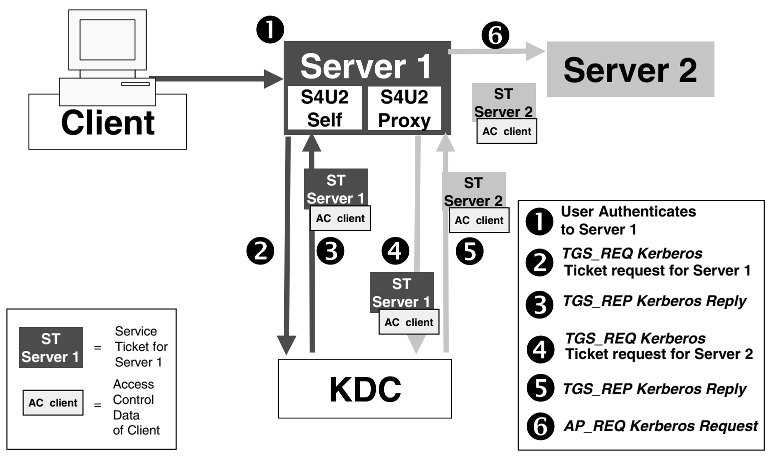

Client
Server 1
S4U2 Self
S4U2 Proxy
Server 2
KDC
ST Server 1
ST Server 2
AC client
❶ User Authenticates to Server 1
❷ TGS_REQ Kerberos Ticket request for Server 1
❸ TGS_REP Kerberos Reply
❹ TGS_REQ Kerberos Ticket request for Server 2
❺ TGS_REP Kerberos Reply
❻ AP_REQ Kerberos Request
ST Server 1 = Service Ticket for Server 1
AC client = Access Control Data of Client

Figure 5.24 *Combined S4U2Self operation and S4U2Proxy operation.*

transparently authenticated to server 1 using the Kerberos protocol. Once the user has authenticated using Kerberos to server 1 and the KDC has issued a service ticket for server 1, server 1 (in fact, the S4U2Proxy extension) can reuse this ticket to request a ticket for server 2 on the user's behalf. This combined S4U2Self and S4U2Proxy operation is shown in Figure 5.24.

Configuring protocol transition

Configuring protocol transition is relatively easy. So far I did not explain the "Use Kerberos only" and "Use any authentication protocol" configuration options (as illustrated in Figure 5.21). These are exactly the options that enable or disable the S4U2Self Kerberos extension. If you select "Use any authentication protocol," protocol transition will be possible; if you select the other one, it will not.

In order for protocol transition to function correctly, the following two conditions should also be met. They are both related to the service account of the service or application that is requesting a new ticket on the user's behalf.

1. In order for the service to obtain an impersonation token, its service account should have the "act as part of the operating system" privilege. If this is not the case, the service will only get an identification token.
2. In order to get the user's authorization data into the newly constructed user ticket, the service account also needs permission to enumerate the user's group memberships. This can be done by

adding the service account to the ACL of the "TokenGroupsGlobalAndUniversal" user object attribute or by adding the service account to the "Pre-Windows 2000 Compatible Access" group.

In the scenario illustrated in Figures 5.23 and 5.24, you would set up protocol transition in the account properties of server 1. If in this scenario the service impersonating the user is the IIS Web service, you would give the extra permissions and privileges mentioned earlier to the IIS Web service account.

A sample scenario

You can test the new Kerberos services in a small lab environment. Figure 5.25 shows the test scenario that I used. This scenario consists of a simple Web application that queries an SQL Server database on the back end. The database query is defined in a COM+ application that is running on the Web server. The query is called from an ASP page. The Web server and SQL server are members of the same domain. The client machine need not necessarily be a domain member.

The goal of this test scenario from an authentication point of view is to let the user use any authentication protocol (with the exception of Kerberos) to authenticate to the Web server. On the back end, to authenticate to the COM+ application and the SQL Server database, we would like to use Kerberos and Kerberos delegation. This can only be set up if the new Kerberos services (S4U2Proxy and S4U2Self) are available on the Web server. Table 5.3 summarizes the software requirements and configuration options you need to keep in mind when setting this up.

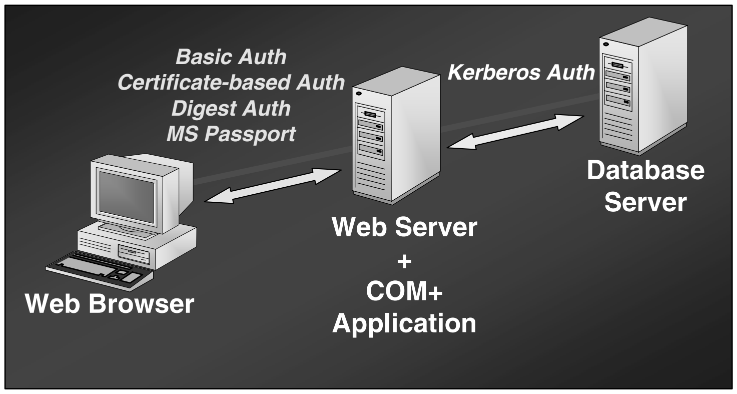


Figure 5.25
Sample scenario.

Table 5.3 *Configuration of Different Components*

Component	Software Requirements and/or Configuration Settings
Web Browser	Support for Basic authentication, Digest authentication, certificate-based authentication (SSL), or MS Passport–based authentication. The user account is a member of the domain.
Web Server	Delegation settings for computer account: ■ Trust this computer for delegation to specified services only. ■ Use any authentication protocol. ■ Add the SQL service of the database server's Service Principal Name (SPN) (A). ■ Web application is set to support Basic authentication, Digest authentication, certificate-based authentication (SSL), or MS Passport–based authentication. ■ The user has the appropriate access permissions to the Web site. ■ The Web server is a member of the domain. ■ Web server is running Windows Server 2003.
COM+ Application	The impersonation level of COM+ application must be set to "delegate." The user has the appropriate access permission to the COM+ dll.
Database Server	The SQL Server is configured to support Windows Integrated authentication. The SQL Server service has a registered SPN that is the one referred to in the Web server's computer account delegation settings [see (A) above]. The user has the appropriate access permissions to the database. The database server is a member of the domain. The database server is running Windows 2000 or Windows Server 2003 and SQL Server 2000.

5.4.2 From authentication to authorization

In this section we will explain the link between Windows Server 2003 authentication and authorization in the context of a Kerberos authentication exchange. Figure 5.26 illustrates the link between these two core operating system security services.

Next we will explain how we get from the Kerberos ticket (the basic entity used for authentication) to the access token (the basic entity used for authorization). An important component in this process is the Kerberos Privilege Attribute Certificate (PAC). Microsoft extended the base Kerberos protocol to include authorization data (e.g., global group memberships). A Windows Server 2003 ticket and TGT both contain a PAC.

Let us start off with a normal Kerberos authentication sequence. We are once more dealing with three entities: a user (Alice), a resource server, and a

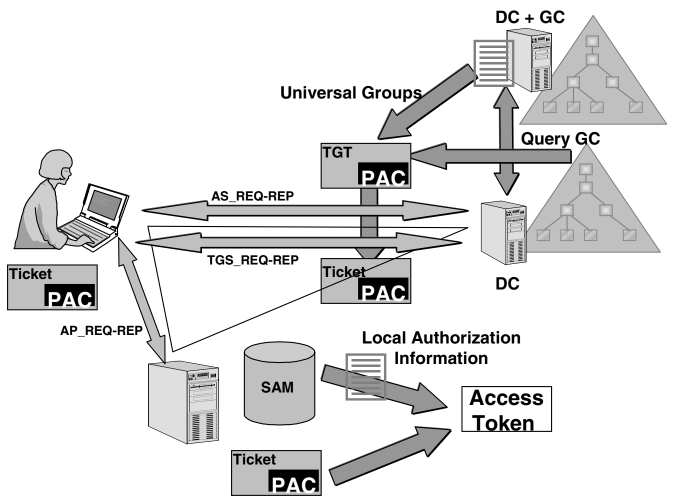


Figure 5.26 *From Windows Server 2003 authentication to authorization.*

Kerberos KDC. Once Alice's workstation has located a domain controller, it will request a TGT. The KDC will generate the PAC, embed the authorization data listed next, put the PAC into a TGT, and send the TGT to Alice.

- Alice's global group memberships and domain local group memberships: These are available from the KDC's local Active Directory (Domain NC).
- Alice's universal group memberships: These are available:
  - In the global catalog. If the KDC server himself or herself does not host a global catalog, the KDC service will need to query a global catalog on another domain controller.
  - In the domain naming context of Alice's logon domain. This is only true if the site of Alice's authenticating DC has universal group caching enabled (see following side note). The GC-less logon process or universal group caching is a new Windows Server 2003 feature that caches a user's universal group membership in the msDs-Cached-Membership attribute of a user account. Universal group memberships are cached in this attribute at the first user logon instance and are by default refreshed every 8 hours.
- The user rights assigned to Alice or any of her groups (universal, global, and domain local). These are available from the domain controller's LSA database.

Table 5.4 gives an overview of which group memberships are available where.

Table 5.4 *Windows Server 2003 Groups: Group Membership and Definition Storage Locations*

Group Type	Group: Available Where?	Group Membership: Available Where?
Universal group	AD: Global catalog	AD: Global catalog AD: Domain NC: only if Universal Group Caching (GC-less logon) is enabled.
Global group	AD: Global catalog	AD: Domain NC
Domain local group	AD: Domain NC	AD: Domain NC
Local group	Local machine: SAM	Local machine: SAM

Alice then decides she wants to access a resource hosted on a member server. Alice sends a request for a ticket to the KDC. This ticket will contain the same PAC as the one contained in the TGT. The ticket is sent back to Alice.

Alice authenticates to the resource server using the ticket. The LSA on the resource server will generate Alice's access token (for use in subsequent authorization decisions). In the access token the LSA will embed:

- Alice's authorization data found in the ticket's PAC (her universal, global, and domain local group memberships and user rights, assigned to her or any of the groups of which she is a member)
- Alice's authorization data found in the local security database (SAM): These are the local group memberships of Alice, the local group memberships of the groups (universal, global, or domain local) of which Alice is a member, and the user rights of Alice and Alice's groups.

To look at the contents of your access token, use the whoami tool (with the /all switch) that comes with the Windows Server 2003 code.

Key things to remember from this section are the following:

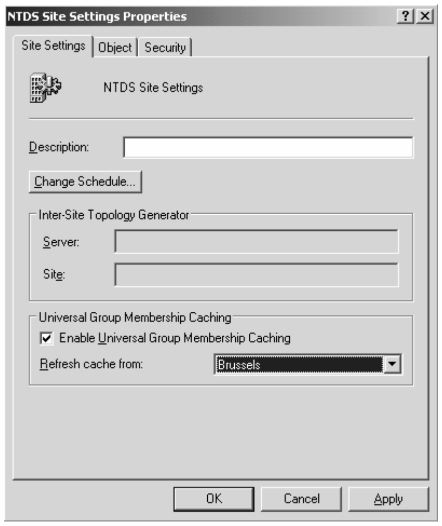
- The PAC data are added to a ticket on the KDC level and are inherited between subsequent TGT and ticket requests and renewals. The PAC data are not refreshed at ticket-request time. This means that if a user's group memberships change during its logon session, he or she will have to log off-log on (just as in NT4), wait for an automatic TGT renewal to occur, or purge the Kerberos ticket cache (using the klist or kerbtray utilities explained next). Note that even though

Enabling a GC-Less Logon Process A GC-less logon or universal group caching is a new Windows Server 2003 feature that enables Windows Server 2003 domain controllers to cache a user's universal group memberships in the msDS-Cached-Membership attribute of a user account. It is enabled in the NTDS settings properties of a site object—as illustrated in Figure 5.27.

Windows 2000 requires the availability of a GC server to retrieve a user's universal group membership when logging on to a domain. In Windows 2000 Microsoft provides a workaround for this requirement by enabling DCs to ignore GC failures, when a GC could not be contacted to find out about a user's universal group memberships. This workaround was based on a registry key called IgnoreGCFailures, which had to be added to the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa registry folder of every DC. This hack also took away the possibility to use universal groups in a Window 2000 forest. (This is documented in Microsoft Knowledge Base article Q241789.)

The new Windows Server 2003 feature does not fully take away the need to put a GC in every site—or at least to have one reachable GC for every site. Although GCs are not needed anymore to find out about a user's universal group memberships, you still need them to resolve UPN names when users are logging on using a UPN.*

* This UPN resolution process will only occur when using alternate UPN suffixes. These are suffixes that are different from the standard suffixes as they occur in the Windows DNS domain names.



☑ Enable Universal Group Membership Caching

Refresh cache from: Brussels

Figure 5.27
Enabling a GC-less logon process.

Windows does not refresh the authorization data it will check whether the account hasn't been disabled (see also the sidenote on "Kerberos and disabled accounts").

Kerberos and Disabled Accounts The following example illustrates how the fact that Kerberos TGTs are reusable as long as they are valid, together with the action of disabling a Windows account can lead to security holes in a Windows multi-domain environment. The AD environment illustrated in Figure 5.28 consists of a single forest with a single domain tree. A user's Windows account and machine account are defined in domain emea, a file server is part of the asiapac domain. When the user logs on to emea, he or she will receive a TGT for the emea domain. When the user accesses the file server in asiapac he or she will in addition get a TGT for the asiapac domain (see also Section 5.3.2). When the administrator disables the user account in emea, he or she won't be able to get any more tickets for resources in emea. The user will however still be able to get new tickets for resources in the asiapac domain—this will be possible as long as the user's TGT for asiapac remains valid. The reason for this is that the DCs in the asiapac domain don't check the user's account status when they issue tickets.

- Unless you have enabled the GC-less logon process (see the previous side note), the presence of at least one domain controller hosting a global catalog per domain tree is mandatory to log on a normal user. An exception to this rule is accounts that are member of the Administrators or Domain Administrators groups: They can log on even when a global catalog server is not available.

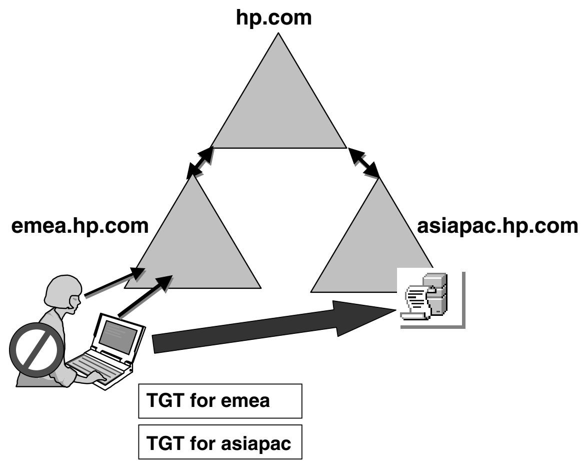


Figure 5.28 *Kerberos and disabled accounts: Example*

5.4.3 Analyzing the Kerberos ticket and authenticator

This section provides some inside information on the Kerberos ticket and authenticator. The concepts of a ticket and an authenticator and the relationship between the two are illustrated in Figure 5.29.

Remember that the primary purpose of a ticket is to securely transport the session key to be used for authentication between two entities. A ticket can only be decrypted by a KDC and the destination resource server. This way the client cannot decrypt and change its own authorization data (the information contained in the PAC). An authenticator is the Kerberos object that is providing the actual authentication. An authenticator can be checked by anyone possessing the corresponding session key. A detailed overview of the content of both the ticket and the authenticator is given in the following sections.

Ticket content

Table 5.5 shows the ticket fields, their meaning, and whether they are sent in encrypted format across the network.

Kerberos encryption types

Windows Server 2003 Kerberos supports the following cryptographic algorithms: RC4-HMAC, DES-CBC-CRC, and DES-CBC-MD5. The default encryption algorithm is "RC4-HMAC"—it was defined in an Internet draft called draft-brezak-win2k-krb-rc4-hmac-05.txt.

The default Kerberos encryption type can be changed using the "Use DES encryption types for this account" AD account property. The property can be set in the account options, which are available from the account tab in the account properties. Enabling this property is required when you're looking after UNIX and Windows Kerberos interoperability. DES encryption is the default in most UNIX Kerberos implementations.

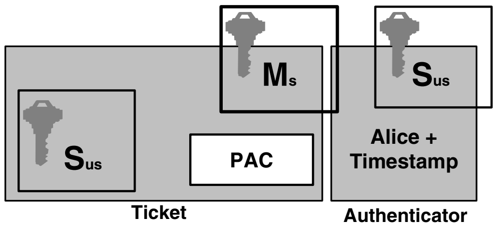


Figure 5.29 *Relationship between Kerberos ticket and authenticator.*

Table 5.5 *Kerberos Ticket Content*

Encrypted?	Name	Meaning
	Tkt-vno	Version number of the ticket format.
	Realm	Name of the realm (domain) that issued the ticket.
	Sname	Name of the server (Principal name).
✓	Flags	Ticket options. These are explained in more detail later in the chapter.
✓	Key	Session key.
✓	Crealm	Name of the client's realm (domain).
✓	Cname	Client's name (Principal name).
✓	Transited	Lists the Kerberos realms that took part in authenticating the client to whom the ticket was issued.
✓	Authtime	Time of initial authentication by the client. The KDC places a timestamp in this field when it issues a TGT. When it issues tickets based on a TGT, the KDC copies the authtime of the TGT to the authtime of the ticket.
✓	Starttime	(Optional) Time after which the ticket is valid.
✓	Endtime	Ticket's expiration time.
✓	Renew-till	(Optional) Time period during which the ticket is automatically renewed without the client having to provide his or her master key.
✓	Caddr	(Optional) One or more addresses from which the ticket can be used. If omitted, the ticket can be used from any address.
✓	Authorization data	(Optional) Privilege attributes for the client. Microsoft calls this part the Privilege Attribute Certificate (PAC).

There are two reasons why Microsoft did not use DES as the default algorithm:

- *Ease of upgrading from NT4 to Windows 2000 or Windows Server 2003.* The key used for RC4-HMAC can also be used with the Windows NT 4 Password Hash.
- *Export law restrictions.* In the early stages of the Windows 2000 development, 56-bit DES could not be exported outside of the United States. Because MS wanted to use the same Kerberos encryption technology in both the domestic and export versions of the product, they chose the 128-bit RC4-HMAC alternative. RC4-HMAC was already exportable at that point in time.

Table 5.6 *Kerberos Encryption Types: Key Lengths in Bits*

Algorithm	Authentication	Signing	Confidentiality Protection
RC4-HMAC	128	128	128 (56)
DES-CBC-CRC	56	56	56
DES-CBC-MD5	56	56	56

The algorithm used for a Kerberos ticket can be checked using the "klist" or "kerbtray" resource kit utilities. These utilities are explained later in this chapter.

Table 5.6 shows the algorithms and their supported key lengths. When the Windows 2000 "Strong encryption fix" has been installed, RC4-HMAC can use 128-bit keys for bulk encryption. This is the default in Windows Server 2003. A Windows domain can contain a mix of clients with and without the fix installed. Windows Kerberos will automatically choose the strongest available encryption algorithm.

The Privilege Attribute Certificate

The Privilege Attribute Certificate (PAC) enables the Kerberos protocol to transport authorization data—in the Windows case these data are user group memberships and user rights. We already explained part of the reason for existence of the PAC in the section on "From authentication to authorization."

Shortly after the release of Windows 2000, Microsoft received some negative press attention because of the proprietary way they used the PAC field in a Kerberos ticket. This forced Microsoft to release the PAC specifications. They can be downloaded from http://www.microsoft.com/downloads/details.aspx?displaylang=en&familyid=BF61D972-5086-49FB-A79C-53A5FD27A092. This document can be used only for informational purposes; it explicitly forbids the creation of software that implements the PAC as described in the specifications. A summary of the specifications can be found at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnkerb/html/MSDN_PAC.asp. Microsoft also submitted their PAC definition as an Internet Draft to the IETF—it's called draft-brezak-win2K-krb-authz-01.txt.

Most non-Microsoft Kerberos implementations ignore the PAC field and its content. Interoperability issues may arise though if a user is a member of a large amount of groups. In that case, the PAC size may become so big that it cannot be transported in a single UDP packet anymore. If this happens, a Windows KDC will request the client to switch to TCP, which cannot be done by some of the early Kerberos implementations (see also Section 5.5.3).

An important PAC security detail is that its content is digitally signed. By signing the authorization data, a hacker cannot make modifications to the data without being detected. This was possible in NTLM version 1: Authorization data for part of NTLM version 1 messages were not protected. Microsoft corrected this error in NTLM version 2, which is included in Windows 2000, XP, Windows Server 2003, and all the NT4 Service Packs from SP4 onward (see also Chapter 4 for more information).

The Kerberos PAC content is signed twice:

- Once with the master key of the KDC (this is the master key linked to the krbtgt account). This signature prevents malicious server-side services from changing authorization data. The LSA on the server side will require a validation of the signature for every ticket coming from a service that is not running using the local system account. To validate the signature, the server will need to set up a secure channel with the KDC that signed the authorization data. This extra validation step might remind you of NTLM and its pass-through authentication. This time, however, the pass-through is not used for validation of a response but for validation of a digital signature.
- Once with the master key of the destination service's service account (the destination service is the one responsible for the resource the user wants to access). This is the same key as the one used to encrypt the ticket content. This signature prevents a user from modifying the PAC content and adding its own authorization data.

The Kerberos ticket has a fixed size, which indirectly also limits the PAC size. If a user is a member of a large amount of groups, this size may be exceeded and, as a consequence, authentication and group policy processing may fail. Microsoft allows you to adjust the maximum size of a Kerberos ticket using the MaxTokenSize registry parameter. This parameter is a REG_DWORD value and is contained in the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\Kerberos registry container. The value should be adjusted on all Windows machines users are logging from to a domain to using Kerberos.

In Windows 2000, the default MaxTokenSize value is 8,000 bytes. In Windows 2000 Service Pack 2 (SP2) and Microsoft Windows Server 2003, the default value is 12,000 bytes.

The MaxTokenSize parameter is documented in Microsoft Knowledge Base article Q327825.

In order to reduce the PAC size, Microsoft implemented a new method to store authorization data in the PAC in Windows 2000 Service Pack 4 (SP4) and later, and in Windows Server 2003. This solution is also available as a hotfix for pre-Windows 2000 SP4 machines (see also Q327825). This new PAC authorization data storage method can be summarized as follows:

- If the global and universal groups a user belongs to are local to the domain the user is in, then only the RID (relative identifier) is stored.
- If the groups are local groups or are from other domains, the entire SID is stored.

This means for example that instead of storing an "S-1-5-21-1275210071-789336058-1957994488-3140" value (the SID), you would only store the "3140" value (the RID) in the PAC. Microsoft provides a special process on the client and server side to explode the RIDs back to the SID format during the Windows authorization process.

Even on platforms where this new PAC authorization data storage method is available, it may be required that the maxtokensize registry value be adjusted.

Kerberos preauthentication data

"Preauthentication" is a feature introduced in Kerberos version 5. With preauthentication data, a client can prove the knowledge of its password to the KDC before the TGT is issued. In Kerberos version 4, anyone, including a hacker, can send an authentication request to the KDC; the KDC does not care. It does not even care about authenticating the client: Authentication is completely based on the client's ability to decrypt the packet returned from the KDC using its master key.

Preauthentication also lowers the probability of an offline password-guessing attack. Without preauthentication data, it is easy for a hacker to do an offline password-guessing attack[13] on the encrypted packets returned from the KDC. A hacker can send out dummy requests for authentication; each time he or she will get back another encrypted packet, which means he or she gets another chance to make a brute-force attack on the encrypted packet and to guess the user's master key.

13. During an offline password-guessing attack, a hacker intercepts an encrypted packet, takes it offline, and tries to break it using different passwords This kind of offline attack is also known as a brute-force attack: The hacker tries out different keys (in this case "passwords") to decrypt a packet until he or she finds the right key that decrypts the packet in cleartext.

Table 5.7 *Kerberos Authenticator Content*

Encrypted?	Name	Meaning
✓	Authenticator-vno	Version number of the authenticator format. In Kerberos v.5 it is 5.
✓	Crealm	Name of the realm (domain) that issued the corresponding ticket.
✓	Cname	Name of the server that issued the corresponding ticket (Principal name).
✓	Cksum	(Optional) Checksum of the application data in the KRB_AP_REQ.
✓	Cusec	Microsecond part of the client's timestamp.
✓	Ctime	Current time on client.
✓	Subkey	(Optional) Client's choice for an encryption key to be used to protect an application session. If left out, the session key from the ticket is used.
✓	Seq-number	(Optional) Initial sequence number to be used by the KRB_PRIV or KRB_SAFE messages (protection against replay attacks).

In a standard Kerberos authentication sequence, the preauthentication data consist of an encrypted timestamp. When logging on using a smart card, the preauthentication data consist of a signed timestamp and the user's public key certificate. In Windows 2000 and Windows Server 2003, preauthentication is the default. An administrator can turn it off using the "Do not require Kerberos preauthentication" checkbox in the account properties ("account" tab). This might be required for compatibility with other implementations of the Kerberos protocol. Preauthentication affects the content of a ticket: Every ticket contains a special flag that is reserved for preauthentication.

Authenticator content

Table 5.7 shows the authenticator fields, their meaning, and whether they are sent in encrypted format across the network.

TGT and ticket flags

In this section, we will analyze the content of a Kerberos TGT and a service ticket; we will focus on the TGT and ticket "flags." The ticket flags and their meaning are explained in Table 5.8.

Next are some important notes on usage of the ticket flags in Windows 2000 and Windows Server 2003 Kerberos.

- By default, every ticket has the "forwardable" flag set. This default behavior can be reversed by setting the "account is sensitive and cannot be delegated" property on an account object. Windows 2000 and Windows Server 2003 Kerberos do not support "proxy" tickets.

- By default, every ticket has the "renewable" flag set. When a ticket expires and a new ticket is needed, the system will not automatically request a new ticket (a TGT or a service ticket) (automatic ticket requests will work as long as a user's cached credentials are available).
- By default, every ticket has the "preauthenticated" flag set.
- Every Windows 2000 TGT has the "initial" flag set.

Table 5.8 *Kerberos Ticket Flags*

Flags	**Meaning**
Forwardable	Indicates to the ticket-granting server that it can issue a new Ticket Granting Ticket with a different network address based on the presented ticket.
Forwarded	The ticket has either been forwarded or was issued based on authentication involving a forwarded Ticket Granting Ticket.
Proxiable	Indicates to the ticket-granting server that only nonticket-granting tickets may be issued with different network addresses.
Proxy	The ticket is a proxy ticket.
May be postdated	Indicates to the ticket-granting server that a postdated ticket may be issued based on this Ticket Granting Ticket.
Postdated	Indicates that a ticket has been postdated. The end service can check the ticket's auth-time field to see when the original authentication occurred.
Invalid	The ticket is invalid.
Renewable	The ticket is renewable. If this flag is set, the time limit for renewing the ticket is set in RenewTime. A renewable ticket can be used to obtain a replacement ticket that expires later.
Initial	The ticket was issued using the AS protocol instead of being based on a Ticket Granting Ticket.
Preauthenticated	Indicates that, during initial authentication, the client was authenticated by the KDC before a ticket was issued. The strength of the preauthentication method is not indicated, but is acceptable to the KDC.
Hardware preauthentication	Indicates that the protocol employed for initial authentication required the use of hardware expected to be possessed solely by the named client. The hardware authentication method is selected by the KDC and the strength of the method is not indicated.
Target trusted for delegation (OK as delegate)	This flag means that the target of the ticket is trusted by the directory service for delegation.

- A ticket has the "Target trusted for delegation" (OK as delegate) flag set if the service or user account for which the ticket was issued has the "account is trusted for delegation" property set, or, in the case of a computer account, if the computer object has "trust computer for delegation" set.

A single ticket can contain multiple flags. The flags are added to a ticket's properties as a hexadecimal 8-bit number, of which only the first 4 bits are significant. One bit can refer to different flags. If flags refer to the same bit position, they are added hexadecimally. This hexadecimal number is displayed when looking at the TGTs in the Kerberos ticket cache using the resource kit tool "klist"; the other resource kit tool "kerbtray" automatically converts the number to its appropriate meaning.

5.4.4 Kerberized applications

Kerberized applications are applications that use the Kerberos authentication protocol to provide authentication (and maybe in a later phase to provide encryption and signing for subsequent messages). Windows 2000 and Windows Server 2003 include the following Kerberized applications:

- LDAP to AD
- CIFS/SMB remote file access. Common Internet File System (CIFS) is the new name of Microsoft's SMB protocol that is mainly used for file and print sharing.
- Secure dynamic DNS update
- Distributed File System Management
- Host to Host IPsec using ISAKMP
- Secure intranet Web services using IIS
- Authenticate certificate request to certification authority (CA)
- DCOM RPC security provider

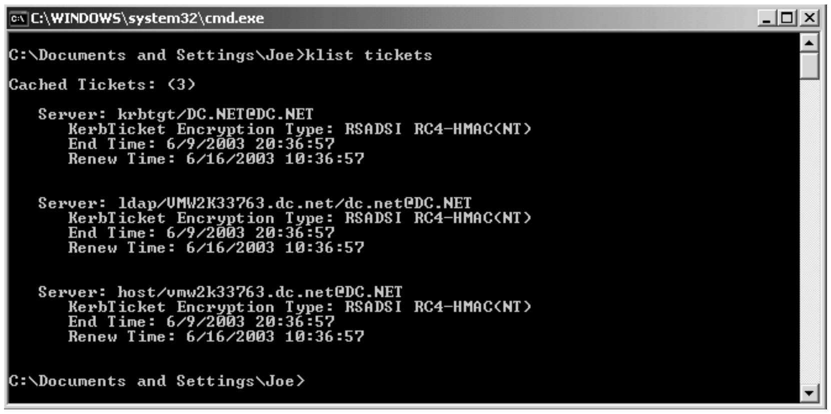
Smart card logon process

Windows 2000 and Windows Server 2003 include extensions to Kerberos Version 5 to support public-key–based authentication. These extensions are known as PKINIT—which stands for use of Public Key cryptography for INITial authentication—and are defined in an IETF Internet draft available from http://www.ietf.org. PKINIT enables the smart card logon process to a Windows 2000 or later domain. PKINIT allows a client's master key to be replaced with its public key credentials in the Kerberos Authentication

Using Klist and Kerbtray The Windows Server 2003 Resource Kit contains two utilities you can use to look at the content of the Kerberos ticket cache: kerbtray.exe (illustrated in Figure 5.30) and klist.exe (illustrated in Figure 5.31). Kerbtray.exe is a GUI tool, and klist.exe is a command-line tool. Both tools can be used to display and/or purge the content of the Kerberos ticket cache.

To bring up the kerbtray dialog box and look at your logon session's Kerberos ticket cache, double-click the kerbtray icon in the status area of your Windows desktop. The kerbtray icon is only displayed if you started the kerbtray program—it looks like a green ticket. The upper pane of the kerbtray dialog box shows all Kerberos tickets (both service tickets and TGTs) that are cached in your logon session's Kerberos ticket cache. The lower part of the dialog box has four tabs: Names, Times, Flags, and Encryption Types. The content of these tabs differs depending on the ticket that is selected in the upper pane.

- The Names tab shows the name of the security principal the Kerberos ticket was issued to [this is the user's User Principal Name (UPN)], together with the name of the service for which the ticket was issued [this is the service's Service Principal Name (SPN)].
- The Times tab shows the validity period of the ticket: its start and end time. For both TGTs and tickets, the default validity period is 10 hours.
- The Flags tab shows the Kerberos ticket flags that have been set in the ticket. Examples of ticket flags are the forwarded and proxy flags used during the Kerberos delegation process. For a more detailed explanation of all the Kerberos ticket flags, I refer to the Kerberos Version 5 (V5) standard document [Request For Comments (RFC) 1510], which can be downloaded from the IETF Web site at http://www.ietf.org.
- Finally, the Encryption Types tab shows the names of the symmetric encryption algorithms that were used by the Kerberos software to encrypt the tickets' content.



```
C:\Documents and Settings\Joe>klist tickets

Cached Tickets: (3)

   Server: krbtgt/DC.NET@DC.NET
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      End Time: 6/9/2003 20:36:57
      Renew Time: 6/16/2003 10:36:57


   Server: ldap/VMW2K33763.dc.net/dc.net@DC.NET
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      End Time: 6/9/2003 20:36:57
      Renew Time: 6/16/2003 10:36:57


   Server: host/vmw2k33763.dc.net@DC.NET
      KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
      End Time: 6/9/2003 20:36:57
      Renew Time: 6/16/2003 10:36:57


C:\Documents and Settings\Joe>
```

Figure 5.30 *Looking at the Kerberos ticket cache using the Klist utility.*

To purge the tickets in the Kerberos ticket cache, right-click the kerbtray icon in your desktop's status area and select Purge Tickets. This option deletes *all* tickets in your ticket cache. Use this option with extreme caution: Deleting tickets may stop you from authenticating to other Windows services during your logon session. If you have purged your tickets, you can only obtain new ones by logging off and then logging on again.

To display the content of the Kerberos ticket cache using the klist command-line utility, type the following at the command prompt:

```
Klist tickets
```

or

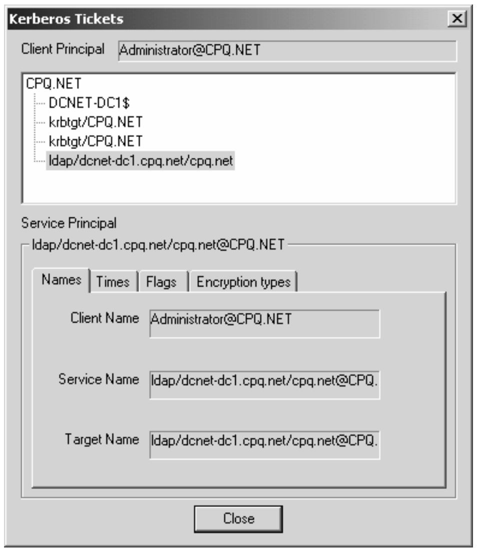
```
Klist tgt
```

The first command will bring up the service tickets in the cache, and the second command will bring up the TGTs in the cache. To purge the cache from the command line, type:

```
Klist purge
```

Again, use the latter command with extreme caution.

The kerbtray utility displays more ticket information than the klist utility does—it also displays the information in a much more readable format. For example, the klist utility displays the TGT tickets flags altogether in a single hexadecimal string: It is up to the user to decipher this string and retrieve the associated ticket flags.



Client Principal Administrator@CPQ.NET

```
CPQ.NET
  DCNET-DC1$
  krbtgt/CPQ.NET
  krbtgt/CPQ.NET
  ldap/dcnet-dc1.cpq.net/cpq.net
```

Service Principal
ldap/dcnet-dc1.cpq.net/cpq.net@CPQ.NET

Client Name Administrator@CPQ.NET
Service Name ldap/dcnet-dc1.cpq.net/cpq.net@CPQ.
Target Name ldap/dcnet-dc1.cpq.net/cpq.net@CPQ.

Figure 5.31 *Looking at the Kerberos ticket cache using the Kerbtray utility.*

Table 5.9 *Mapping the Standard Kerberos "Master Key" to the PKINIT "Public-Private Key"*

Standard Kerberos Usage of Master Key	PKINIT Replacement
Client-side encryption of the preauthentication data	Private key
KDC-side decryption of the preauthentication data	Public key
KDC-side encryption of session key	Public key
Client-side decryption of session key	Private key

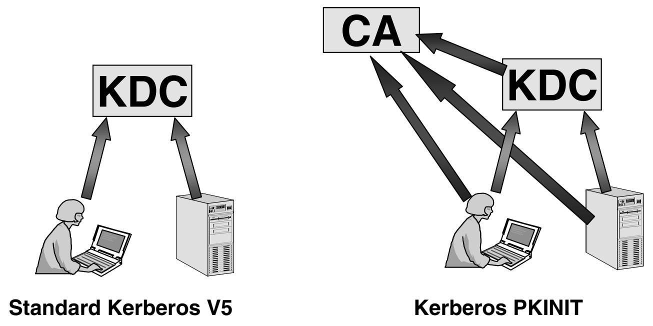
Request (KRB_AS_REQ) and Reply (KRB_AS_REP) messages. This is illustrated in Table 5.9.

PKINIT introduces a new trust model (illustrated on the right side of Figure 5.32) in which the KDC is not the first entity to identify the users (as is the case for classical Kerberos). Before KDC authentication, users are identified by the certification authority in order to obtain a certificate. In this new model the users and the KDC obviously both need to trust the same CA.

Figure 5.33 shows the way the Kerberos smart card logon process works (notice that the cryptic names of the Kerberos messages have changed):

- Alice starts the logon process by introducing her smart card and by authenticating to the card using her PIN code. The smart card contains Alice's public key credentials: her private key and certificate.
- A TGT request is sent to the KDC (AS); this request contains the following (PA-PK-AS-REQ):
  - Alice's principal name and a timestamp
  - The above signed with Alice's private key
  - A copy of Alice's certificate

Figure 5.32 *Smart card logon trust model.*



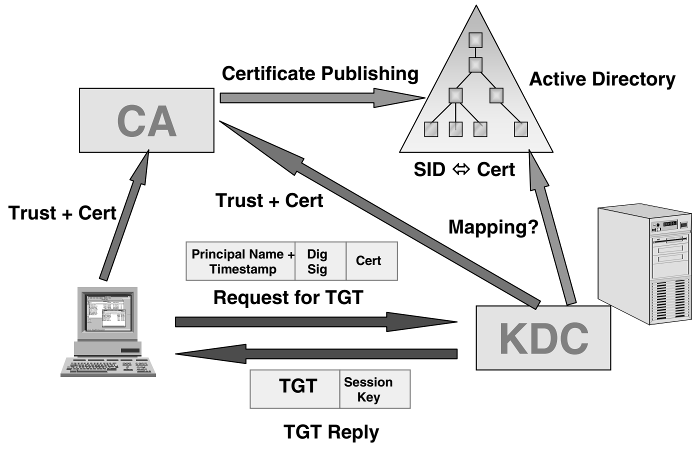


Figure 5.33 *Smart card logon process.*

- To validate the request and the digital signature on it, the KDC will first validate Alice's certificate. The KDC will then query the Active Directory for a mapping between the certificate and a Windows account. If it finds a mapping, it will issue a TGT to the corresponding account.
- The KDC sends back the TGT to Alice. Alice's copy of the session is encrypted with her public key (PA-PK-AS-REP).
- To retrieve her copy of the session key, Alice uses her private key.

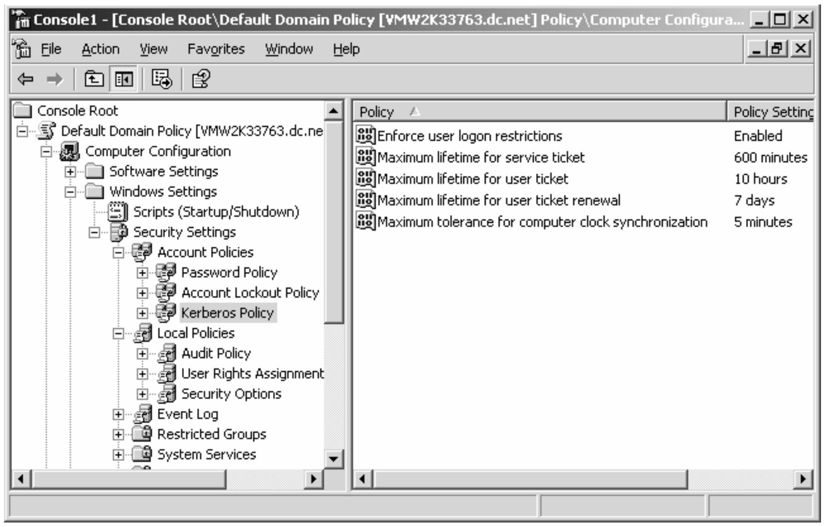
We will come back to smart card support in Windows Server 2003 in Chapter 17.

5.5 Kerberos configuration

5.5.1 Kerberos GPO settings

The Windows Server 2003 Account Policies [Part of the Group Policy Object (GPO) computer configuration] include a special subfolder for Kerberos-related policy settings (illustrated in Figure 5.34). It contains the following GPO entries:

- Enforce user logon restrictions: This setting enforces the KDC to check the validity of a user account every time a ticket request is submitted. If a user does not have the right to log on locally or if his or her account has been disabled, he or she will not get a ticket. By default, the setting is on.



Policy	Policy Setting
Enforce user logon restrictions	Enabled
Maximum lifetime for service ticket	600 minutes
Maximum lifetime for user ticket	10 hours
Maximum lifetime for user ticket renewal	7 days
Maximum tolerance for computer clock synchronization	5 minutes

Figure 5.34
Kerberos-related GPO settings.

- "Maximum lifetime for service ticket": In Microsoft terminology, a service ticket is a plain Kerberos ticket. Its default lifetime is 10 hours.
- "Maximum lifetime for user ticket": In Microsoft terminology, a user ticket is a Kerberos TGT. Its default lifetime is 10 hours.
- "Maximum lifetime for user ticket renewal": By default, the same ticket [service or user ticket (TGT)] can be renewed up until 7 days after its issuance. After 7 days, a brand-new ticket has to be issued.
- Maximum tolerance for computer clock synchronization: This is the maximum time skew that can be tolerated between a ticket's time-stamp and the current time at the KDC. Kerberos is using a time-stamp to protect against replay attacks. Setting this setting too high creates a bigger risk for replay attacks. The default setting is 5 minutes.

These Kerberos policies can only be set on a per-domain basis (the same is true for account lockout policies and password policies). You cannot define, for example, different Kerberos account policy settings per individual organizational unit (OU).

Another Kerberos-related GPO entry is located in Local policies\user rights assignment: "enable computer and user accounts to be trusted for delegation" sets the "trusted for delegation" property of user and computer objects in a domain, site, or organizational unit. Kerberos delegation was explained earlier in this chapter.

5.5.2 Kerberos-related account properties

Every Windows 2000 user account has a set of Kerberos-related properties. Most of them are related to Kerberos delegation, and one is related to the use of preauthentication.

Every user account has the property "Account is sensitive and cannot be delegated." Every machine account has a special delegation tab in its properties that can be used to define machine-related Kerberos delegation settings. This is a brand-new tab in the machine properties that was not available in Windows 2000. The details behind the machine-related delegation settings were explained in Section 5.4.1. One more point worthy of mentioning is that domain controllers are by default trusted for delegation. If a user account has the "account is sensitive and cannot be delegated" property set, the administrator instructs the KDC not to issue any forwardable tickets to that particular user account.

The "Use DES encryption types for this account" account property changes the default Kerberos encryption type from RC4 to DES (as explained in Section 5.4.3).

Every user account also has a "Do not require Kerberos preauthentication" property. This setting must be enabled when the account is used in a Kerberos implementation or application that supports preauthentication. This is usually the case in UNIX Kerberos implementations. Windows Kerberos preauthentication was discussed earlier in this chapter.

5.5.3 Kerberos transport protocols and ports

RFC 1510 defines that a Kerberos client should connect to a KDC (port 88) using the connectionless UDP protocol. Microsoft Kerberos by default uses UDP. Microsoft Kerberos can also use TCP to take advantage of TCP's bigger Maximum Transmission Unit (MTU) capacity. Microsoft uses TCP if the ticket size is bigger than 2 kB. Any ticket fitting in a packet of 2 kB is sent using UDP, which has a 1,500-octet MTU limit. The Windows 2000 Kerberos ticket can easily grow beyond this limit if it is carrying a large PAC field—this can, for example, occur when a user is a member of a large number of groups.

The default 2-kB limit can be changed using the following registry hack. Setting this value to 1 will force Kerberos to use TCP all the time.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\
Kerberos\Parameters
Value Name: MaxPacketSize
```

```
Data Type: REG_DWORD
Value: 1–2000 (in bytes)
```

Kerberos uses port 88 on the KDC side and a variable port on the client side. If your Kerberos clients communicate only with KerberosV5 KDCs (the Kerberos version used in Windows 2000 and Windows Server 2003), it is enough to keep port 88 open on your firewall. If they communicate with KerberosV4 KDCs, you must also open port 750. Table 5.10 gives an overview of all Kerberos-related ports.

5.5.4 Kerberos time sensitivity

Time is a critical service in Windows 2000 and Windows Server 2003. Timestamps are needed for directory replication conflict resolution, but also for Kerberos authentication. Kerberos uses timestamps to protect against replay attacks. Computer clocks that are out of sync between clients and servers can cause authentication to fail or extra authentication traffic to be added during the Kerberos authentication exchange.

To illustrate the importance of time for Kerberos authentication, let's look at what really happens during a KRB_AP_REQ and KRB_AP_REP Kerberos exchange:

1. A client uses the session key it received from the KDC to encrypt its authenticator. The authenticator is sent out to a resource server together with the ticket.

Table 5.10 *Kerberos-Related Ports*

Port	Protocol	Function Description
88	UDP TCP	Kerberos V5
750	UDP TCP	Kerberos V4 Authentication
751	UDP TCP	Kerberos V4 Authentication
752	UDP	Kerberos password server
753	UDP	Kerberos user registration server
754	TCP	Kerberos slave propagation
1109	TCP	POP with Kerberos
2053	TCP	Kerberos demultiplexer
2105	TCP	Kerberos encrypted rlogin

2. The resource server compares the timestamp in the authenticator with its local time. If the time difference is within the allowed time skew, it goes to step (4). By default, the maximum allowed time skew is 5 minutes—this setting can be configured through domain-level GPOs.
3. If step (2) failed, the resource server sends its local current time to the client. The client then sends a new authenticator using the new timestamp it received from the resource server.
4. The resource server compares the timestamp it received from the client with the entries in its "replay cache" (this is a list of recently received timestamps). If it finds a match, the client's authentication request will fail. If no match is found, client authentication has succeeded, and the resource server will add the timestamp to its replay cache.

The service responsible for time synchronization between Windows 2000, Windows XP, and Windows Server 2003 computers is the Windows Time Synchronization Service (W32time.exe). The Windows time service is compliant with the Simple Network Time Protocol (SNTP) as defined in RFC 1769 (available from http://www.ietf.org/rfcs/rfc1769.txt). SNTP makes sure that the computer clocks are within 20 seconds of each other. A protocol that can provide more accurate time synchronization than SNTP is the Network Time Protocol (NTP). NTP is defined in RFC 1305 (available from http://www.ietf.org/rfcs/rfc1305.txt). Because the Windows 2000 AD replication and Kerberos do not require the level of time accuracy offered by NTP, the Windows developers decided to implement the SNTP protocol as the time protocol for Windows 2000 and later OSs.

Basic SNTP operation

All Windows 2000, XP, and Windows Server 2003 machines have the W32Time service installed by default. In the service list the service is referred to as the Windows Time service—in Windows Server 2003 and XP it is started automatically. The time service will automatically perform time synchronization at machine startup and at regular intervals (initially every 8 hours).

At machine startup, the client contacts an authenticating domain controller and exchanges packets to determine the latency of communication between the two computers. W32time will determine what time the local machine time should be converged to—this time is referred to as the target time. If the target time is ahead of local time, local time is immediately set to the target time. If the target time is behind local time, the local clock is slowed over the next 20 minutes to align the two times, unless local time is more than 2 minutes out of synchronization, in which case the time is immediately set.

At regular intervals, the client machine will perform periodic time checks. To do this the client connects to the "inbound time partner" (the Windows authenticating domain controller) once each "period." The initial period is 8 hours. If the local time is off from the target time by more than 2 seconds, the interval check period is divided in half. This process is repeated at the next interval check until either:

- The local time and target time remain within 2 seconds of each other.
- The interval frequency is reduced to the minimum setting of 45 minutes.

If accuracy is maintained within 2 seconds, the interval check period is doubled, up to a maximum period of 8 hours.

The default time convergence hierarchy constructed in a Windows 2000 and Windows Server 2003 forest follows the following rules:

- All client desktops and member servers nominate as their inbound time partner the authenticating domain controller. If this domain controller becomes unavailable, the client reissues its request for a domain controller.
- All domain controllers in a domain nominate the primary domain controller (PDC) emulator Flexible Single Master Operation (FSMO) to be the inbound time partner.
- All PDC emulator FSMOs in the enterprise follow the hierarchy of domains in their selection of an inbound time partner.
- The PDC emulator FSMO at the root of the forest is authoritative and can be manually set to synchronize with an outside time source.

Many organizations also rely on an external time source for time synchronization. This usually means that the PDC emulator of their root domain synchronizes with an external time server. In organizations that have a Windows forest that is geographically spread out, an external time source for a DC in every geography may be preferred over one time source for the root domain only.

The external time source can be a time server on the Internet such as the server of the Swiss Federal Institute of Technology in Zurich. It can also be a time server appliance hosted in the enterprise—one of the companies sell-

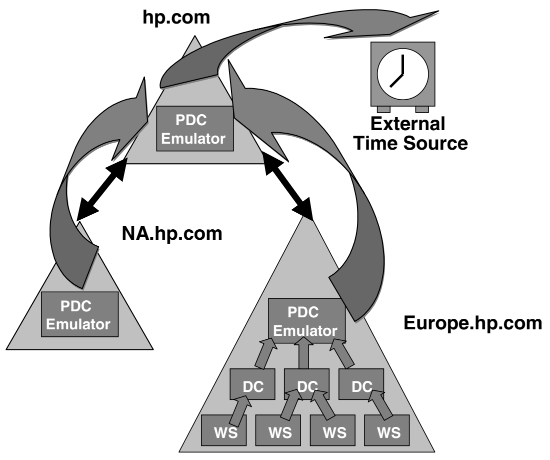


Figure 5.35
Sample SNTP hierarchy.

ing time server appliances is Symmetricom (previously known as Datum—more info at http://www.datum.com).

A sample SNTP hierarchy is shown in Figure 5.35. This default SNTP hierarchy can be modified by using the utilities that will be explained in the next section.

Configuring the windows time service

Microsoft provides two tools to configure and diagnose the Windows Time service:

- Net time—which is used to configure the time service and the synchronization hierarchy. The following net time command will change the time server on the local machine to mytimeserver.hp.com:

  ```
  Net time /setsntp:mytimeserver.hp.com
  ```

- w32tm—which is used to diagnose and configure the time service. For example, to monitor and analyze the time synchronization in the hp.com domain, I would type:

  ```
  w32tm /monitor /domain:hp.com
  ```

Both tools allow you to configure the time hierarchy to use the Windows defaults (as explained earlier in this section) or to use special designated time servers.

In Windows Server 2003, Microsoft added a new section in the GPO settings to configure the Windows Time Service. You can find it under Computer Configuration\Administrative Templates\System\Windows

Time Service. The time service's configuration data are kept in the following registry key: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\w32Time.

For many more Windows time service configuration details, read the Microsoft white paper available from http://www.microsoft.com/windows2000/docs/wintimeserv.doc.

5.6 Kerberos and authentication troubleshooting

In the next two sections, we will explore some basic Kerberos and Windows Server 2003 authentication troubleshooting tools. An indispensable tool for every administrator is the Event Viewer. The next section will list some common Kerberos error messages as they appear in the Event Viewer. The following side note explains how to enable advanced Kerberos event logging.

Enabling Advanced Kerberos Event Logging Advanced Kerberos event logging can be enabled using the following Windows registry hack. Set the Loglevel registry key (REG_DWORD) to value 1. Loglevel is located in the following registry key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parameters.

5.6.1 Kerberos error messages

In Windows Server 2003, Microsoft included some Kerberos-specific event IDs. They are listed in Table 5.11. If you want to go even more in detail, Table 5.12 shows the Kerberos-related error messages as they appear in the Windows Event Viewer. Both can give interesting hints when troubleshooting Kerberos authentication problems.

Table 5.11 *Kerberos-Specific Event IDs*

Event ID	Meaning
672	An authentication service (AS) ticket was successfully issued and validated.
673	A ticket granting service (TGS) ticket was granted.
674	A security principal renewed an AS ticket or TGS ticket.
675	Kerberos preauthentication failed. This event is generated on a key distribution center (KDC) when a user types in an incorrect password.

Table 5.12 *Kerberos Error Messages and Meaning*

Code	Short Meaning	Error Explanation
0x6	Client Principal unknown	The KDC could not translate the client principal name from the KDC request into an account in the Active Directory. To troubleshoot this error, check whether the client account exists in AD, whether it has not expired, and whether AD replication is functioning correctly.
0x7	Server Principal unknown	The KDC could not translate the server principal name from the KDC request into an account in the Active Directory. To troubleshoot this error, check whether the client account exists in AD, whether it has not expired, and whether AD replication is functioning correctly.
0x9	Null key error	Keys should never be null (blank). Even null passwords generate keys because the password is concatenated with other elements to form the key.
0xE	Encryption type not supported	The client tried to use an encryption type that the KDC does not support, for any of the following reasons: The client's account does not have a key of the appropriate encryption type; the KDC account does not have a key of the appropriate encryption type; the requested server account does not have a key of the appropriate encryption type. The type may not be recognized at all, for example, if a new type is introduced. This happens most frequently with MIT compatibility, where an account may not yet have an MIT-compatible key. Generally, a password change must occur for the MIT-compatible key to be available.
0x17	Password has expired	This error can be caused by conflicting credentials. Let the user log off and then log on again to resolve the issue.
0x18	Preauthentication failed	This indicates failure to obtain ticket, possibly due to the client providing the wrong password.
0x1A	Requested server and ticket do not match	This error will occur when a server receives a ticket destined for another server. This problem can be caused by DNS problems.
0x1F	Integrity check on decrypted field failed	This error indicates that there is a problem with the hash included in a Kerberos message. This could be caused by a hacker attack
0x20	Ticket has expired	This is not a real error; it just indicates that a ticket's lifetime has ended and that the Kerberos client should obtain a new ticket.
0x22	Session request is a replay	This error indicates that the same authenticator is used twice. This can be caused by a hacker attack.
0x19	Preauthentication error	The client did not send preauthentication, or did not send the appropriate type of preauthentication, to receive a ticket. The client will retry with the appropriate kind of preauthorization (the KDC returns the preauthentication type in the error).
0x25	Clock skew too great	There is time discrepancy between client and server or client and KDC. To resolve this issue, synchronize time between the client and the server.

Table 5.12 *Kerberos Error Messages and Meaning (continued)*

Code	Short Meaning	Error Explanation
0x26	Bad address in Kerberos session tickets	Session tickets include the addresses from which they are valid. This error can occur if the address sending the ticket is different from the valid address in the ticket. A possible cause of this could be an Internet Protocol (IP) address change. In Windows 2000, this change is dynamic and existing cached tickets could be invalidated. Another possible cause is when a ticket is passed through a proxy server. The client is unaware of the address scheme used by the proxy server, so unless the program caused the client to request a proxy server ticket with the proxy server's source address, the ticket could be invalid.
0x3C	Generic error	A generic error that may be a memory allocation failure. The event logs may be useful if this error occurs.
0x29	Kerberos AP exchange error	This indicates that the server was unable to decrypt the ticket sent by a client, meaning that the server does not know its own secret key, or the client received the ticket from a KDC that did not know the server's key. This can be tested by determining if the server can obtain a ticket to itself, or if anybody else can locate the server. The secure channel used by NTLM is also an indicator of the validity of the password on local machine accounts.

5.6.2 Troubleshooting tools

Microsoft delivers several tools to troubleshoot Kerberos (see Table 5.13). They are spread across the resource kit, the support tools, and the platform SDK. Most of them are command prompt tools.

Table 5.13 *Kerberos Troubleshooting Tools*

Tool	Comments
mytoken.exe (Platform SDK)	Command prompt tool to display the content of a user's access token: This includes the user's rights and group memberships.
whoami.exe (Default Windows installation)	Command line tool to look at the content of the user's access token (use the /all switch).
klist (Resource Kit)	Command prompt tool to look at the local Kerberos ticket cache. Klist can also be used to purge tickets. Klist is a very simple but very important tool that you can use to find out how far the authentication got.
Kerbtray (Resource Kit)	GUI tool that displays the content of the local Kerberos ticket cache.

Table 5.13 *Kerberos Troubleshooting Tools (continued)*

Tool	Comments
Netdiag (Support tools)	Netdiag helps isolate networking and connectivity problems by providing a series of tests to determine the state of your network client. One of the "NETDIAG" tests is the Kerberos test. To run the Kerberos test, type "netdiag /test:Kerberos" at the command prompt.
Replication monitor (replmon) (Support tools)	Using Replication monitor, an administrator can not only check the replication traffic but also the number of AS and TGS requests and the FSMO roles.
Network monitor (Server CD)	Network monitor does not come out of the box with a parser for the Kerberos protocol. However, a special Kerberos parser dll is available from Microsoft.
Setspn (Support Tools)	Tool allowing you to manage (view, reset, delete, add) service principal names (SPNs).

5.7 Kerberos interoperability

As mentioned earlier in this chapter, Kerberos is an open standard that is implemented on different platforms. Because of this Kerberos can be used as an SSO solution between Windows and other platforms.

5.7.1 Non-Windows Kerberos implementations

Table 5.14 lists other Kerberos implementations and the platforms on which they are available.

5.7.2 Comparing Windows Kerberos to other implementations

Before going into the details of the interoperability scenarios, it is interesting to look at what makes Windows 2000 and Windows Server 2003 Kerberos different from the other implementations. The Microsoft implementation of Kerberos is different in the following ways:

- It is tightly integrated with the Windows 2000 and Windows Server 2003 OS kernel: Every Windows 2000 and Windows Server 2003 system runs the Kerberos Security Support Provider (SSP) and every DC has a KDC service.

Table 5.14 *Non-Windows Kerberos Implementations*

Kerberos Implementation	Platform
MIT Kerberos v1.1	NetBSD
CyberSafe TrustBroker	UNIX, MVS, Windows 95, NT4
Sun SEAM	Solaris
DCE Kerberos (IBM)	AIX, OS/390
Computer Associates Kerberos [Platinum (OpenVision)]	Windows 95, 3.1, 3.11
Kerberos PAM	Linux, HP-UX
Heimdal	UNIX

- Kerberos principals locate the KDC using DNS. Windows 2000 and Windows Server 2003 DNS includes special SRV records that provide the location of a Kerberos KDC.
- MS implemented the RC4-HMAC encryption algorithm (56/128 bit keys) as the preferred Kerberos encryption type. MS still supports DES-CBC-CRC and DES-CBC-MD5 (56-bit keys) for interoperability reasons. See Section 5.4.3 for more information about this.
- The MS implementation does not support the MD4 checksum type.
- Windows Kerberos KDCs require Kerberos clients to perform preauthentication by default. More information about this is available in Section 5.5.2.
- The MS implementation does not include support for DCE-style cross-realm trust relationships.
- Microsoft uses their proprietary SSPI API (see Chapter 4) to access Kerberos services. They do not support the raw krb5 API.
- Microsoft uses the authdata field in the ticket to embed authorization data. Microsoft refers to this field as the Privilege Attribute Certificate (PAC). See also Section 5.4.3 for more information about this.

5.7.3 Interoperability scenarios

In this section we will focus on setting up Kerberos interoperability between Windows 2000 or Windows Server 2003 Kerberos and a Kerberos implementation that runs on top of UNIX platforms. Kerberos authentication interoperability can be set up in three different ways:

- The Windows Kerberos KDC is the KDC for both Windows and UNIX security principals (the principals are administered from the Windows KDC) (Scenario 1).
- The UNIX KDC is the KDC for both Windows and UNIX security principals (the principals are administered from the UNIX KDC). This scenario includes no Windows domain controllers (Scenario 2).
- A cross-realm trust relationship is defined between a Windows domain and a UNIX Kerberos realm. A part of the principals is administered from the Windows KDC, another part is administered from the UNIX KDC. In this case, there are two KDCs—one KDC on each side of the trust relationship (Scenario 3).

Next we will explain these three scenarios using examples of Windows-UNIX Kerberos authentication interoperability.

Lots of valuable information on how to set up interoperability can be found in the following white papers:

- "Windows 2000 Kerberos interoperability" available from http://www.microsoft.com/windows2000/techinfo/howitworks/security/kerbint.asp
- "Windows 2000 Kerberos Interoperability" by Christopher Nebergall available from http://www.sans.org/rr/paper.php?id=973

Principals defined on a Windows KDC

This scenario allows for Kerberos principals on both Windows and non-Windows platforms to log on using Windows credentials and a Windows KDC. To enable a user to log on to Windows from a UNIX workstation, the UNIX krb5.conf Kerberos configuration file must be edited to point to the Windows KDC. Afterward, the user can log on using his or her Windows account and the "kinit" command (kinit is the equivalent of logon in UNIX Kerberos implementations).

The setup gets a little bit more complicated when enabling a service, running on a UNIX platform, to log on using Windows credentials and a Windows KDC. In this scenario the Windows administrator has to run through the following configuration steps:

- Edit the krb5.conf file on the UNIX machine to point to the Windows KDC.

- Create a service account for the UNIX service in the Active Directory.
- Use ktpass.exe to export the newly created service account's credentials from AD and create a keytab file. The keytab file contains the password that will be used by the UNIX service to logon to the Windows domain.
- Copy the keytab file to the UNIX host and merge it with the existing keytab file.

Ktpass comes with the Windows support tools. It allows an administrator to configure a UNIX computer or UNIX-based service as a security principal in the Active Directory.

The following example illustrates this last scenario. A company has a UNIX database server whose content should be accessible through a Web interface for every Windows user. To set this up, configure the database service as a principal in the Windows domain, and install an IIS server as a Web front end for the database server. To allow for credential forwarding between a Windows user and the IIS server, the IIS server must be "trusted for delegation."

Principals defined on a non-Windows KDC

This scenario allows for Kerberos principals on both Windows and non-Windows platforms to log on using UNIX credentials and a UNIX KDC. For a stand-alone Windows workstation or member server to use a UNIX KDC, the following has to be done:

- Create a host for the workstation in the UNIX realm.
- Configure the Windows workstation or member server using ksetup.exe to let it point to the UNIX KDC and realm and to set the machine password (this will automatically switch the workstation or member server to workgroup mode). Ksetup.exe also comes with the Windows support tools.
- Restart the workstation or member server and run ksetup.exe again to map local machine accounts to UNIX principals.

Cross-realm trust

This is probably the most flexible interoperability scenario available. This scenario will enable non-Windows Kerberos principals to log on to their UNIX KDC and to access resources in a Windows domain and also the other way around: For Windows principals to log on to their Windows KDC and to access resources in a UNIX realm.

The setup of a cross-realm trust between Windows and a UNIX realm is relatively straightforward. On the Windows side two things must be done: A trust relationship must be created using the AD Domains and Trusts snap-in, and a realm mapping for the UNIX realm should be added to the system registry. To add a realm mapping, use the ksetup tool. A realm mapping should not only be added on the Windows Domain controller, but also on every machine from which resources will be accessed in the UNIX realm. On the UNIX side, a trust relationship can be created using the kadmin tool.

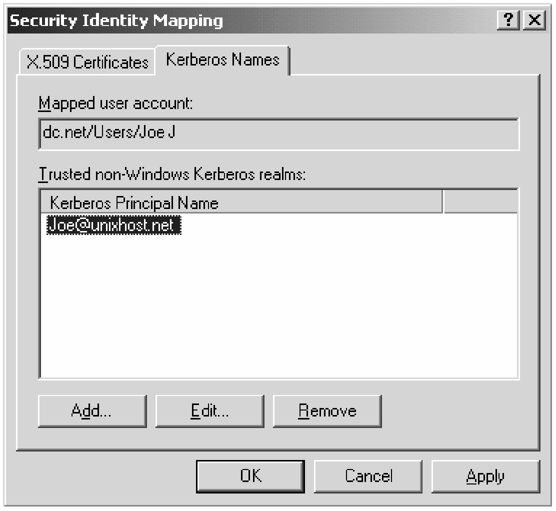
If all user accounts are defined in the UNIX realm, a domain layout that is very similar to the NT4 master domain model of account domains and resource domains is created. In that case, the UNIX realm acts as a master account domain containing all the accounts. The Windows domain acts as a resource domain, containing resources and mappings from the UNIX accounts to Windows SIDs.

If you are planning to use this domain-realm layout, some extra configuration, besides the creation of a cross-realm trust, is needed on the Windows side. The reason for this is the difference in the accounting and aauthorization systems used in Windows and UNIX. Whereas UNIX relies on principal names for both accounting and authorization, Windows relies entirely on security identities (SIDs). Even though there is a trust relationship between the UNIX realm and the Windows domain, users authenticated through the KDC in the UNIX account domain can by default not access any resource in the Windows, because they do not have an SID.

To resolve this problem, shadow or proxy accounts must be created on the Windows side. A proxy account is an attribute (the "altSecurityIdentities" attribute) of a Windows account that contains a UNIX principal name. In other words, proxy accounts provide a way to map a UNIX account to a Windows account or SID.

Setting Up Kerberos Proxy Accounts Kerberos proxy or shadow accounts can be defined from the AD Users and Computers MMC snap-in.

- In the MMC snap-in's View menu option, select Advanced Features…
- Right-click the user account for which you want to define the proxy account and select Name Mappings… This will bring up the Security Identity Mapping dialog box (illustrated in Figure 5.36).
- Select the Kerberos Names tab, and then Add… to add the UNIX Kerberos Principal Name.



Mapped user account:
dc.net/Users/Joe J

Trusted non-Windows Kerberos realms:

Kerberos Principal Name
Joe@unixhost.net

Figure 5.36
Defining Kerberos account mappings.

Let's look at what will happen now when a UNIX principal wants to access a resource that is hosted on a machine that is a member of a Windows domain (illustrated in Figure 5.37):

- The UNIX principal is logged on to the UNIX domain; his or her credential cache contains a TGT for the UNIX domain.
- The UNIX principal wants to access resource A in the Windows domain. His or her local KDC refers him to the Windows KDC.
- The Windows KDC creates a new service ticket for the UNIX principal to access resource A. Because the TGT used by the UNIX principal contains an empty PAC, the Windows KDC will query the Active

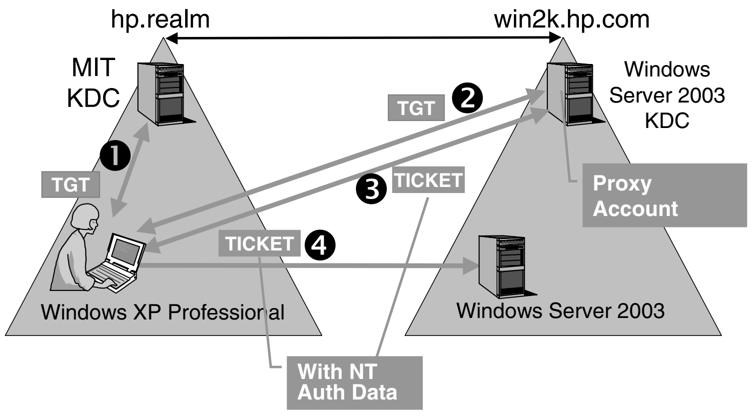


Figure 5.37
UNIX-Windows Server 2003 Kerberos interoperability using a cross-realm trust.

Directory for an account mapping between the UNIX principal and a Windows SID. The newly issued service ticket will contain the PAC data corresponding to the Windows SID.

- Using the service ticket, the UNIX principal authenticates to the machine hosting resource A.

In case the Windows domain also contains NT4 servers that can only authenticate using NTLM, this scenario also requires some password synchronization tool between the UNIX Realm (where the accounts and their passwords are defined) and the Windows domain. Such a tool is available in CyberSafe's Trustbroker product.