

ITPro™
SERIES

WindowsITPro eBooks

By Jan De Clercq

Understanding and Leveraging

SSL-TLS for Secure Communications



Contents

Chapter 2: Leveraging SSL/TLS for Secure Web Communications	21
Setting Up SSL/TLS on a Web Server	21
Generating an SSL Server Certificate Request	21
Generating the SSL Server Certificate	25
Installing the SSL Server Certificate on Your Web Server	27
Configuring SSL on Your Web Server	28
Getting SSL Client Certificates	31
Client-Side SSL Configuration and Usability Features	32
Comparing SSL to Other Web Client Authentication Protocols	36
HTTP Authentication	36
Basic Authentication	38
Digest Authentication	39
Web Client Authentication Protocol Comparison	40
Conclusion	41

Chapter 2:

Leveraging SSL/TLS for Secure Web Communications

This chapter explains how you can leverage Secure Sockets Layer/Transport Layer Security (SSL/TLS) for Secure Web (HTTP) communications. The combination of the HTTP protocol and SSL is also referred to as “HTTP over SSL,” or, in short, HTTPS. In addition, the chapter explains how to set up SSL/TLS on the Web server side, outlines SSL client-side configuration features, and compares SSL client certificate-based authentication to other Web authentication methods.

Setting Up SSL/TLS on a Web Server

The use of SSL/TLS requires X.509 certificates (see chapter 1 of this eBook for an introduction to X.509 certificates). You always need a server certificate, and if you want client certificate-based strong authentication, you also need client certificates. To illustrate the SSL/TLS Web server setup, we will use the Microsoft Web server, Internet Information Services (IIS) 6.0, as an example. IIS 6.0 is the Web server that comes with the Windows Server 2003 operating system.

In addition, a detailed, step-by-step guide on how to enable IIS for SSL, “Securing Your Microsoft MSIS Web Server with a Thawte Digital Certificate,” is available from the Thawte Web site (<http://www.thawte.com/ucgi/gothawte.cgi?a=w44600158747054000>). Thawte also provides an SSL setup guide for the Apache Web server, “Securing Your Apache Web Server with a Thawte Digital Certificate” (<http://www.thawte.com/ucgi/gothawte.cgi?a=w44570158747048000>).

Setting up SSL in an IIS 6.0 environment typically includes the following steps:

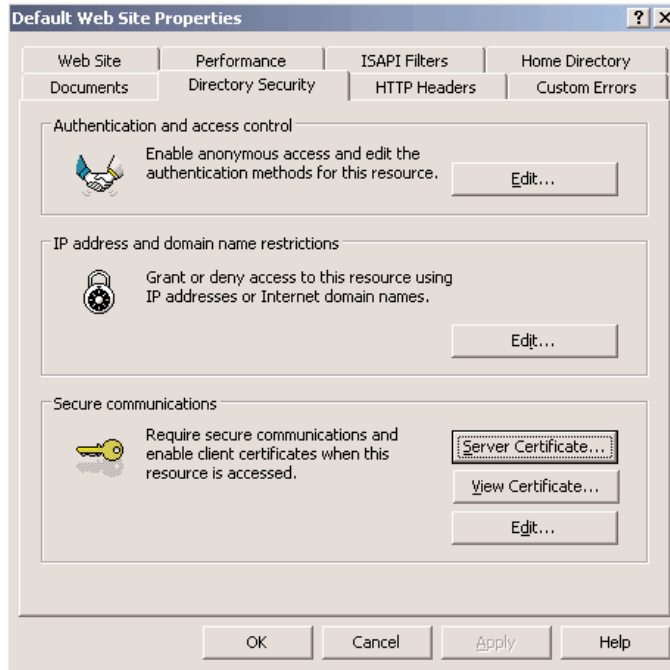
1. Generate a server certificate request file.
2. Generate the server certificate.
3. Install the server certificate on the Web server.
4. Configure SSL on the Web server.
5. Optionally, generate, acquire, and install client certificates.

To familiarize yourself with the process of requesting and setting up an SSL server certificate for your IIS Web server, you can request a test certificate from Thawte. To request the certificate, go to the following URL: <http://www.thawte.com/ucgi/gothawte.cgi?a=w46840165357049000>.

Generating an SSL Server Certificate Request

The easiest way to generate an SSL server certificate request is to use the IIS Web Server Certificate Wizard (illustrated in Figure 2-1), which guides you through the request file-generation process.

Figure 2-1
SSL Web Server Certificate Wizard



You can start the wizard from the Microsoft Management Console (MMC) IIS Internet Services Manager snap-in. Right-click the Web site on which you want to set up SSL, and select Properties. Then, in the “Secure communications” section of the Directory Security tab, select Server Certificate (as Figure 2-2 illustrates).

Figure 2-2
Starting the Web Server Certificate Wizard



In the wizard, you have the option to generate the request offline (the “Prepare the request now, but send it later” option, illustrated in Figure 3) or to send the SSL server certificate request immediately to an online CA (the “Send the request immediately to an online certification authority” option, also shown in Figure 3). The second option will both generate and send the certificate request file and automatically install the certificate. This option will work only if you have an operational Windows enterprise certification authority (CA) in your IIS environment. A Windows enterprise CA is an AD-integrated CA that is published in AD.

Figure 2-3*IIS Certificate Wizard: delayed or immediate request options*

When you choose to generate the request offline, you will have to manually submit the SSL certificate request file to the CA and manually install the certificate on the Web server. You typically choose this option when you are requesting an SSL server certificate from a commercial CA, such as Thawte.

In the Web Server Certificate Wizard, you must enter the following information:

- Organization name
- Organizational units
- Country code
- State or province
- Locality
- Common name

An important step in the certificate-request file-generation wizard process is specifying the Web site's common name. You must make sure that the common name you enter in the wizard matches the name that the browser uses in the URL to securely connect to your Web site. These names must match because the common name is X.509 terminology for the name that uniquely identifies the SSL server certificate, and that identifier links the certificate to your organization. The common name appears in the server certificate, and if the name in the certificate does not match the name in the URL, the browser will generate an SSL error.

Microsoft supports the use of wildcards in the certificate's common name. Using wildcards in the SSL certificate lets you reuse the same certificate for different physical Web sites. Doing this might be an interesting option for Web farms, in which the only difference among the different Web farm members is the machine.

The IIS SSL/TLS provider supports the following wildcards. You can use wildcards for whatever URL you want; all the examples match `www.mydomain.com`:

- `*.mydomain.com`
- `w*.mydomain.com`
- `*w.mydomain.com`.

You should never use wildcards in your organization's domain name (the "mydomain.com" portion); doing so would compromise SSL server authentication. Microsoft's support for wildcards is in line with RFC 2595 and is documented in the Microsoft article "Accepted Wildcards Used by Server Certificates for Server Authentication" (<http://support.microsoft.com/default.aspx?scid=kb;en-us;258858>).

If you don't immediately submit the SSL server certificate request to an online Windows enterprise CA—in other words, when you submit the request to a non-AD-integrated internal CA or to a commercial CA (such as Thawte or Verisign)—you must save your request to a file (by default, this file is named certreq.txt). If you open this file, you will find text that is formatted similar to the following text:

```
---BEGIN CERTIFICATE REQUEST---
MIIB2TCCAUICAQAwZgxCzAJBgNVBAYTALVTMRAWdgYDVQQIEwdHZW9yZ2lhmREwDwYDVQQQ
HEwhDb2x1bWJ1czEzEjEzMBkGB1UEChMSQUZMQUMgSW5jb3Jwb3JhdGVkMQswCQYDVQQLLEwJVV
DEYMBYGA1UEAxMPd3d3LmFmbGFjbknkuY29tMSAwHgYJKoZIhvcNAQkBFhFKR2FybW9uQGfMmb
GFJlNmVbTCBnzANBGlkqkhiG9w0BAQEFAAOBjQAwGykCgYEAAsRqHZCLIRlxqqh8qs6hCC0KR9qEPX
2buwMA6GxegICKp0i/IYY5+Fx3KZWXmta794nTPShh2LmRdn3iwxwQRKyqYKmp7wHCwtNm2taCRV
oboCQ0uyZjs+DG9mj+b0rMK9rLME+9wz1f8L0FuArWhedDBnI2sm0KQID45mWwB0hkCAWAAAaAA
MA0GCSqGSIb3DQEBAUAA4GBAJNlxh0iv9P8cdjMsqyM0WXXWgagdRaGoa8tv8R/U0uB0S8/H
qu73umaB9vj6VHY7d9RKqDELfc/xLXeDwoXNiF8quTm43pmY0WcqnL1JZDGHMQkzzGtG502CLTHM
ELUGTdKpAK6rJKkucP0DKKEJKcmTySSnvgUu7m
---END CERTIFICATE REQUEST---
```

Generating the SSL Server Certificate

SSL/TLS server certificates can be generated in different ways:

- An internal Certification Authority (CA) can generate the certificates. For example, this internal CA can be a Microsoft Windows 2000 or Microsoft Windows Server 2003 CA.
- An external commercial CA (e.g., Thawte or Verisign) can generate the certificates.
- You can use the *Internet Information Services (IIS) 6.0 Resource Kit* utility SelfSSL to generate self-signed certificates.

Once you have generated the SSL server certificate request file, you must use the file to generate the actual certificate. SelfSSL is a command-line tool that lets you generate self-signed SSL/TLS certificates; in other words, you do not need a CA or PKI infrastructure. SelfSSL not only generates a self-signed certificate but also installs the certificate on the Web server. If you use SelfSSL with the /T switch, the tool also adds the self-signed certificate to the list of trusted certificates in the local machine's certificate store.

If you did not submit the request file to an online Windows enterprise CA, or if you did not use the SelfSSL tool, you must manually submit the request to the CA that will issue the server certificate. If the CA you are using is an internal Windows CA, the easiest way to submit your SSL server certificate-request file is to use the CA's Web interface. To do so, connect to the CA's Web site using the `http://<servername>/certsrv` URL. On the welcome page, select "Request a certificate," then select "Submit an advanced certificate request." Finally, select "Submit a certificate request by using a base64-encoded CMC or PKCS#10 file, or submit a renewal request by using a base64-encoded PKCS#7 file." The next Web page will then let you paste the base64-encoded content of the certificate

request file (the part that starts with “—BEGIN NEW CERTIFICATE REQUEST—” and ends with “—END NEW CERTIFICATE REQUEST—”) into the box on the page, or to browse for a file to insert as the request content, as Figure 2-4 illustrates.

Figure 2-4
CA Web interface: Inserting the SSL certificate-request content

Microsoft Certificate Services -- RootCA [Home](#)

Submit a Certificate Request or Renewal Request

To submit a saved request to the CA, paste a base-64-encoded CMC or PKCS #10 certificate request or PKCS #7 renewal request generated by an external source (such as a Web server) in the Saved Request box.

Saved Request:

Base-64-encoded certificate request (CMC or PKCS #10 or PKCS #7):

```
MIID0zCCAqQCAQAQAwYDELMAkGA1UEBhMCQkUxEDAO
BgNVBACoTBUV2ZXXJlMQ0wCwYDVQQKEwRUZSNOHQ0w
VQQDEWh3MmszLURjMTCBnzANBgkqhkiG9w0BAQEF
Y7zAEUfUuU9qamVnzdlIaTSxBpZP1W5LhzFXrFTcj9
nCYIBciLS1F3pZccqx/uCgajXp1ONM4Q+ORjY1KdQ
/Ec4ospYQIzZ6Zk92ksd85TNhyO114fEVEcCAwEA
```

[Browse for a file to insert.](#)

Certificate Template:

Administrator

Additional Attributes:

Attributes:

Submit >

If you want to submit your SSL server certificate-request file to a commercial CA such as Thawte, you must follow the enrollment instructions the commercial CA provides. These instructions are different for each commercial CA and for different SSL server certificate types. For an overview of the Thawte enrollment instructions, consult “Thawte’s Quick Enrollment Guide: SSL Web Server Certificate, SGC SuperCert and Code Signing Certificate” (http://www.thawte.com/guides/pdf/enroll_sum_eng.pdf).

When you examine the SSL certificate offerings of different commercial CAs, you will notice that the CAs offer different SSL certificate types. The differences between these certificates sit primarily in the level of server authentication they provide—in other words, in the effort the commercial CA puts into making sure the server’s identity is authentic before the CA issues the SSL server certificate. The commercial CA can simply rely on checking whether the Web site has a registered DNS name and whether the requesting party has the right to request or install a CA on that domain – or it can start more advanced identity checks that include third parties, such as auditors and notaries. The complexity and quality of the Web site authentication procedure also have a direct impact on the price of the SSL certificate. Another differentiator for the SSL certificates commercial CAs offer is their support for a mechanism called Server-Gated Cryptography (SGC), which we explain in more in detail below.

Thawte provides three types of SSL server certificates:

1. *SSL Web Server Certificates*. The prime characteristic of these standard Thawte SSL server certificates is that they offer a high level of authentication. This high level of authentication explains why their enrollment procedure is more complex and takes more time than SSL123 Certificates (explained below). How to enroll for a standard SSL Web Server Certificate is detailed in the “Enrollment Guide for Thawte SSL Web Server Certificates and SGC SuperCerts” (http://www.thawte.com/guides/pdf/enroll_ssl_eng.pdf).
2. *SSL123 Certificates*. These certificates offer a lower level of authentication than the standard SSL Web Server Certificates. SSL123 Certificates validate only that your domain is registered and that you have authorized the purchase of the certificate. These certificates are cheaper because of the lower level of authentication. They also require less time for enrollment than the standard SSL Web Server Certificates. How to enroll for a SSL123 Certificate is detailed in the “Enrollment Guide for Thawte SSL123 Certificates” (http://www.thawte.com/guides/pdf/enroll_ssl123_eng.pdf).
3. *SGC SuperCert Certificates*. These SSL server certificates incorporate SGC technology, which lets certain Windows clients that have encryption restrictions automatically step up to 128-bit SSL encryption. These clients include Windows 2000 editions shipped before March 2001 that do not have the Windows 2000 High Encryption Pack or Windows 2000 Service Pack 2 installed. These systems prevent 128-bit encryption, regardless of the version of Internet Explorer (IE) being used. A Thawte SGC SuperCert lets SSL-secured Web sites extend 128-bit encryption to all their clients who use browsers (Microsoft Internet Explorer [IE] 4.X or Netscape 4.06 and later) that are limited to 40-bit or 56-bit encryption capabilities.

For many years, the U.S. government restricted U.S. vendors from exporting “strong” cryptography. While these restrictions were in place, many Internet users around the world downloaded or purchased computers with export version browsers, browsers capable of supporting only 40-bit or 56-bit SSL encryption. To allow vendors to adhere to the export restrictions while still addressing the need for secure communications outside the United States, Microsoft developed SGC and Netscape developed “step-up” technology.

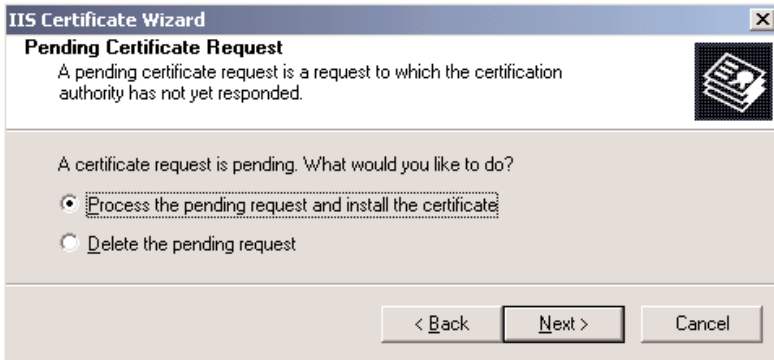
SGC lets users with an export version browser temporarily step up to 128-bit SSL encryption if they visit a Web site with an SGC-compatible SSL certificate, such as Thawte’s SGC SuperCert. Without an SGC certificate on the Web server, Web browsers and PCs that do not inherently support 128-bit strong encryption will receive only 40-bit or 56-bit encryption.

How to enroll for a SGC SuperCert SSL Web server certificate is detailed in the “Enrollment Guide for Thawte SSL Web Server Certificates and SGC SuperCerts” (http://www.thawte.com/guides/pdf/enroll_ssl_eng.pdf).

Installing the SSL Server Certificate on Your Web Server

Once the certificate has been generated, you can install it on your Web server. Once more, you can use the Web Server Certificate Wizard, or you can use the IIScertdeploy.vbs script that comes with the *IIS 6.0 Resource Kit* to install the certificate. If you use the wizard, this time select the “Process the pending request and install the certificate” option (illustrated in Figure 5). In the following screens, you will then be prompted to point the wizard to the newly generated Web server SSL certificate.

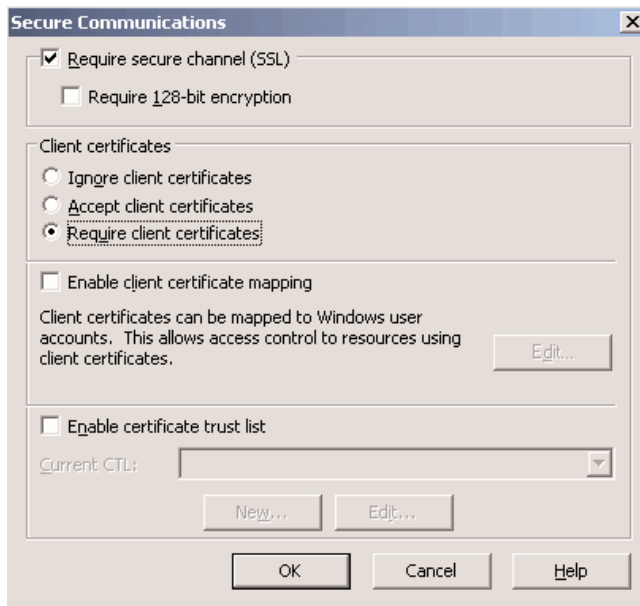
Figure 2-5
IIS Certificate Wizard: process request option



Configuring SSL on Your Web Server

In the MMC IIS Internet Services Manager snap-in, you can configure SSL/TLS using the Edit... button in the "Secure communications" section of the Directory Security tab, (see also Figure 2-2). This button is enabled only if you have successfully installed a server certificate. Figure 2-6 illustrates the configuration screen.

Figure 2-6
Configuring SSL/TLS

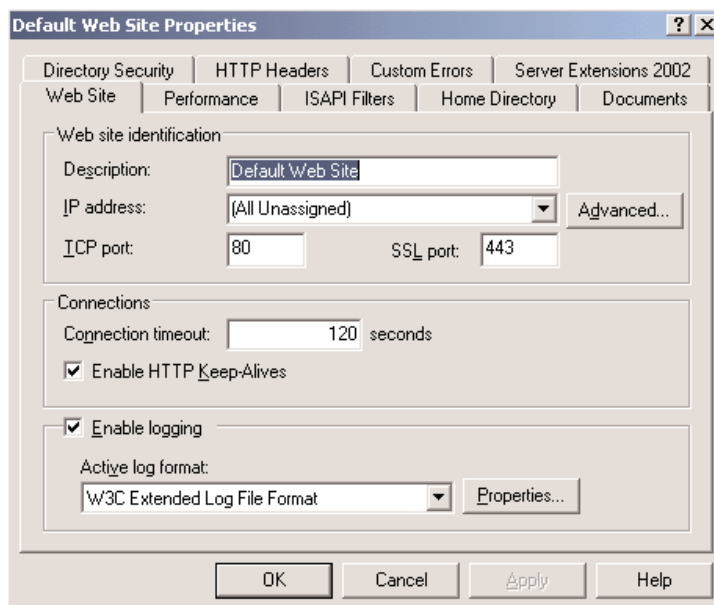


In the configuration screen, you can

- Enable/disable SSL for a Web site.
- Require 128-bit encryption, which provides extra protection for Web sites that deal with sensitive personal and financial information.
- Set client certificate-based authentication options. Both the “Accept client certificates” and “Require client certificates” options require the deployment of certificates to your browser clients “Require client certificates” means that users can access your Web site only if they have an SSL client certificate installed. “Accept client certificates” means that users also can access your Web site if they don’t have an SSL client certificate installed; in that case, they can authenticate to the Web server using another HTTP authentication method. We discuss the other HTTP authentication methods in more detail in the section “Comparing SSL to Other Web Client Authentication Protocols.” As with SSL server certificates, you can acquire SSL/TLS client certificates from an internal CA or from an external commercial CA.
- Enable certificate mapping (explained later in this section).
- Enable certificate trust lists (CTLs) (explained in chapter 3 of this eBook).

If you want to configure the port that is used for SSL communications, you can do so from the Web Site tab in your Web site’s Default Web Site Properties dialog box (as Figure 2-7 illustrates). The default port used for SSL is 443. The SSL port field is enabled only if you have installed an SSL server certificate on your Web server. Note that different Web sites can use the same SSL port provided they have different IP addresses.

Figure 2-7
Configuring SSL/TLS: SSL port



30 Understanding and Leveraging SSL-TLS for Secure Communications

To facilitate Web server access-control enforcement, you can map IIS SSL client certificates to Windows security identities. This feature is called *certificate mapping*. Certificate mapping lets you apply authorization settings that are defined for Windows security identities to users who have used an SSL client certificate to authenticate to IIS.

You can define certificate mapping in the IIS metabase (this is the IIS configuration database) or in the AD. Defining certificate mapping in the AD uses a service known as the “Windows directory service mapper.” AD-based mapping is an interesting option if you have multiple Web servers that all need certificate mappings defined. Instead of defining the mappings on every individual Web server, you can define them once in the central AD repository.

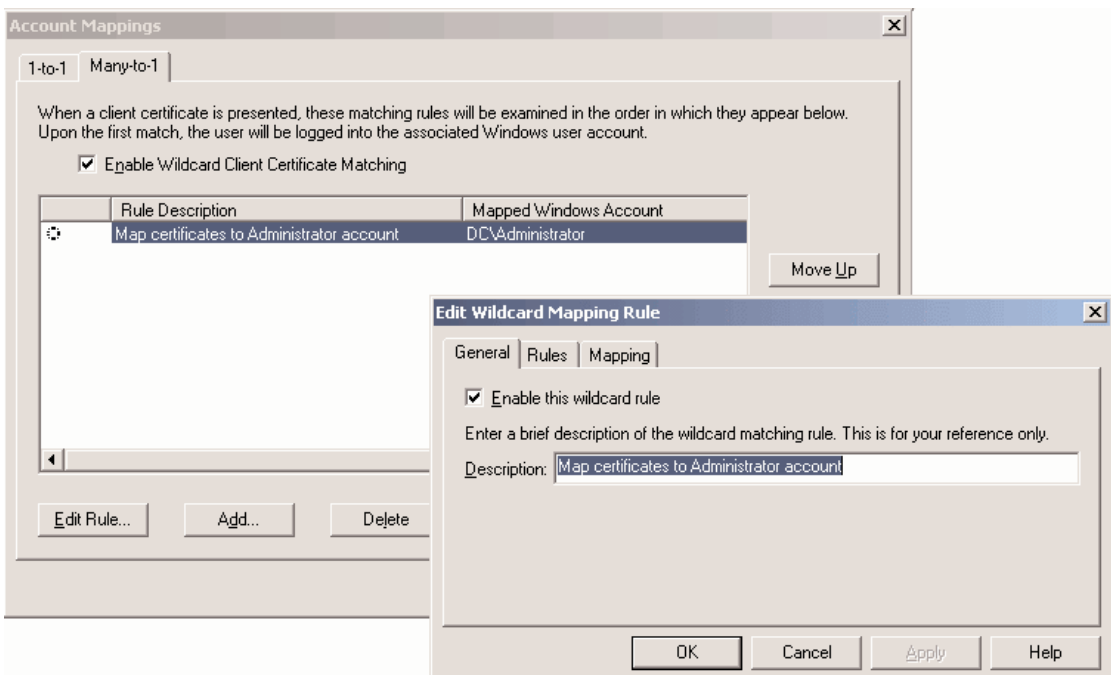
You can define IIS metabase-based mapping from the Internet Services Manager (ISM) Secure Communications dialog box (illustrated in Figure 2-6). This option is available only if you have selected the “Enable client certificate mapping” checkbox.

You can set up certificate mapping defined in the metabase in one of two modes: 1-to-1 mapping and Many-to-1 mapping.

- IIS looks at the complete content of the SSL client certificate in 1-to-1 mapping to map the certificate to a Windows security identity.
- Many-to-1 certificate mapping is based on rules (as Figure 2-8 illustrates). In this case, IIS looks at particular attributes of the SSL client certificate (as defined in the rules) to map the certificate to a Windows security identity.

Figure 2-8

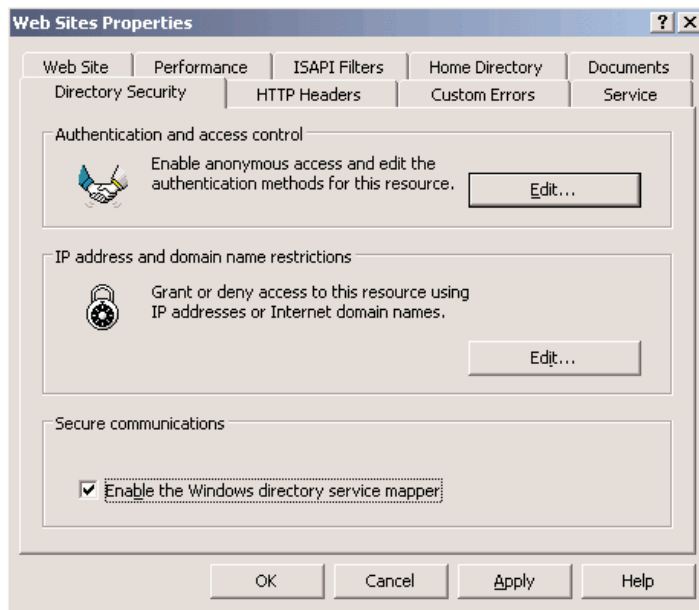
Setting up a Many-to-1 certificate-mapping rule in the IIS Internet Service Manager (ISM)



You can define AD-based mapping from the MMC Active Directory Users and Computers MMC snap-in. Use the “Name mappings...” option in an account object’s context menu—this option is available only if the snap-in is in Advanced Features viewing mode. AD-based mapping allows only for 1-to-1 mapping. AD-based mapping, or the Windows directory service mapper, is enabled from the Properties dialog box of the Web Sites container in the ISM (as Figure 2-9 illustrates).

Figure 2-9

Enabling the Windows directory service mapper



Getting SSL Client Certificates

If you also want to use certificates on the browser side to strongly authenticate your SSL clients, you must make sure that you provide your clients with SSL client certificates. Not all organizations want to do this, so this step is optional. A typical scenario that requires SSL client certificates is a secure extranet Web site. Note that SSL client certificates are not the only client authentication method available. We discuss other methods in the section “Comparing SSL to Other Web Client Authentication Protocols.”

You can request client SSL/TLS certificates from an internal CA or an external CA. If you are using a Windows Server 2003–rooted PKI, users can request certificates using their MMC Certificates snap-in or the CA’s Web interface (accessible using <http://<servername>/certsrv>). Administrators can also automatically enroll users for SSL/TLS certificates using a Group Policy Object (GPO) setting.

An important—but often forgotten—last step is to make sure that the CA that issued the client and server SSL certificates is trusted by your clients. In Windows, ensuring that the CA is trusted comes down to making sure that the CA’s certificate is stored in the client’s trusted root certificate store. To look at the content of this store, open the MMC Certificates snap-in. Trust is not a real

concern when you're using certificates that are issued by a commercial CA, such as Thawte or Verisign. These commercial CA's certificates come with the Windows OS software and are trusted by default. Trust is an issue, however, if you're using certificates generated by an internal PKI that is run by your organization or a partner organization.

If you're using a certificate generated by an internal PKI—if the CA's certificate doesn't come preinstalled with the OS software—you can use different methods to add the certificate of the CA that issued the server and client certificates collectively to all your user browsers' trusted root certificate stores:

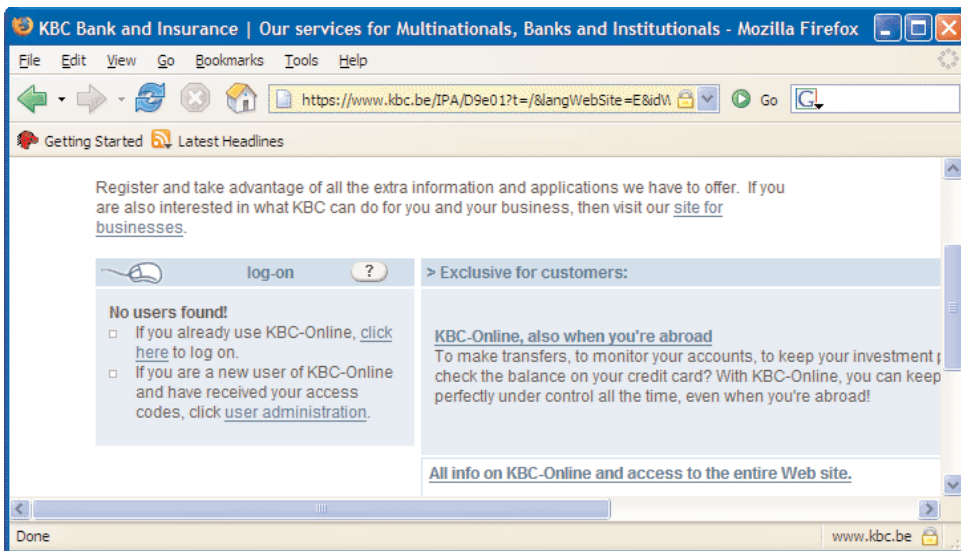
- Use the Internet Explorer Administration Kit (IEAK) to create an IE installation kit that adds the CA's certificate to the trusted root certificate store.
- Put the CA's certificate on a publicly accessible Web site where users can download and choose to trust it.
- Distribute the CA's certificate using a GPO setting.

Client-Side SSL Configuration and Usability Features

Today, all Web browsers come with built-in support for the SSL/TLS protocols. The most commonly used Web browsers today are the Microsoft Internet Explorer (IE), Netscape Navigator, Mozilla Firefox, and Opera Web browsers. All of these browsers inform their users when they access an SSL-secured Web site, and they provide an interface so you can look at the content of a Web site's SSL certificate and the client's SSL certificates.

Typically, users are informed of an SSL-secure Web site by a lock symbol that shows up in the browser's tray (as Figure 2-10 illustrates for the Mozilla Firefox browser).

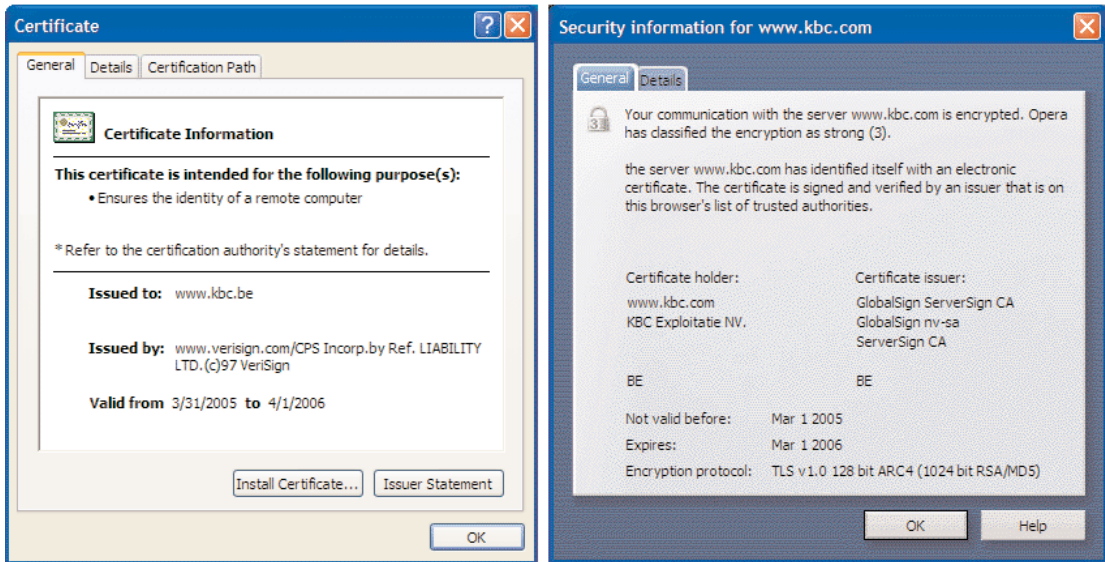
Figure 2-10
SSL/TLS lock icon in Mozilla Firefox



If you double-click the lock icon, you get an overview of the Web site's SSL/TLS certificate details (as Figure 2-11 illustrates for the Internet Explorer and Opera browsers).

Figure 2-11

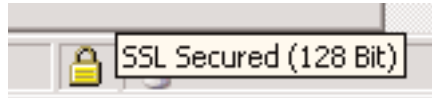
SSL/TLS certificate viewer in the Internet Explorer and Opera Browsers



The browsers' SSL certificate viewers display an SSL certificate's content and its properties. The General tab of the IE SSL certificate viewer, for example, shows general information such as the applications the certificate can be used for, the subject name, the issuer name, the validity period of the certificate, and whether the certificate has a corresponding private key stored in the user's profile. The Details tab shows all X.509 certificate entries: the standard fields, and the critical and noncritical certificate extensions. The Certification Path tab shows the certification path and the certificate status.

Another simple way to check whether an SSL-secured session has been set up between your browser and a Web server is to check the URL. If a secured session has been set up, the http section of the URL will change from "http://" to "https:". For example, instead of <http://www.thawte.com>, your browser will show <https://www.thawte.com>.

The IE browser also informs users about the SSL encryption strength that is used in a particular SSL session. Simply move your mouse over the lock symbol, and IE will show the SSL encryption strength (as Figure 2-12 illustrates). In the other browsers, you must double-click the lock symbol and look at the certificate properties to find out the SSL encryption strength.

Figure 2-12*SSL/TLS certificate encryption strength*

Web browsers differ in the SSL/TLS configuration options they offer. These options include

- The capability to switch on/off SSL 2.0/SSL 3.0/TLS 1.0 support
- The capability to configure CRL-based certificate revocation checking
- The capability to configure Online Certificate Status Protocol (OCSP)-based certificate revocation checking

Table 2-1 gives an overview of the SSL/TLS configuration options of different Web browsers. Some of these configuration options (such as the support for CRL and OCSP revocation checking) are explained in more detail in chapter 3.

Table 2-1 Comparison of browser SSL/TLS configuration options

	Internet Explorer 6.0	Netscape Navigator 8.0	Mozilla Firefox 1.0	Opera 8.0
SSL 2.0 support	Yes	Yes	Yes	Yes
SSL 3.0 support	Yes	Yes	Yes	Yes
TLS 1.0 support	Yes	Yes	Yes	Yes
CRL support	Yes	Yes	Yes	Yes
OCSP support	No ¹	Yes	Yes	No

¹ Microsoft will embed OCSP support in the IE 7.0 release

Figure 2-13 shows the IE SSL/TLS configuration interface (which is accessible from the Advanced tab in the IE Internet Options dialog box).

Figure 2-13
IE SSL/TLS configuration interface

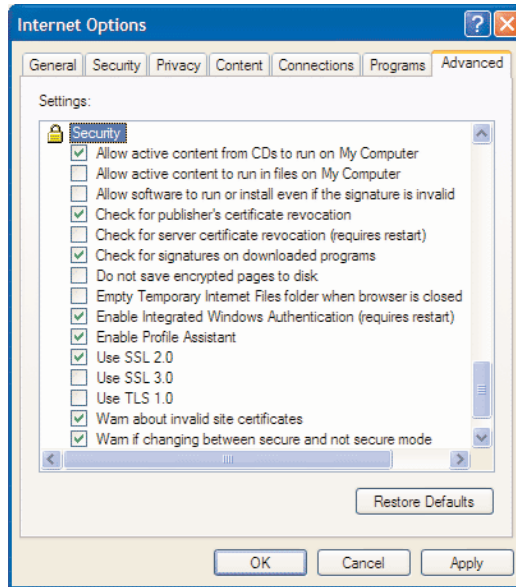
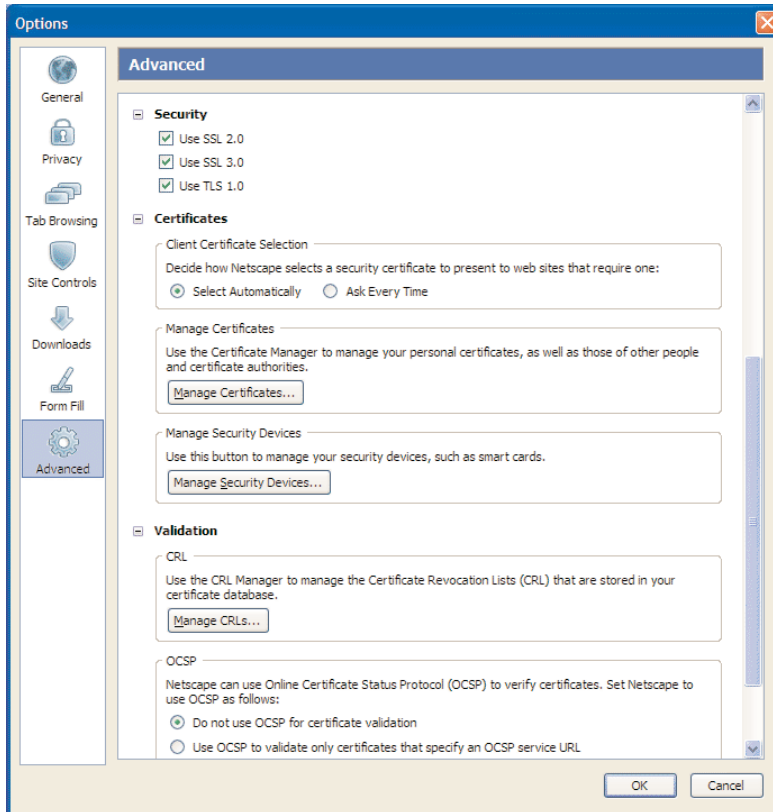


Figure 2-14 shows the Netscape Navigator SSL/TLS configuration interface (which is accessible from the Advanced page in the Navigator Options dialog box).

Figure 2-14
Netscape Navigator SSL/TLS configuration interface



Comparing SSL to Other Web Client Authentication Protocols

In this section, we compare SSL client certificate-based authentication to other Web (or HTTP) authentication protocols: basic authentication and digest authentication. First, we explain how an HTTP authentication sequence works.

Remember from the previous sections that client certificate-based authentication is optional in an SSL setup. Client certificate-based authentication offers strong authentication of the Web clients to the Web server, but it also requires the deployment of SSL client certificates to all Web clients.

HTTP Authentication

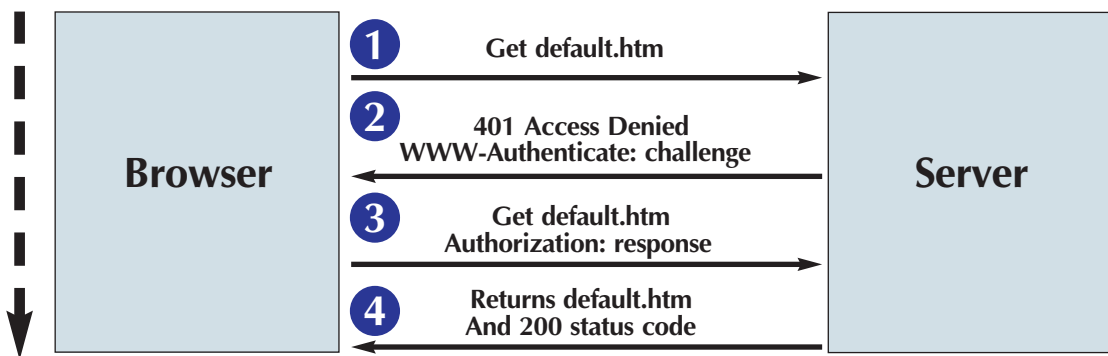
The HTTP protocol specification includes a set of specific HTTP headers to deal with authentication in a Web environment. The HTTP specification also defines Web authentication methods: basic authentication, which is a part of the HTTP version 1.0 specification, and digest authentication, which is a part of the HTTP version 1.1 specification. Now let's look in more detail at HTTP authentication and each of these authentication protocols.

Figure 2-15 shows the different messages that are exchanged between a Web browser and server during an HTTP-based authentication exchange:

1. The browser requests data from the Web server using an HTTP GET verb.
2. If the Web server requires the client to authenticate himself or herself, it sends an HTTP 401 error message back to the browser, along with
 - A list of the authentication schemes it supports
 - A challenge. Not all Web authentication protocols use this challenge. When the browser is using digest authentication, for example, it uses the challenge to calculate the authentication reply that is sent back to the server. The challenge is sent as one or more WWW-Authenticate headers in the server's response.
3. The browser chooses one of the Web-authentication protocols that it supports and constructs an authentication response. The response is based on the user's credentials and—depending upon the authentication protocol used—also on the challenge that was sent by the server. To obtain the user's credentials, the browser will prompt the user. The response is sent back to the server as part of an authorization header.
4. The server authenticates the data it got from the browser and, assuming everything is okay, sends the response and the requested resource back to the user. The response message includes an HTTP 200 status code, which is a “no errors” message.

Figure 2-15

Typical HTTP authentication exchange



Note that step 2 of this authentication exchange begins with “*If* the Web server requires the client to authenticate himself or herself.” Authentication is indeed an option. Many Web sites do not care about the identity of a user. Good examples of Web sites for which user identity is not an issue are those that contain public information that should be available to everyone. For such sites, authentication is just overhead that slows down the information exchange. That is why most Web servers (including IIS) also provide an anonymous access method. If a Web site is configured for anonymous access, anyone—even those users who do not provide credentials—is allowed to access the site.

Basic Authentication

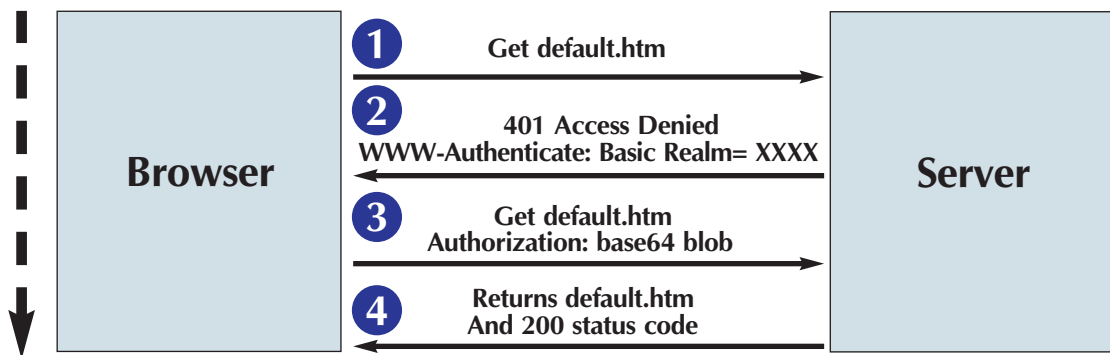
The basic authentication protocol is part of the HTTP/1.0 protocol specification, which means basic authentication can work with any browser type. Basic authentication provides a simple mechanism to transmit user credentials (a user ID and password) to a Web server.

Figure 2-16 shows the different messages that are exchanged between a Web browser and the server during a basic authentication exchange:

1. The browser requests data from the server by using an HTTP GET verb.
2. If the Web server requires the client to authenticate himself or herself, it sends an HTTP 401 error message back to the browser, along with
 - A list of the authentication schemes the server supports, this time including the “basic” verb
 - The name of the basic authentication realm. (We explain the concept of a realm shortly.)
3. Before it replies to the challenge, the browser prompts the user for its username and password. This credential prompt will not occur if the user has previously cached his or her credentials in the browser credential cache, or if the URL that was used to access the Web site contains the username and password. The browser then responds to the authentication request by base64 encoding its username, password, and the authentication realm, and then sending the encoded blob back to the Web server as an authorization host header.
4. The server authenticates the browser’s response and, if everything is okay, sends the response, and the requested resource, back to the user. The response message includes an HTTP 200 status code.

Figure 2-16

Basic authentication exchange



You can shortcut the basic authentication exchange by including the user credentials in the URL that is passed to the Web server. To do this, use the following URL format: `http://username:password@yourwebsite.com`. By default, IIS gives a user three basic authentication attempts.

You should be aware of a potential security hole when you take this approach. The credential information sent between a Web browser and server when you use basic authentication is not secured. The credentials are just base64 encoded, which means they are relatively easy to decode. To demonstrate, base64 encodes messages by storing every three 8-bit characters as four 6-bit

characters. The easiest way to decode base64 is to use one of the online base64-decoder tools such as the tool at <http://base64-encoder-online.waraxe.us/>. Try, for example, to decode the following basic authentication string at the above URL:

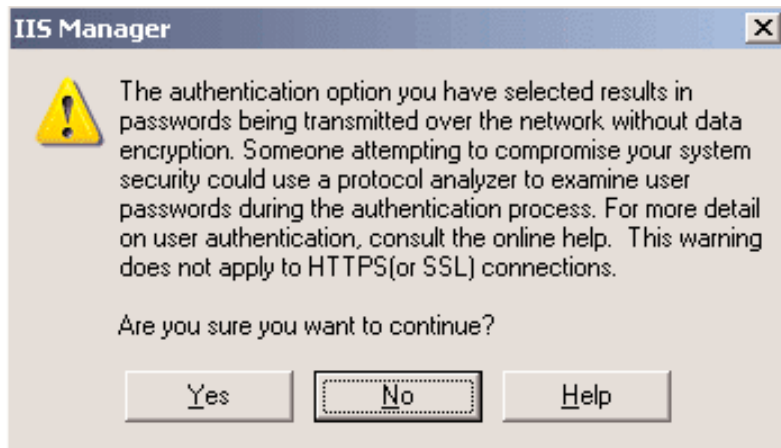
```
ZG9tYWluXHVzZXJlbnBhc3N3b3Jk
```

What is the result of the decoding? Just by copying this string into the box, you should receive the decoded results, `domain\username;password`.

Because decoding a base64 encoding is relatively easy, securing the HTTP traffic using the SSL/TLS protocols is advisable. When you enable basic authentication on a Web resource, IIS warns you about this potential security hole (as Figure 2-17 shows).

Figure 2-17

Basic authentication IIS warning



Digest Authentication

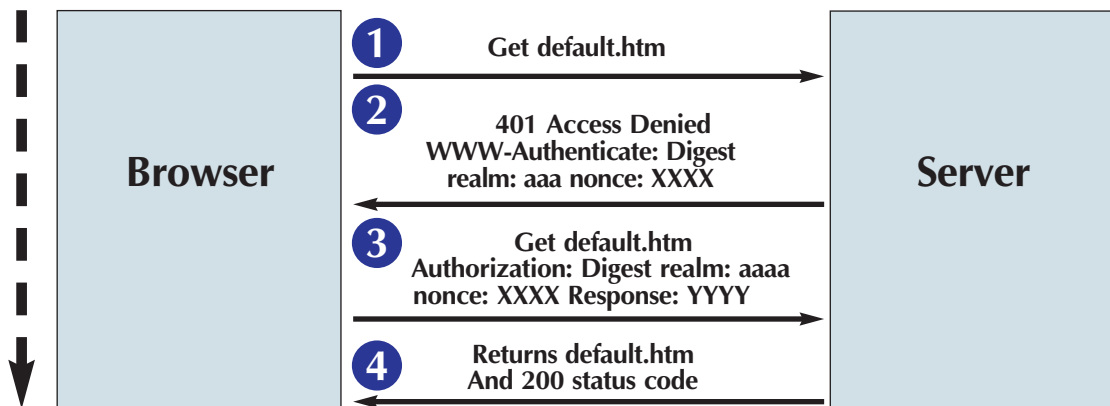
Digest authentication was originally specified as part of the HTTP 1.0 protocol specification; an enhanced version is defined in the HTTP 1.1 protocol specification. Both versions are defined in RFC 2617. Digest authentication uses a challenge-response-based authentication method. A key advantage of digest authentication is that, unlike basic authentication, it does not transmit the user credentials in the clear over the network and thus does not require the use of the SSL or TLS protocols.

HTTP 1.1 and digest authentication are not yet supported by all browser and Web server types and versions. On the Microsoft side, digest authentication is supported by Internet Explorer 5.0 and IIS 5.0 and later versions.

Figure 2-18 shows how digest authentication works:

1. The user's browser requests access to the default.htm Web page.
2. The Web site replies with a 401 access-denied message. The server's reply also tells the browser that the Web page requires digest authentication (the WWW-authenticate header contains the word "digest") and lists the digest authentication realm to which the Web page

- belongs (in the example, realm “aaaa”). Most importantly, the reply also contains a digest challenge, which is usually referred to as the *nonce* (in the example, “XXXX”).
- Before it replies to the challenge, the browser prompts the user for its username and password. The browser hashes the challenge using the user’s password. The outcome of this hash (in the example, response “YYYY”), together with the nonce (in the example, “xxxx”) and the realm (in the example, “aaaa”), are then sent back to the Web server.
 - The Web server validates the response it gets from the browser by calculating the same hash and comparing it with what it got back from the browser. The Web server can calculate the hash because it also has access to the user’s password (which is stored in AD). If everything is okay, the Web server sends the response, together with the requested resource, back to the user. The response message includes an HTTP 200 status code.

Figure 2-18*Digest authentication exchange*

Advanced digest authentication is an enhanced version of the digest authentication protocol. The key advantage of advanced digest authentication is that it does not require clear-text storage of users’ passwords (even though it requires the use of an AD account). On the Microsoft side, advanced digest authentication is available for Web authentication only under the following conditions: IIS 6.0 is running on a Microsoft Windows Server 2003 box, the browser used on the client side is at least Internet Explorer 5.0 or later, the user account used for authentication is defined in a Windows Server 2003 domain that is at functionality level 2 (this means that the domain contains only Windows Server 2003 domain controllers), and the UseDigestSSP metabase property is set to 1 (true). If any one of these conditions is not met, simple digest authentication will be used.

Web Client Authentication Protocol Comparison

Table 2-2 compares the HTTP authentication protocols discussed above (basic authentication and digest authentication) to SSL client certificate-based authentication.

Table 2-2 HTTP authentication method comparison

	Basic Authentication	Digest Authentication	SSL Client Certificate-Based Authentication
Protocol based on open standard	Yes	Yes	Yes
Browser support	Internet Explorer Netscape Navigator Mozilla Firefox	Internet Explorer	Internet Explorer Netscape Navigator Mozilla Firefox
Communication security strength	Weak—base64 encoded, requires SSL	Stronger—because based on a challenge-response mechanism	Strong—because based on asymmetric cryptographic mechanism
Requires SSL	Yes	No	Yes
Supports authentication through firewalls and proxies	Yes	Yes	Yes
Authentication strength	Low—because based on the use of a user ID password	Low—because based on the use of a user ID password	High—because based on the use of private keys and certificates

Conclusion

In this chapter, we have illustrated the process of setting up SSL/TLS to secure Web communications between a Web server and a Web browser. Even though support for SSL/TLS is a common feature for today's Web servers and browsers, setting it up correctly is not always that trivial. We have highlighted some of the hidden traps you should pay special attention to when you configure SSL/TLS for secure Web communications. We also have illustrated the superior features of SSL certificate-based authentication when it comes to authenticating clients in Web environments.

In chapter 3, we will examine some advanced topics you might encounter when you use SSL to secure Web communications. We will look at how to optimize SSL server-side performance and how to deal with load balancing and firewalls. We will also look in greater detail at the SSL certificate-validation process.