Realtime publishers

The Shortcut Guide™ To

Securing Your Exchange Server and Unified Communications Infrastructure Using SSL

sponsored by

GeoTrust®

Don Jones

Introduction to Realtime Publishers

by Don Jones, Series Editor

For several years now, Realtime has produced dozens and dozens of high-quality books that just happen to be delivered in electronic format—at no cost to you, the reader. We've made this unique publishing model work through the generous support and cooperation of our sponsors, who agree to bear each book's production expenses for the benefit of our readers.

Although we've always offered our publications to you for free, don't think for a moment that quality is anything less than our top priority. My job is to make sure that our books are as good as—and in most cases better than—any printed book that would cost you $40 or more. Our electronic publishing model offers several advantages over printed books: You receive chapters literally as fast as our authors produce them (hence the "realtime" aspect of our model), and we can update chapters to reflect the latest changes in technology.

I want to point out that our books are by no means paid advertisements or white papers. We're an independent publishing company, and an important aspect of my job is to make sure that our authors are free to voice their expertise and opinions without reservation or restriction. We maintain complete editorial control of our publications, and I'm proud that we've produced so many quality books over the past years.

I want to extend an invitation to visit us at http://nexus.realtimepublishers.com, especially if you've received this publication from a friend or colleague. We have a wide variety of additional books on a range of topics, and you're sure to find something that's of interest to you—and it won't cost you a thing. We hope you'll continue to come to Realtime for your educational needs far into the future.

Until then, enjoy.

Don Jones

Copyright Statement

Chapter 1: A SAN Certificate Crash Course

SAN—or *Subject Alternative Name*—certificates have become increasingly popular in the past few years. Sometimes called *wildcard certificates* (although that term has a distinct meaning that we'll explore), these special certificates can secure multiple hosts in your organization with a single certificate purchase. In this guide, we'll examine what SAN certificates are, how they work, and specific ways in which they differ from ordinary certificates. We'll look at pros and cons for SAN certificates, and look at ways to mitigate any downsides. We'll also explore how to deploy them in a variety of ways throughout your organization, providing you with better security and potentially much less management overhead.

Digital Certificates

Digital certificates come in a number of flavors, or classes. These classes align to specific use cases, such as securing email, securing communications channels with SSL/TLS, signing software packages, and so forth. The different classes of digital certificates are all more or less identical from a physical point of view; the real difference tends to lie in their price, which is driven by the amount of verification that goes into the certificate-issuing process. In this guide, we'll be focusing on Class 2 certificates that are used to authenticate and encrypt communications channels across networks, typically using Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols.

Certificates consist primarily of two asymmetric encryption keys. *Asymmetric* because each key is different, yet they can be used in conjunction with one another for various tasks. The *private key* is one held by the certificate owner and, ideally, never released to anyone else. The *public key* is made readily available to anyone who asks for it. Simply put, anything encrypted with one key can only be decrypted by the other key. That behavior enables two distinct uses for certificates:

- In *digital signing,* a signature is encrypted with the private key. When transmitted to a recipient, the recipient attempts to decrypt the signature using the corresponding public key. If decryption is successful, the recipient is assured of the sender's identity because only someone holding that sender's private key could have created that signature. Figure 1.1 illustrates this use.

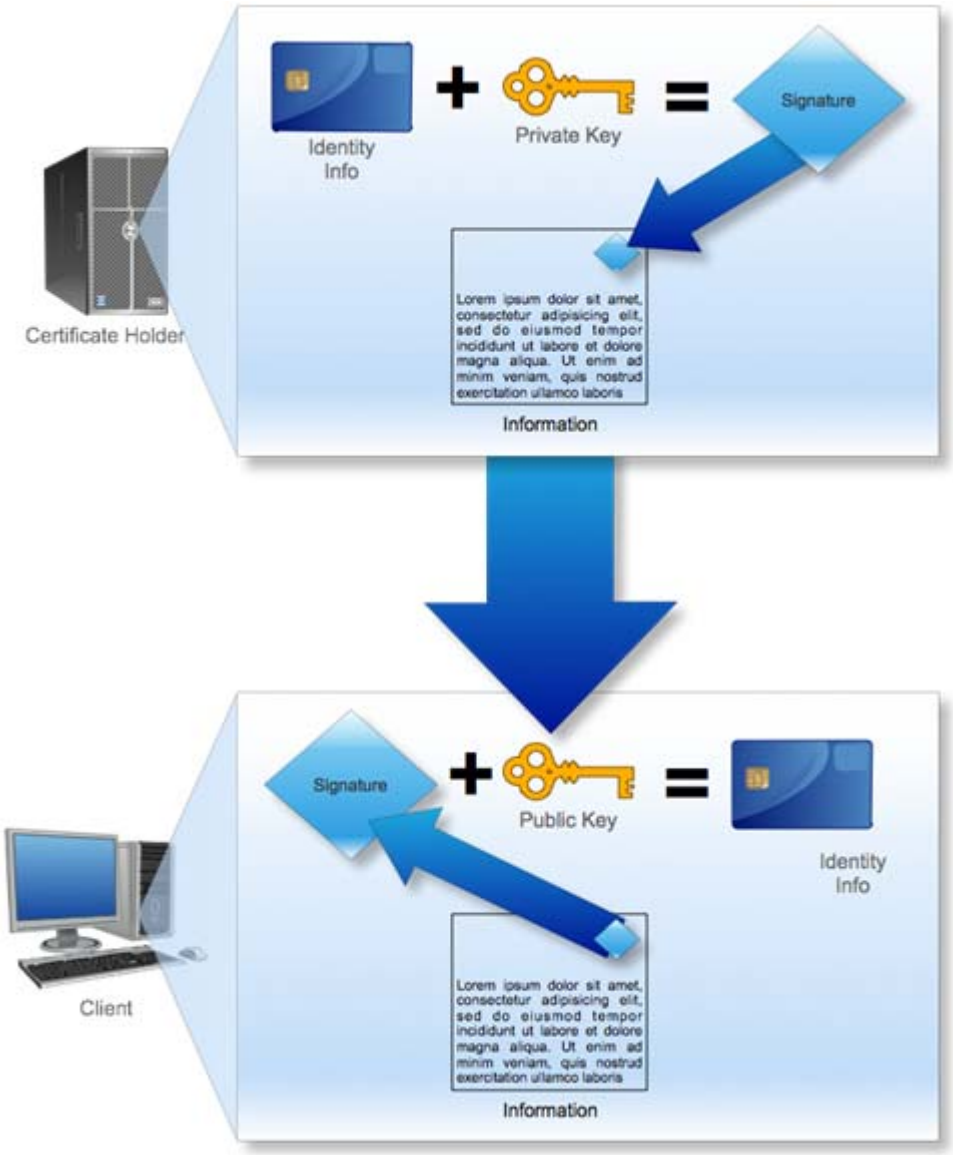

Figure 1.1: Digital signatures.

- In *encryption,* the recipient's public key is used to encrypt data. The recipient then uses their private key to decrypt the data. The recipient is assured that no one else could have seen the decrypted data because only their private key was capable of retrieving it. Figure 1.2 illustrates this use.

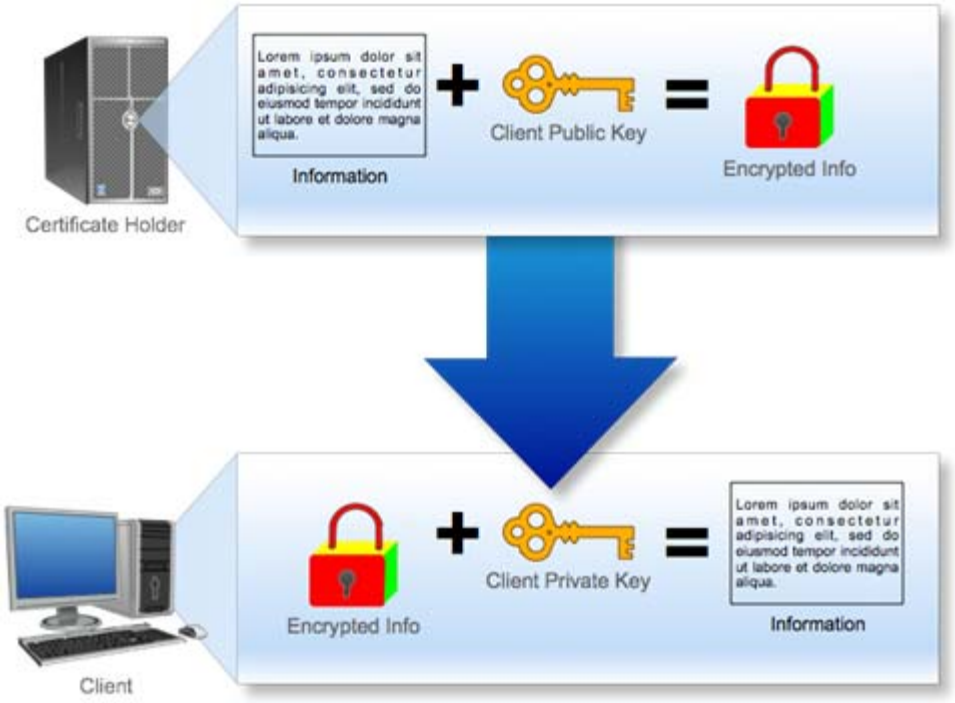

Figure 1.2: Encryption.

Actually obtaining keys can be a complex process. In an SSL transaction, an end user generally connects to a server and requests the server's public key. Of course, the endpoint (which might be the user's Web browser, for example) can't simply take the server's word for it—it has to verify the validity of the key, and verify that the key does in fact belong to the organization the server claims to represent. One way to do that is to connect to the Certification Authority (CA) that issued the key, and ask the CA to verify the key's validity. Specific protocols allow exactly that to happen, although it can slow down the initial connection to the server while the verification takes place.

Trust chains are another technique that an endpoint can use to verify a server's key, and these do not require external communications. Essentially, endpoints are preconfigured with a list of top-level, or *root,* CAs that they trust. Figure 1.3 shows an operating system (OS) configuration with trusted root CAs. This configuration includes the public key for those CAs, meaning the endpoint has ready access to that public key.

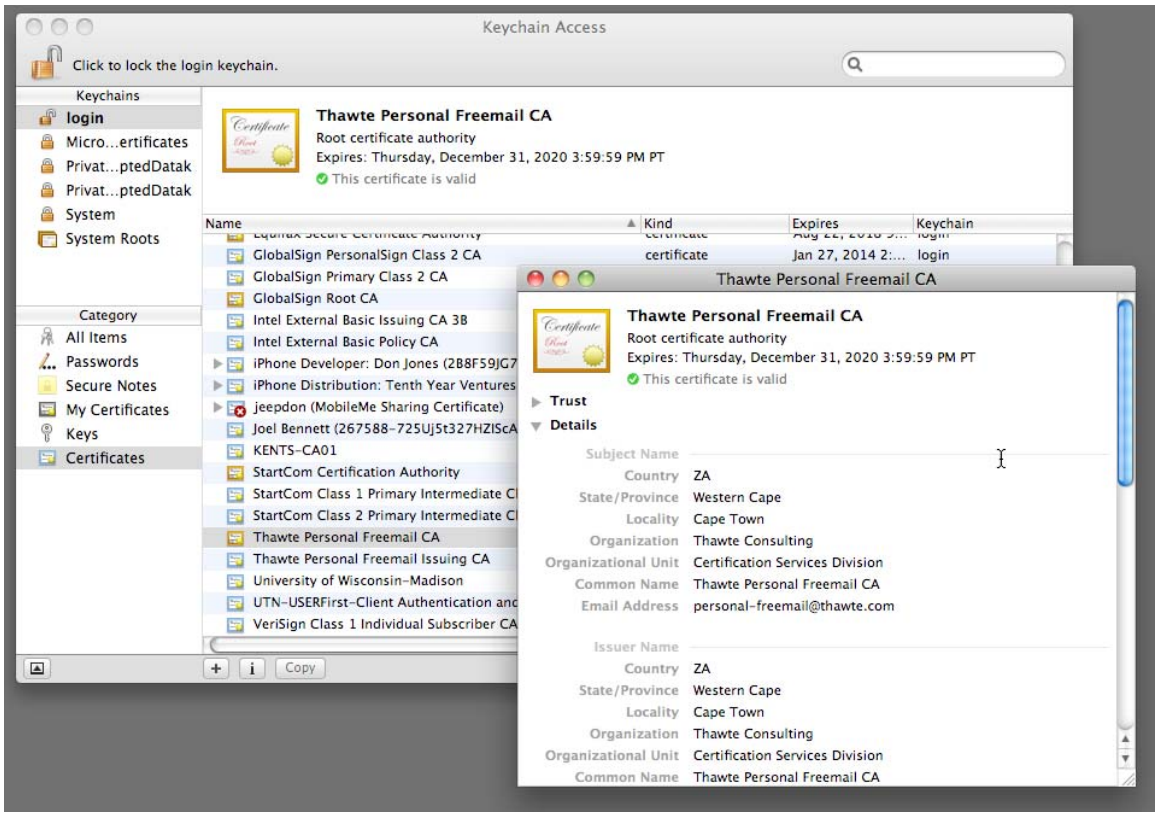

Figure 1.3: Examining a CA's root certificate.

When a CA issues a certificate for a server, part of what's included is the trust chain back to the CA. In other words, the CA creates the verification information that an endpoint would need, encrypts it with the CA's private key, and includes that information in the certificate. The actual certificate holder can't access or modify that information because they would have no means of re-encrypting it. Endpoints can decrypt the information to verify the certificate holder's identity, and make sure that it matches whoever the server claims to represent. Figure 1.4 illustrates the trust chain.

It's possible for certificates to become *compromised,* which typically occurs for one of two reasons:

- The CA realizes that they issued a certificate in error—perhaps improperly identifying a certificate holder.
- The certificate holder loses control of their private key, believing that it may be in the hands of someone else—who could use it to impersonate the certificate holder.

In these cases, the certificate is *revoked* by the CA. Originally, a list of revoked certificates was made available as a Certificate Revocation List (CRL), which endpoints could download and check prior to accepting a certificate. The size of these lists quickly grew to the point where constantly downloading them became completely impractical. The Online Certificate Status Protocol (OCSP) was designed to give endpoints a fast way to check in with a CA regarding the revocation status of a certificate. Endpoints can cache the OCSP response for a short period of time (or permanently, for revoked certificates) so that they don't have to continually re-check for the same certificate.

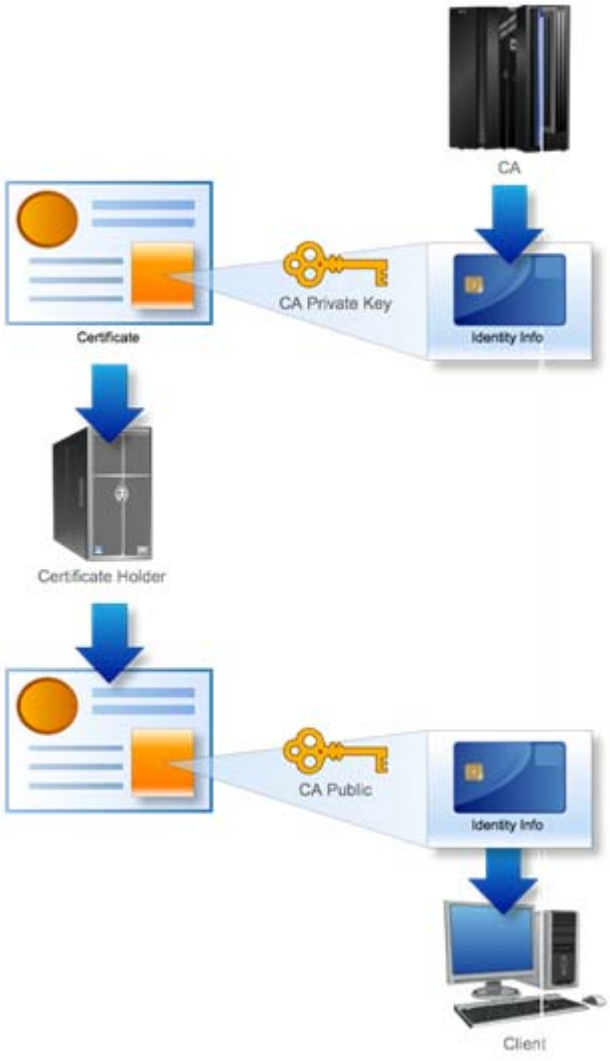

Figure 1.4: Trust chain.

Figure 1.5 shows an OS OCSP configuration dialog box, enabling the feature. OCSP does add a short delay to the initial connection with a server.

Figure 1.5: Enabling OCSP on a client computer.

Certificates also expire, and endpoints aren't supposed to accept them once they reach their expiration date. Once expired, certificates are no longer listed as "revoked" (even if they were) because an expired certificate is no good. That's a key reason for endpoints to observe the expiration date: You can't tell whether an expired certificate was revoked earlier in its life. You have to assume that all expired certificates are compromised and revoked. A problem is that most modern OSs don't make this clear to users. They'll display a warning dialog box, such as the one in Figure 1.6, but they allow users to override the warning. Users are rarely cognizant of the risks involved in doing so.

Figure 1.6: Expired certificate warning.

In a typical transaction involving a server and a client, only the server has a digital certificate. That means the client can verify the server's identity, but the server cannot verify the client. This is *non-mutual,* or one-way, authentication. However, the server can create a temporary *session certificate* and send it to the client. That gives the client a public and private key, and the server knows the public key (because the server created it). That session certificate is used by the server to encrypt information sent to the client, and the client uses the private key portion of the session certificate to decrypt that same information. This all happens automatically and behind the scenes as part of protocols such as SSL and TLS.

SSL vs. TLS

SSL was the original protocol used to authenticate and encrypt communications channels on the Internet, primarily with protocols like HTTPS. Today, *most* "SSL" transactions actually use a successor protocol, TLS, although we still tend to use the term "SSL" in conversation. In terms of the functionality they deliver, the two terms are interchangeable, although there are obviously significant technical differences between them.

The last version of SSL was introduced in 1996, and the first version of TLS in 1999. The current version of TLS is 1.2 (also referred to as SSL 3.3), and it was introduced in 1998. The major differences between protocol versions primarily relate to the encryption protocols used. Today, TLS/SSL is used to secure any number of Internet protocols, including HTTPS, certain versions of secure FTP, IMAP, POP3, SMTP, and more.

What Makes a SAN Certificate Different?

A SAN certificate *is* a digital certificate. It works exactly as described earlier, can be revoked, supports signatures and encryption, and so forth. The difference lies in what it contains.

A normal certificate issued for a server is variously called a "Web server certificate" (although it can be used in applications far broader than Web servers), an "SSL certificate" (although it's more commonly used with the TLS protocol), or a "Class 2 certificate." Commercial CAs issuing these certificates typically go through specific procedures designed to verify the identity of the organization to which the certificate is sold. For example, a CA may check public records, require a copy of a business' articles of incorporation, check with commercial credit-reporting agencies, and so forth. The goal of this process is to ensure that a certificate issued to "XYZ Corporation" is *in fact* going to representatives of "XYZ Corporation."

A Matter of Trust

Figure 1.3 shows a list of certificates installed on a computer, including several root CA certificates. Most organizations can exercise centralized control over the root CA certificates installed on their computers. By having a root CA's certificate installed, you're essentially saying that you *trust the CA to do a good job of verifying the identities of the people to whom they issue certificates.* The CA is attesting to the identity of their certificate customers, and you trust them to be thorough in verifying those customers' identities on your behalf.

If a CA does a poor job of this, you could find yourself at the wrong end of a deception. If "Hackers R Us" was able to obtain a certificate claiming that they were "XYZ Corporation," they could *pretend to be* XYZ Corporation—and would have a good chance of pulling it off. You probably wouldn't trust the CA who issued that certificate for much longer.

There can be huge price differences between certificates from commercial CAs, and the price is tied at least partially to the thoroughness with which the CA vets their customers' identities. By obtaining certificates from a broadly-trusted CA, you can ensure that your certificate will be accepted by a wider range of end users.

If someone chooses not to trust a CA, and later encounters a certificate issued by that CA, they'll see a warning like the one in Figure 1.7, which indicates that the certificate being presented was issued by a non-trusted CA. The user is left to decide whether they want to go ahead and trust the certificate—and unfortunately they usually have little information from which to make an informed decision.

Figure 1.7: Receiving a certificate issued by an untrusted CA.

That same verification process applies to SAN certificates. However, although a normal SSL certificate (I'll just use that term from here on) is *issued* to an entire organization, it only *names* a single host. Thus, a certificate that names "www.company.com" can't be used to secure "commerce.company.com" because the host name—"commerce"—isn't named on the certificate. Browsers will display a warning like the one in Figure 1.8 when they're presented with a certificate whose listed name doesn't match the name of the site they accessed.

Figure 1.8: Certificate/site name mismatch warning.

This kind of warning can result even from simple mistakes. For example, a Web browser tries to access "company.com," but the server's certificate lists "www.company.com." It's the correct server and Web site, but the browser can't determine that, and so the message is somewhat misleading.

The upshot of this is that, originally, companies had to acquire and manage a plethora of certificates: one for each Web site, email server, or whatever other host they wanted to protect. SAN certificates simply added the ability to include one or more *alternative names* within the certificate. In other words, in addition to listing "www.company.com," the certificate might also list "commerce.company.com," "company.com," "mail.company.com," and any other hosts the company wanted to protect. So-called *wildcard certificates* might simply list "*.company.com," making the certificate valid for any host within that domain.

There is a distinction between wildcard certificates and SAN certificates: A certificate can include a wildcard without using the SAN certificate extension. However, in common parlance, people tend to mix up the two names, and the two types do provide similar functionality. Later in this chapter, we'll explore the specific technical differences.

So the advantage of SAN certificates is pretty obvious: You manage one certificate for many, or potentially all, of the hosts in your domain. You save money, you save management overhead, and you help reduce complexity.

The Downside of SAN Certificates

Of course, with every upside, there's a downside, and with SAN certificates, it's one of risk. Keep in mind that the crucial part of any certificate is its private key. The assumption around which all certificates are built is the one that the private key is held *only* by the entity to whom the certificate was issued. If that private key is compromised and held by anyone else, the entire system falls apart.

With a traditional, single-host SSL certificate, a compromised private key is a hassle, but it's not a major one. Let's say somebody breaks into a remote office and manages to swipe a whole server that used an SSL certificate. That means they have the certificate's private key. You call up the CA that issued the key (or, more commonly, use a Web site interface), request that the key be revoked, and you're done. You get a replacement certificate that contains a new private key, and you go about your business. All of your other servers continue operating without impact because they all use different single-host SSL certificates, and so all have their own unique private keys. Compromising one key doesn't compromise any others.

You probably see where this is going. If that stolen Web server had contained a SAN certificate, *every machine using the same certificate* would be compromised—and that might well include every server in your organization. Now you're looking at a major hassle: You only have to revoke and re-issue one certificate, but you have to run around installing it on *every* server that used the old certificate. You want to revoke the compromised certificate quickly, to reduce the chance of the thief impersonating your organization, but the second you revoke that certificate, every server using it will stop being able to use it.

> **Note**
>
> Technically, there's a bit of lag time between revocation and the certificate actually being useless, especially because a lot of clients don't routinely use OCSP or CRLs to verify certificate validity. You should still act swiftly, and assume that a revoked certificate is immediately useless, because clients who *do* use OCSP will find out about the revocation the next time they see your certificate.

This risk isn't impossible to mitigate, however, and companies take differing approaches. For example, you might only use a single SAN certificate for servers that are physically co-located. After all, if a thief gains physical access to one of them, he'd have physical access to all of them, and could compromise them all. Further, if one of them *is* compromised, they're at least co-located, so the certificate replacement process can be done a bit more swiftly. Each distinct location might use its own SAN certificate. You're still juggling multiple certificates that way, but it's far fewer than using single-host certificates. Figure 1.9 illustrates this deployment technique.

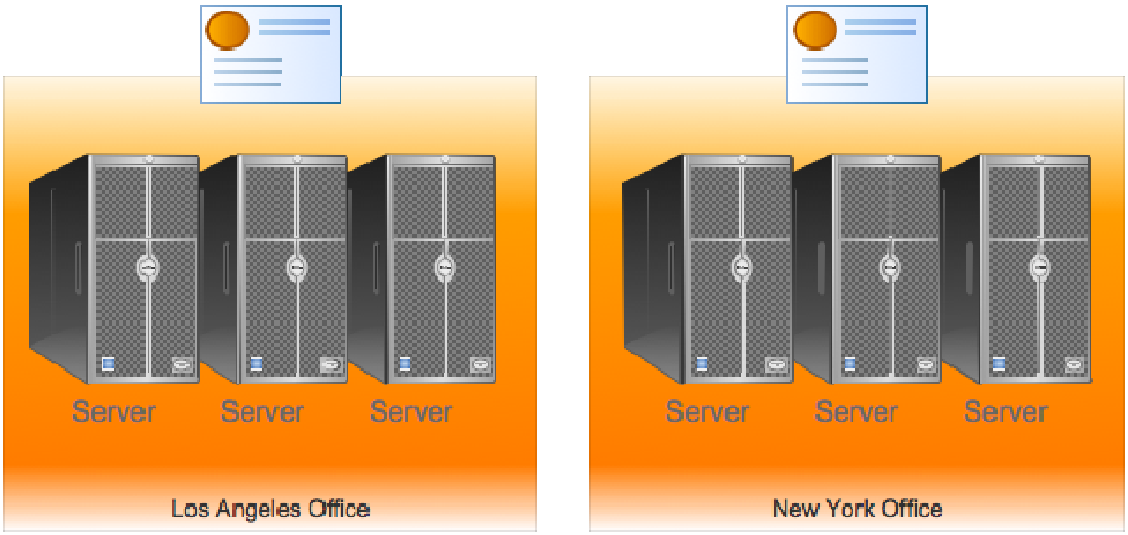

Figure 1.9: Grouping servers by physical proximity.

Another approach is to use a single SAN certificate for all servers that have the same degree of physical protection. For example, all servers in a locked, well-secured data center could share a certificate, on the theory that they're less likely to be compromised than a server that's kept in a broom closet in a remote office. Figure 1.10 shows how this might work.

Figure 1.10: Grouping servers by physical security.

Note that my emphasis is on physical access to servers. That's perhaps the most *easily-visualized* way for a certificate to be compromised. In other words, it's easy to imagine that, if someone could lay hands on the server, they could get the certificate. There are obviously ways in which a server could be *remotely* compromised, without physical access. Gaining remote administrative access to a server—through a remote shell, Remote Desktop connection, or other means—might enable certificate access, too. If that's a concern, you'll simply have to make a decision on the balance between ease of use and ease of replacement when it comes to SAN certificates. Some companies use scripts to automate the certificate-replacement process (it has to be done when a certificate expires, for example, so it's a process you'll undergo periodically even if there's no compromising event). Those companies are confident in their ability to replace a compromised certificate quickly and efficiently, so they're more comfortable using a SAN certificate quite broadly—perhaps on all or most of their servers.

Yet another approach to SAN certificate usage is to employ a different SAN certificate for each *class* of server: Web servers, email servers, and so forth. The theory here is that, if a certificate is compromised, only a single class of service—Web, messaging, and so on—is impacted. Figure 1.11 illustrates this deployment breakdown.

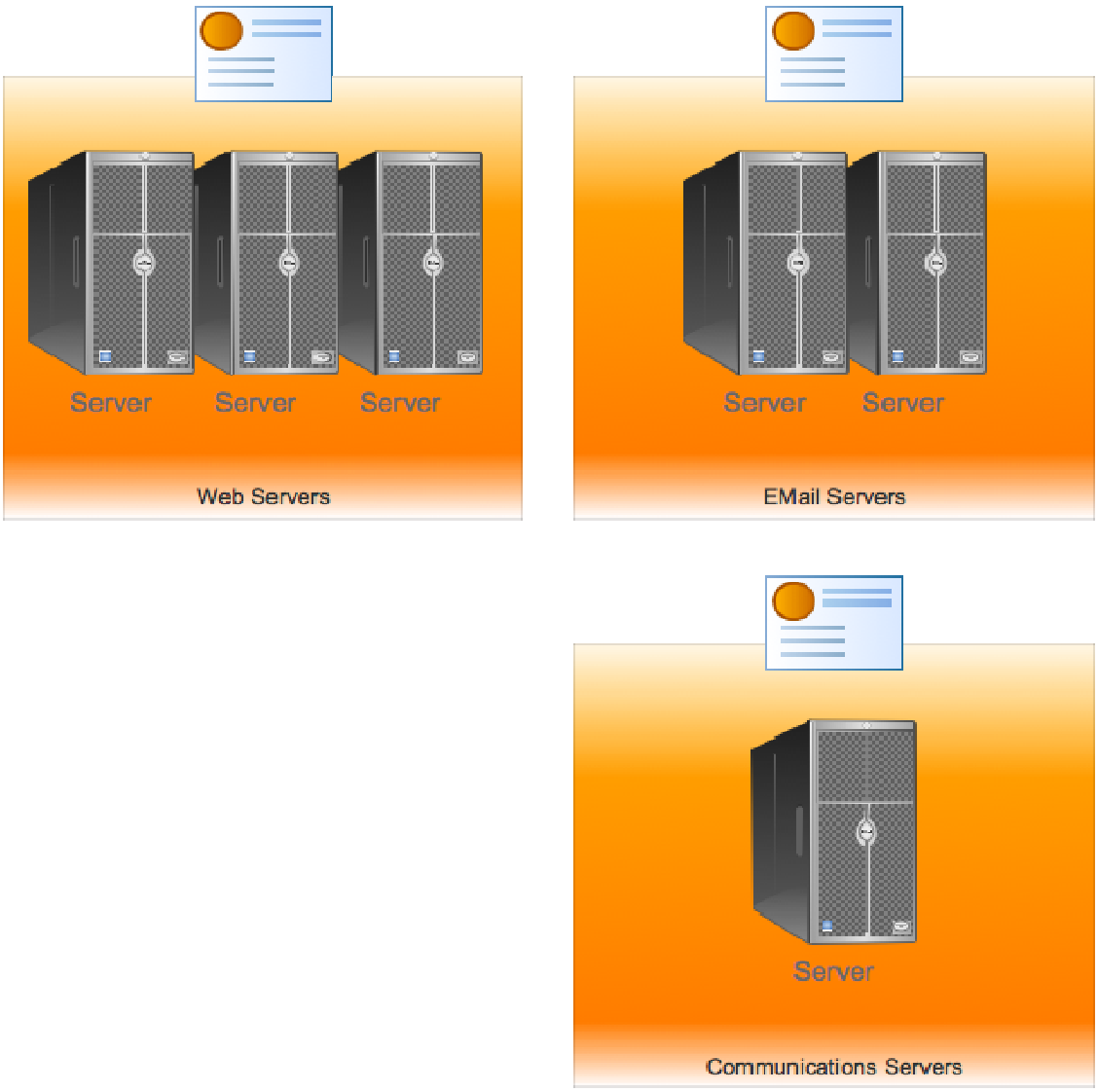

Figure 1.11: Grouping servers by logical function.

The "single point of failure" that a SAN certificate seems to represent is *not* a reason not to use them. You simply need to be aware of the risks, and take appropriate steps to mitigate and manage those risks.

SAN Certificate Technical Details

As already described, a SAN certificate is just a "standard" certificate with an extra field: Subject Alternative Name. As Figure 1.12 shows, all certificates includes a *Subject* field, which lists the host name that the certificate protects. In Figure 1.12, that host name is E2K7S04. A SAN certificate's Subject Alternative Name field simply lists additional host names—E2K7S04 and others, in this example.

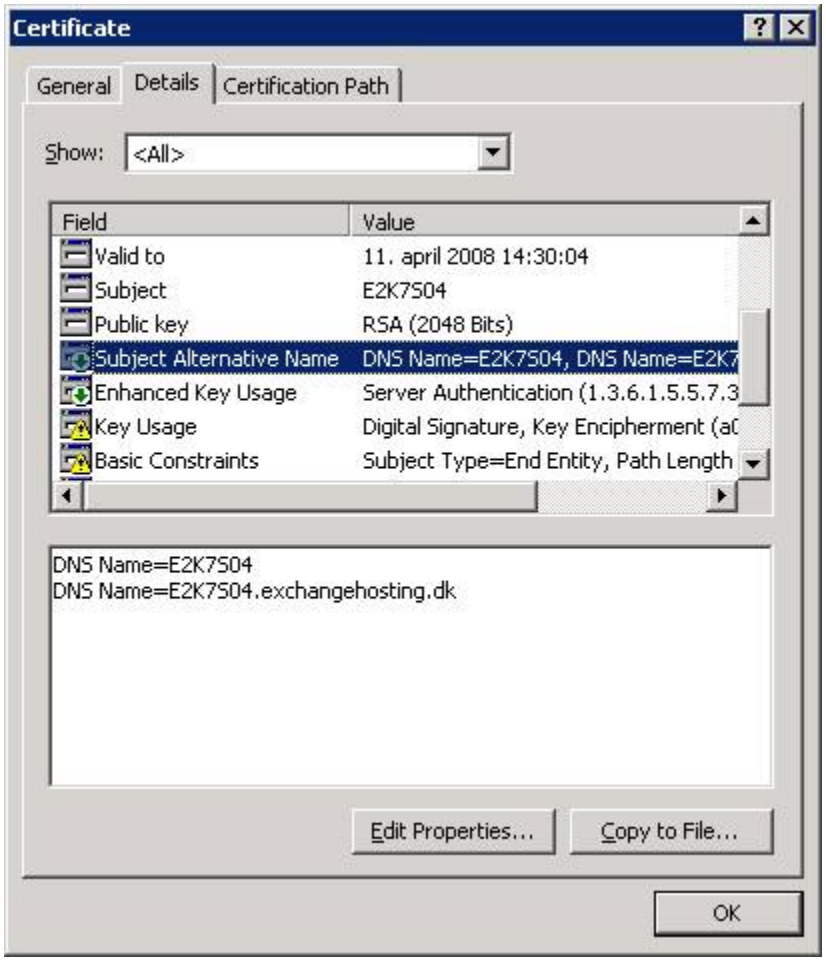

Figure 1.12: SAN certificate details.

The client software to which this certificate is presented must recognize and understand the existence and purpose of this additional field. All modern Web browsers do so, and the code libraries typically used by programmers to implement SSL/TLS communications all do so. It's rare to find a piece of software that can't work with a SAN certificate. Older mobile devices are perhaps the most likely to have problems with a SAN certificate, meaning they'll only recognize the certificate's use with the host name listed in the Subject field, and won't pay attention to the SAN field.

Consider the following specific compatibility information:

- Internet Explorer, Firefox, Opera, and Safari versions from 2003 onward all support SAN certificates. Internet Explorer has done so since 1998.
- Windows Mobile 5 and later support SAN certificates, but Windows Mobile 5 does not recognize wildcards (*.company.com). Many CAs, however, permit the use of wildcards in the Subject field and list specific hosts in the SAN field, which helps mitigate this incompatibility. Windows Mobile 6 addressed this problem.
- Palm Treo devices running PalmOS do not support SAN.
- Symbian OS supports SAN from version 9.2 and later.
- Other modern mobile devices running Apple iOS, Google Android, Windows Phone 7, and so on all support SAN and wildcards.

Wildcard certificates, as implied, typically include the wildcard—such as "*.company.com" in the Subject field. With supported clients, this has an effect similar to that of a SAN certificate, although it's technically still a single-server certificate. It's common to see certificates that use a wildcard in the Subject field, while listing specific hosts in the SAN field. Technically, if a certificate has a SAN field, the client is supposed to ignore the Subject field and only look for name matches in the SAN field.

Coming Up

In the next chapters, we'll look at deployment specifics for SAN certificates, including Web browsers, messaging servers, Unified Communications servers, and more. We'll examine best practices around SAN usage and management, and cover related security best practices (after all, security doesn't start and stop with a SAN certificate).

Download Additional Books from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this book to be informative, we encourage you to download more of our industry-leading technology books and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.