Realtime publishers

The Shortcut Guide™ To

Certificates in the Enterprise

sponsored by

thawte™ it's a trust thing™

Don Jones

Copyright Statement

Chapter 2: The Many Faces of Digital Certificates

We're all accustomed to digital certificates when they're used to identify and secure the connection to a Web server—"SSL certificates," although that name is technically a bit inaccurate, as we'll discuss in a moment. But certificates' usefulness goes *far* beyond Web servers and the HTTPS protocol. In fact, in many ways, certificates have always held the solution for some of IT's most irritating and difficult challenges. Want to eliminate malware? Try using certificates. Want to be compliant with some piece of legislation? Certificates can help. Trying to find a way to authenticate business partners to an extranet? Yes, certificates offer a solution.

Certificates have been around for a long time, and you might not even be aware of the ways in which they already help you solve tricky problems. In this chapter, we'll start with some of the most common uses—ones that you're probably already familiar with—just so we can be comprehensive. We'll then go beyond those common uses and start talking about some of certificates' lesser-known capabilities in the enterprise.

For E-Commerce

The SSL digital certificate is probably one of the most commonly issued types of certificates in the universe. We rely on them to help encrypt the connection between our Web browser and a Web server so that sensitive information such as credit card numbers can't easily be intercepted and stolen by an electronic eavesdropper.

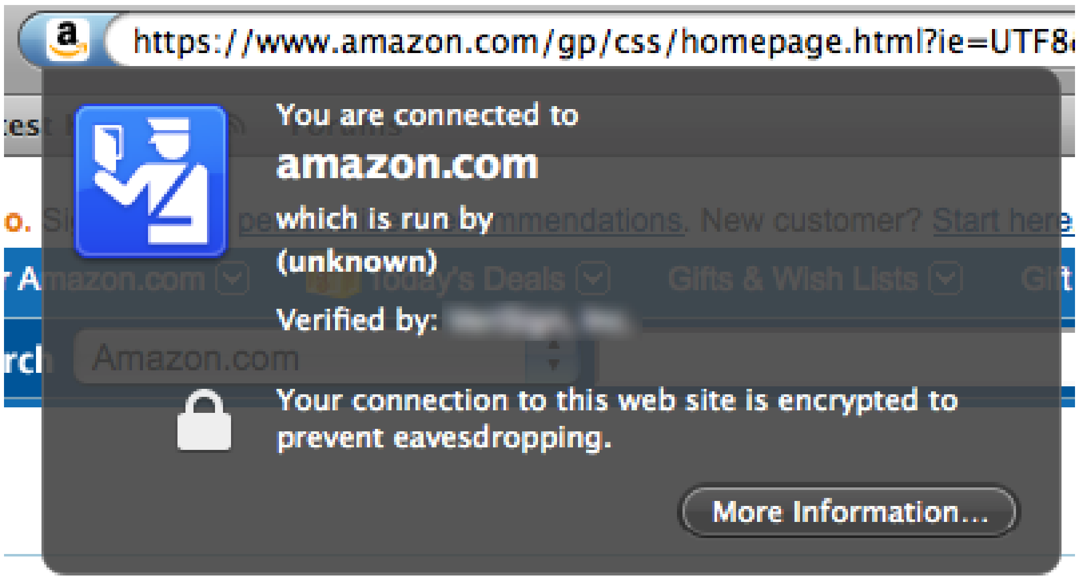
Note

Technically, *SSL* is an inaccurate term. The Transport Layer Security (TLS) protocol replaced Secure Sockets Layer a long time ago. But everyone was already so used to calling it "SSL" that the name stuck. In addition to "SSL certificate," you might also see the term "Web server certificate" or "Web server digital ID." They're all the same thing: Certificates that, in part, enable the TLS protocol to provide data encryption.

Funny thing is that we tend to *only* think of Web server certificates for encryption—but encryption isn't even their first purpose, and encryption isn't even a *mandatory* part of the HTTPS protocol.

Think about it: The idea is to avoid having your credit card number or other sensitive information disclosed to some unintended party, right? So encryption is useless if the person at the other end of the encrypted connection *isn't the intended party!*

As covered in the previous chapter, the *first* purpose of a digital certificate is *identification.* When your browser uses the HTTPS protocol to connect to a Web site, the browser's first concern is validating the *identity* of the people who run that Web site. More modern Web browsers do a better job of helping convey that identify-verification, as Figure 2.1 shows.


https://www.amazon.com/gp/css/homepage.html?ie=UTF8

You are connected to
amazon.com
which is run by
(unknown)
Verified by:

Your connection to this web site is encrypted to prevent eavesdropping.

Figure 2.1: Browsers now do a better job of displaying Web identities in an HTTPS connection

In this example, I can clearly see that I am, in fact, connected to Amazon.com, and that the optional encryption has been enabled. It's important to understand that merely having an HTTPS connection does *not* ensure encryption! What's more, the mere presence of HTTPS doesn't ensure that the Web site is the one you think it is—although most Web browsers will display a warning if you connect to a site and it's trying to use someone else's certificate.

All this identity checking, however, is only useful when you trust the person or company that actually issued the certificate. In the screenshot, the "Verified by" information tells me who issued the certificate, and I need to decide whether I trust that company to have done a good job of verifying Amazon.com's actual identity before they issued a certificate attesting to that identity. Most browsers now offer something like the "More Information" button shown in Figure 2.1, which I could click to review the certificate in detail, see who issued it, and potentially even see contact information for the issuer, in case I needed to further verify the identity of the Web site.

For Email

If Web server certificates aren't the most commonly issued kind of certificate, then email certificates almost certainly are. They're the other most well-known form of certificate, and unlike Web server certificates, more users are actually aware of these certificates' role in identity. Email certificates actually have two distinct uses: signatures and encryption.

A signature—or *digital signature,* if you prefer—is a way for someone to send an email and attach their identity to it. It's also a way for them to ensure that the message isn't tampered with in transit. Essentially, the sender's email client creates a *hash* from the email contents; the hash is sort of like a checksum in that it's mathematically derived from the email's contents using a well-known mathematical calculation. *Anyone* should be able to take the same email contents and derive the same hash. Certificates don't come into play when making the hash; instead, they're used to encrypt the hash. So the final, signed email contains the clear-text email, the encrypted hash, and a copy of the sender's public key (remember, that key is one-half of the sender's digital certificate). The hash was encrypted using the sender's private key—something only they have access to.

When someone receives that email, their email client does several things:

- It grabs the public key and verifies that it was issued by a trusted Certification Authority (CA). Because the key comes as part of a complete certificate, the email client can see the entire chain of issuing CAs and verify the validity of the public key—there's no need to go to one of the CAs and actually retrieve the public key.
- It calculates its own hash of the email message.
- It uses the public key to decrypt the included hash and compares it with its own hash. If the two differ, the message was tampered with and the signature is said to be *broken.*

So the signature is about verifying the identity of the sender and the validity of the email message. It's not about protecting the privacy of the message—anyone can read that, even if they choose not to verify the signature.

Encryption is another story. The sender needs to first obtain the recipient's public key—one easy way to do so is to have the intended recipient first send a digitally signed email. Most email clients will automatically extract their public key from the signature and store it in the local address book. Once the sender has the recipient's public key, that key is used to encrypt the contents of the message so that only the recipient's private key—which only they have—can decrypt it. It's common for the sender to also digitally sign and encrypted email, but it's not required.

For FTP

Encrypted connections for HTTP have been around for a long time, but protecting the contents of FTP transfers took a bit longer for the industry to standardize. Today, the same certificates used to secure a Web server connection can also be used to secure FTP connections through the use of what's generically called *Secure FTP*. There are actually two distinct flavors of encrypted FTP:

- FTP over SSH, most commonly referred to as "Secure FTP," tunnels a normal, unsecured FTP connection over an encrypted Secure Shell (SSH) connection.
- FTP/SSL, or FTPS, uses the same TLS protocol that HTTPS uses to encrypt transferred data. Common variations include FTPES and AUTH TLS; as with HTTPS, encryption is actually an optional part of the protocol and the first and foremost intention is to identify validation.
- SSH File Transfer Protocol (SFTP) is a part of the SSH protocol suite and is not technically a variant of FTP.
- Secure Copy (SCP) is also its own protocol and is not a variant of FTP. It relies on SSH to provide authentication.

Of these many techniques, only FTPS typically uses digital certificates. SSH uses encryption keys, but these are often less-secure symmetric keys, such as passwords, although some implementations of SSH do allow for the use of digital certificates' asymmetric encryption.

As more and more companies deal with increasingly demanding legislative and industry requirements for security, encrypted file transfer is becoming a must-have part of the infrastructure. FTPS, which utilizes a variant of FTP and relies on asymmetric encryption, is one of the more popular standards. It is supported by the free FileZilla FTP server and client software (http://filezilla-project.org, which also supports SFTP) and its use of digital certificates helps not only encrypt data but also ensure that you're sending data to the person or company that you intended. Remember: Without *identity,* encryption itself is fairly pointless.

For Chat/IM

Just as email was the communications revolution a decade ago, chat and instant messaging (IM) have become today's revolutionary way to communicate electronically in even more real-time conversations. And, just as we might want privacy and identity verification for email, we might well need it for a chat or IM session. After all, if you plan to include sensitive data in your conversation, you might want to ensure that the person on the other end is who you think they are—and that the data won't be subject to eavesdropping and tampering while in-transit.

Secure chat typically utilizes the same type of personal digital certificate that email uses, and like email, requires the use of client software that understands how to use the certificate. Unlike email, however, there are fewer standardized protocols for secure chat, and fewer IM or chat clients have built-in support for certificate-based security. One client that does is Apple's iChat client for Mac computers, although it uses internally generated certificates rather than one from a CA; the downside of this is that you're trusting Apple's built-in software to do a good job of verifying identity. In fact, by not using a CA, iChat's security is primarily good for encryption, not identity: You can be sure that your data isn't being eavesdropped, but you can't be sure of whom you're actually sending it to in the first place.

For Software and for Eliminating Malware

Software is an area where digital certificates truly have an opportunity to improve the lives of IT users and professionals. Digitally signed software can be a key to creating a more secure, reliable, and stable IT environment. Signed software can also help protect the author or publisher's reputation, by making it more difficult for someone to "hijack" the author or publisher's name and use it on poor-quality software.

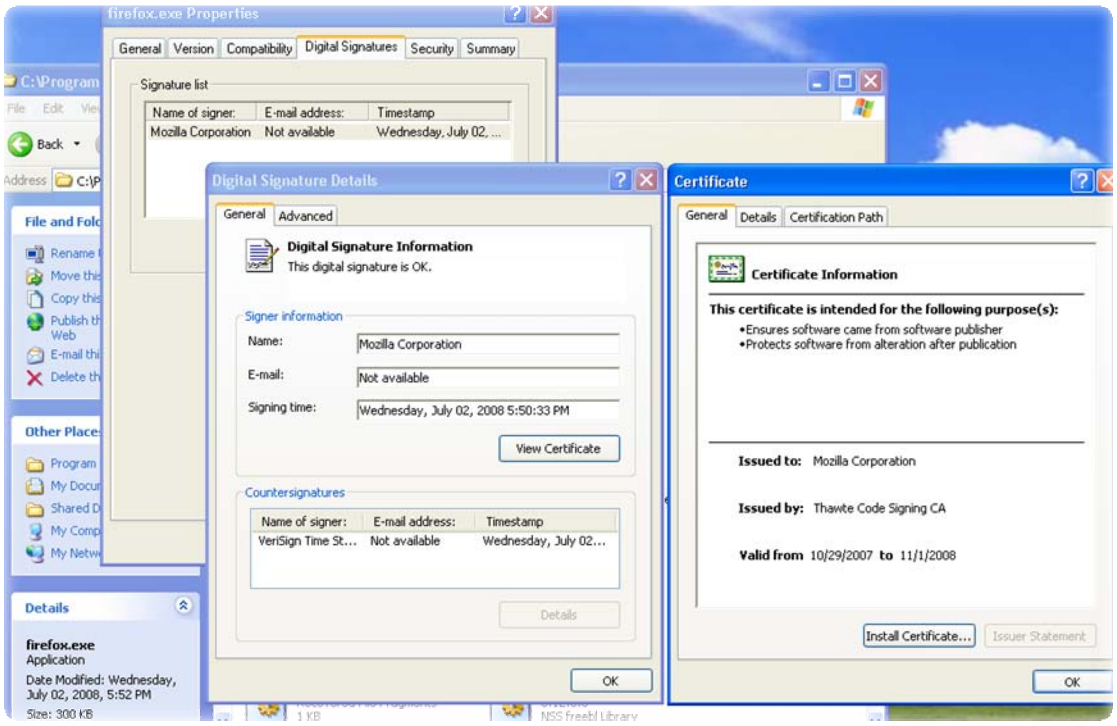
It Starts with a Signature

Software security begins when an application is digitally signed. Because software represents an enormous potential risk—after all, software can potentially wreck your machine—CAs typically require much more extensive identity-verification procedures in order to issue a code-signing certificate. The philosophy goes something like this: You probably wouldn't run software that can from a source known to be unreliable. You might not run software unless it *definitely* comes from a source *known* to be reliable. You'll probably run software if you know it comes from a reliable, trustworthy source.

For example, regardless of what you think of products from Microsoft corporation, you can be pretty sure that their software isn't deliberately intended to wreck your computer—not like, say, a virus. So when you have a piece of software *that is definitely from Microsoft,* and hasn't been tampered with, that software is probably safer to run than a piece of random software downloaded from the Internet that *isn't* from a large, trustworthy company and *may* have been tampered with.

So what you need is a way of knowing two things: who wrote the software and whether it was tampered with after it left their hands. With that information, you can make an informed decision about whether the software is likely to be safe. Code-signing certificates can give you exactly the information you need (more so on Windows computers and less so on UNIX-based computers, where code-signing of common application files is less common).

Take a look at Figure 2.2. Here, I'm using Windows XP to examine the digital signature on the main executable for the Firefox Web browser. I can see that the certificate used to sign this application was issued to the Mozilla Corporation. If I trust the certificate's issuing CA, then I know that the executable was indeed provided by Mozilla. Because the signature is intact, I know that the executable hasn't been tampered with since it left Mozilla's hands. If I trust Mozilla, then I can safely run this application and not expect any maliciousness.



Signature list

Name of signer:	E-mail address:	Timestamp
Mozilla Corporation	Not available	Wednesday, July 02, ...

Digital Signature Information
This digital signature is OK.

Signer information

Name: Mozilla Corporation
E-mail: Not available
Signing time: Wednesday, July 02, 2008 5:50:33 PM

Countersignatures

Name of signer:	E-mail address:	Timestamp
VeriSign Time St...	Not available	Wednesday, July 02...

Certificate Information

This certificate is intended for the following purpose(s):
- Ensures software came from software publisher
- Protects software from alteration after publication

Issued to: Mozilla Corporation

Issued by: Thawte Code Signing CA

Valid from 10/29/2007 to 11/1/2008

firefox.exe
Application
Date Modified: Wednesday, July 02, 2008, 5:52 PM
Size: 300 KB

Figure 2.2: Examining the signature on a digitally signed executable.

That's a lot of "ifs," though, and it really underscores the role of *trust* in digital certificates. I *trust* the certificate issuer to have done a good job of giving this certificate only to the one and only real Mozilla Corporation. I *trust* Mozilla to not try and maliciously wreck my computer. But those issues of trust are human issues that no technology can solve; the certificate gives me a way to intelligently *apply* the trust that I've decided to grant to these entities.

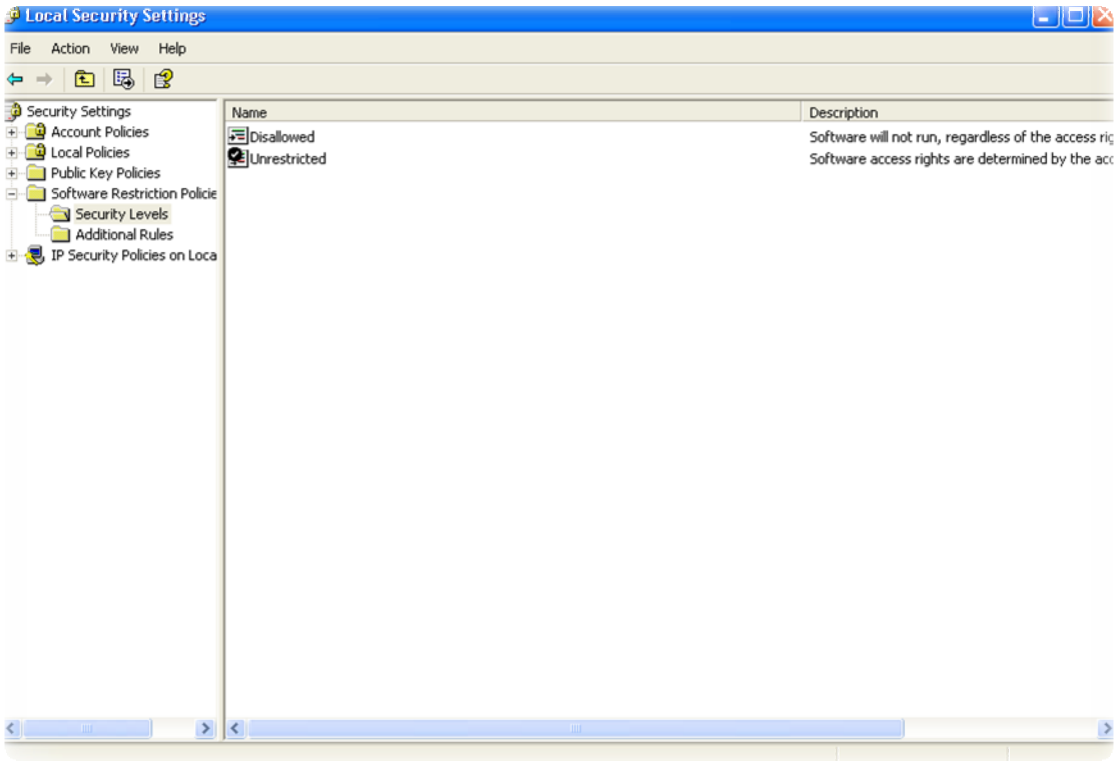
More and more software—especially on Windows—is being digitally signed. Other platforms are starting to emphasize code-signing, too: Apple's Mac OS X 10.5, for example, introduced new code-signing capabilities and concepts to that operating system (OS). As the OS begins to support and recognize signed code, we gain an incredible ability to control the applications that run in our environments.

Whitelisting Applications

For years, we've been fighting back malware through anti-software: antivirus, anti-spyware, anti-whatever. Essentially, anti-software is a way of *blacklisting* software—that is, identifying software that we *don't* want to run. It's silly because there is always going to be more software we *don't* want than software we *do,* and the list of bad software grows and changes incredibly quickly. But, for years, it's been our best defense against malware.

Digital certificates bring a new possibility: *whitelisting.* We identify the applications we allow, and everything else is disallowed. No more anti-software; we positively identify software by its digital signature, and we allow only identified software to run.

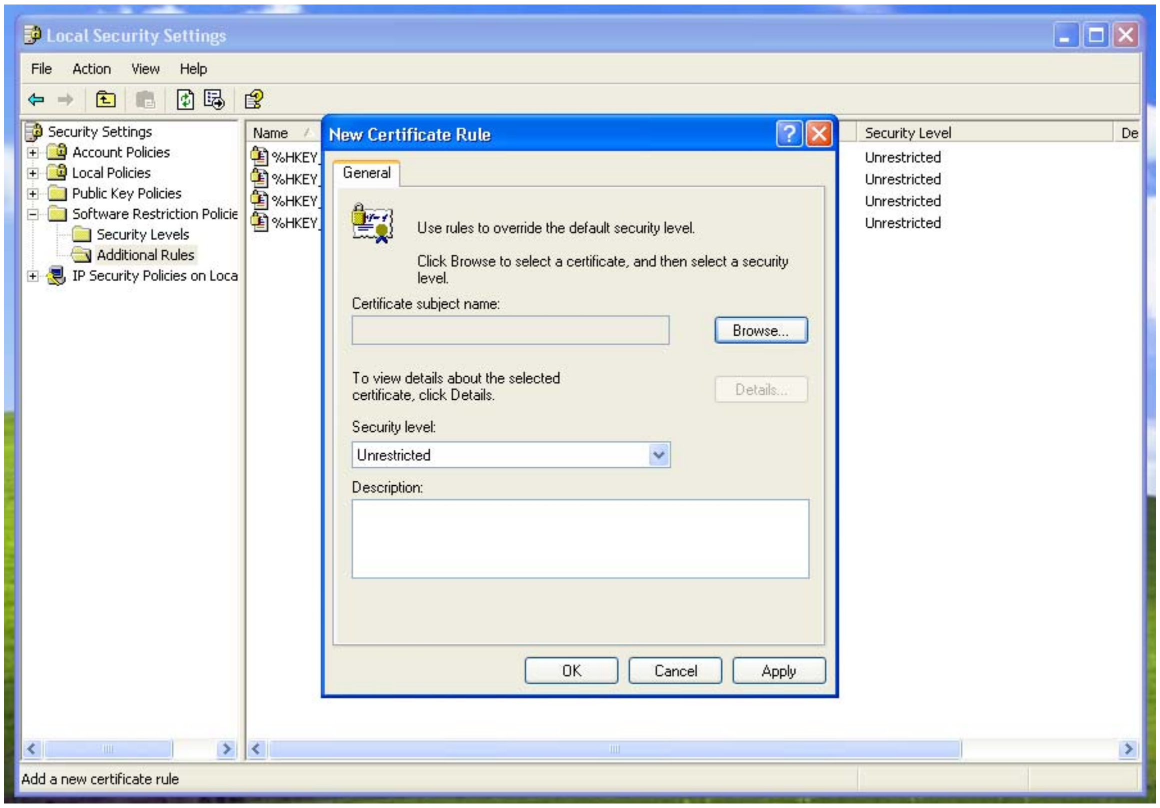
Microsoft built this capability into Windows XP and later versions of Windows. As shown in Figure 2.3, the OS default is to let everything run. However, by changing that to "Disallowed," you can then create rules that create a whitelist of applications that are permitted to run.



Name	Description
Disallowed	Software will not run, regardless of the access rig
Unrestricted	Software access rights are determined by the acc

Figure 2.3: Unrestricted by default—but with the potential to become a whitelisting machine.

Figure 2.4 shows how you can create a new *certificate rule,* which uses digital certificates to identify the software you'll permit to run.



Use rules to override the default security level.

Click Browse to select a certificate, and then select a security level.

Certificate subject name:

To view details about the selected certificate, click Details.

Security level: Unrestricted

Description:

Figure 2.4: Creating a new certificate rule.

In Windows, this technology is referred to as *Software Restriction Policies,* and it can be deployed centrally using Windows' Group Policy technology. Of course, one obvious—and significant—drawback is actually inventorying and identifying the software you want to permit. Microsoft plans to help with that in Windows version 7 by providing tools that help automate the process of identifying allowed software, and then turning that into a set of rules for you.

The bottom line: Code signing allows us to identify software that we trust, and allows us to finally consider ways of moving to a more secure environment that doesn't rely on continually updated anti-software to remain safe.

Software, Everywhere

Desktop software isn't the only kind that can benefit from the trust and security of digital certificates. Downloadable software—such as Java applets, ActiveX controls, Adobe AIR applications, and more—all benefit from having recognized, verifiable authors and tamper-evident code.

For Authentication

With their emphasis on identity, it's hardly surprising that digital certificates can find a use in authenticating users. In fact, *smart cards*—a popular means of authentication held to be far more secure than simple passwords—are really little more than intelligent repositories for digital certificates.

Here's how: A smart card contains a copy of a user's private key and is usually issued to a user in person, where their identity can be visually confirmed and checked against stored digital photos, government photo ID cards, and so forth. The smart card is inserted into a card reader, which is connected to the user's computer. The private key, however, *never leaves the card.* Instead, authentication requests are transmitted to the card and encrypted on the card—usually after the user "unlocks" or "enables" the card by entering a personal identification number (PIN). The encrypted data is sent back to the computer, which sends it off to the network's authentication system—such as a RADIUS server or Windows Active Directory (AD). That system was usually used to issue the private key in the first place, and retains a copy of the user's *public* key, which can be used to decrypt the authentication request. The mere fact that the decryption worked means that (a) the user has the right smart card and (b) the user knows the card's PIN. These two factors—something the user *has* and something the user *knows*—serve to authenticate the user more surely and securely than a password. It makes authentication easier on the user, too, since the user can remember a simple PIN instead of a complex, lengthy, ever-changing password.

Interesting Fact

Various—and thus-far-unsuccessful—attempts have been made to integrate smart cards into e-commerce. For example, American Express' original "Blue" card was a smart card, and card members were offered a reader that connected to their home computers. The idea was to use the card to encrypt sensitive information before sending it to online merchants; merchants would send the entire encrypted package to American Express for authorization, and the card number would never be entered online. Although consumers have yet to embrace this type of online security, it demonstrates the flexibility that digital certificates offer for protection and identification.

For Remote Access

As more users work from home and other remote locations, connecting them to their corporate data becomes more and more important. Traditionally, remote access was made through direct dial-up lines into the corporate data center, which was pretty secure all by itself—albeit numbingly slow for serious work. As high-speed Internet connectivity became more readily available, Virtual Private Networks (VPNs) started to replace direct dial-up lines. VPNs offer the ability to encrypt all the traffic flowing between the data center and the remote worker, and can do so using either simple symmetric encryption (passwords) or, for better security, using digital certificates and asymmetric encryption.

Today, a wide variety of remote access techniques are available—and all can benefit from better security and more positive user identification when combined with digital certificates:

- Remote desktop-style connectivity, such as Windows' Remote Desktop, can benefit from more secure authentication (as discussed earlier in this chapter).
- VPN-based connectivity can use digital certificates to provide easier authentication of users and better, asymmetric encryption for the VPN connection.
- Web-based remote applications, such as Outlook Web Access, can leverage Windows-integrated smart card authentication for better user identification, and utilize HTTPS to encrypt the data sent back and forth between the Web server and browser.

Note

Another popular means of making remote access more secure is hardware tokens, which generate single-use passwords (one-time passwords or OTPs) that are mathematically synchronized with a special server in the data center. Combined with a user name or PIN, these tokens offer some of the same benefits as certificates: easier for users, more secure than passwords, and the one-time password can potentially serve as a symmetric encryption key. But hardware tokens often require more supporting infrastructure than smart cards (although they don't require end users to have a card reader in their computer), and they don't *typically* provide encryption advantages as smart cards can.

For Web Services and Cloud Computing

Companies are expanding their notions of what constitutes an "application." Today, applications living on Web servers may provide information to or accept information from remote applications—these are *Web services.* Even the place where Web-based applications live is changing as *cloud computing* makes it easier to create highly distributed, highly scalable applications outside the company's own data centers. And as these applications—and all the sensitive data they deal with—leave the safety of the data center, companies will rely more and more on digital certificates to help provide security.

There are several main ways in which both Web services and cloud-based applications can benefit from digital certificates:

- Users of the applications can authenticate by means of digital certificates, ensuring that only authorized, positively identified users can access the applications and their data
- Data can be protected in-transit by asymmetric encryption provided by a Web server certificate
- Users can verify the identify of the Web server through its Web server digital certificate
- Even non-sensitive data can be protected from tampering by means of digital signatures, applied by the Web server and validated on the user's end

Companies that want to leverage the flexibility and scalability of cloud computing and Web services can do so *without* compromising their industry- or legally-mandated security requirements. Positive identification forms the basis for legally acceptable auditing measures, and encryption provides the data protection that modern compliance efforts demand.

For Mobile Applications

Cell phones are becoming more than just cell phones: Today's Blackberries, iPhones, and other mobile devices are "platforms," running a myriad of complex applications. Like any other computing device, there is always a concern about viruses and other malware; the additional concern on a phone is that the malware could gain access to the cellular data and voice networks, wreaking havoc on the infrastructure and causing massive service outages to millions of customers.

Interestingly, phone manufacturers recognized this possibility almost from day one, and *never* considered adopting the anti-software approach that we take for granted on our desktop and laptop computers. Instead, they opted for a "whitelisting" approach almost from the outset of mobile applications. Typically, cell phone manufacturers select a single CA, or a small set of CAs, that are trusted to issue code-signing certificates of the authors and publishers of mobile applications. In some cases, the CA resides with the manufacturer (as is the case with Apple and their iPhone); in other cases, existing commercial CAs are authorized to issue code-signing certificates to software authors and publishers. The phones themselves are simply hardcoded to not install—let alone run—applications that aren't signed with a trusted certificate. Figure 2.5 illustrates this simple way of ensuring only known, identified authors are allowed to put software on the phone.

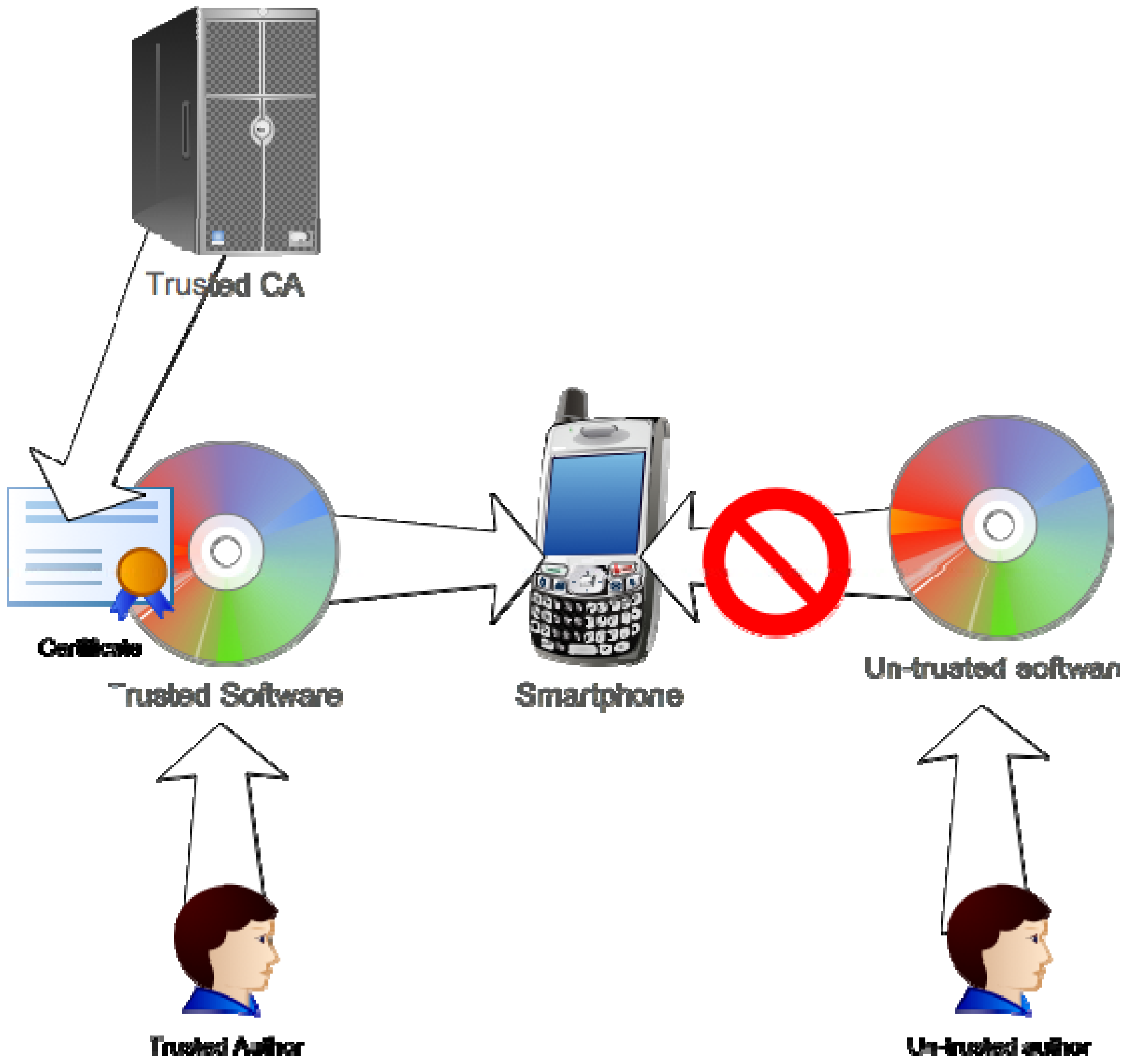


Figure 2.5: Protecting mobile phones by means of digital certificates.

It's important to note that, as with most other uses of certificates, the certificate only provides *a basis for our trust.* In other words, the certificate alone doesn't prevent malware. What prevents malware is the fact that, because of the certificate, we *know who the author is* and we can go find the author—and take appropriate action—if his or her software does any damage. Maliciousness exists most easily in an environment of anonymity; certificates deny anonymity and create an inhospitable environment for malice.

For Extranets

Extranets—an Internet-based application designed to connect companies with their customers in a less-public fashion than a normal Internet Web site—are providing companies and customers with easier ways to interact and share information. Because the information on an extranet is typically more sensitive than what would be on a public Web site, security is always a concern. Extranets can benefit in several ways from digital certificates:

- Customers can authenticate using digital certificates—either smart cards or, more frequently, certificates stored in their Web browser and presented on-demand to the Web server
- The company's Web server certificate ensures customers that they're connecting the legitimate extranet servers
- Both certificates allow for asymmetric encryption, protecting the data that passes between customers and the company

Most modern Web browsers support the storage and use of so-called *personal certificates*:

- All versions of Mozilla Firefox
- Internet Explorer versions 4 and later on Windows
- Opera 6.01 and later
- Chimera, OmniWeb, and Opera browsers on UNIX
- Netscape Communicator versions 4, 6, and 7 (although bugs exist in versions 6.2.1 through 6.2.3)

Cross-Reference

The Web page at http://www.dartmouth.edu/~pkilab/pages/More_Using_Web_Res.html#Browsers provides an excellent overview of Web authentication, and many of the concepts it covers apply well to extranet scenarios.

For Social Media

The increasing popularity of social networking and social media sites carries some alarming implications: More and more less-technically-savvy users relying on software that could potentially damage their computers, reputations, and so forth. Once again, certificates become a key way of identifying trusted software and Web site modules so that users can be assured of a safer computing experience.

Some social networking sites recognize the dangers inherent in exposing large groups of people to one another online, and are taking steps—utilizing certificates—to help protect their user communities:

- Web server certificates are used to secure logon sessions and other pages where sensitive data is being transmitted.
- Web server certificates help defeat phishing schemes, where malicious Web sites attempt to collect sensitive information by appearing to be the trusted Web site—users can be instructed to look for browser cues since only the legitimate site can be identified by means of a certificate
- Add-in software modules placed on users' Web pages can be digitally signed, permanently and positively identifying the software's author and helping to prevent tampering

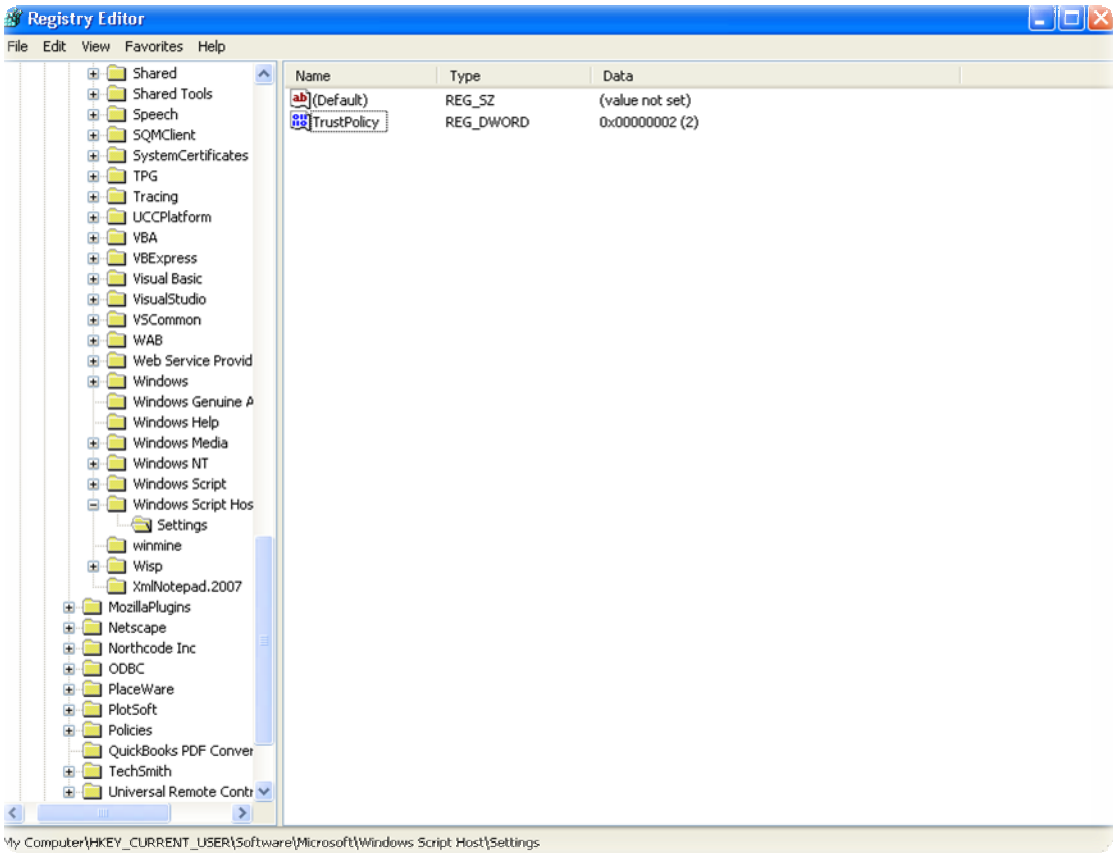
For Securing the Intranet

Digital certificates can play a tremendous role in securing internal resources as well, especially from particularly bothersome challenges that, in many cases, have been plaguing IT professionals for well over a decade.

Malicious Scripts

Take Microsoft's VBScript and Windows Script Host, an addition to Windows 95 that became extremely unwelcome for its use as a virus-authoring language. Similarly, Visual Basic for Applications (VBA), a feature embedded in versions of Office since 1997, was used to author more than a few famous viruses. Today, the potential for those VBScript- and VBA-based viruses exists and in most environments is mitigated only by anti-software such as antivirus utilities.

However, both Office *and* Windows Script Host offer the ability to completely cut off viruses—using digital certificates. Both technologies can be configured—centrally, through Group Policy—to run only scripts that contain a trusted digital signature, ensuring that the author's identity is known and that the script is exactly as the author wrote it. Again, this won't *stop* malice, but it does attach a name and a face to malice, which is usually enough to deter attackers. Figure 2.6 shows the registry setting necessary to run only signed scripts under the Windows Script Host; similar configuration settings in Office are enabled by default in more recent versions.



Name	Type	Data
(Default)	REG_SZ	(value not set)
TrustPolicy	REG_DWORD	0x00000002 (2)

My Computer\HKEY_CURRENT_USER\Software\Microsoft\Windows Script Host\Settings

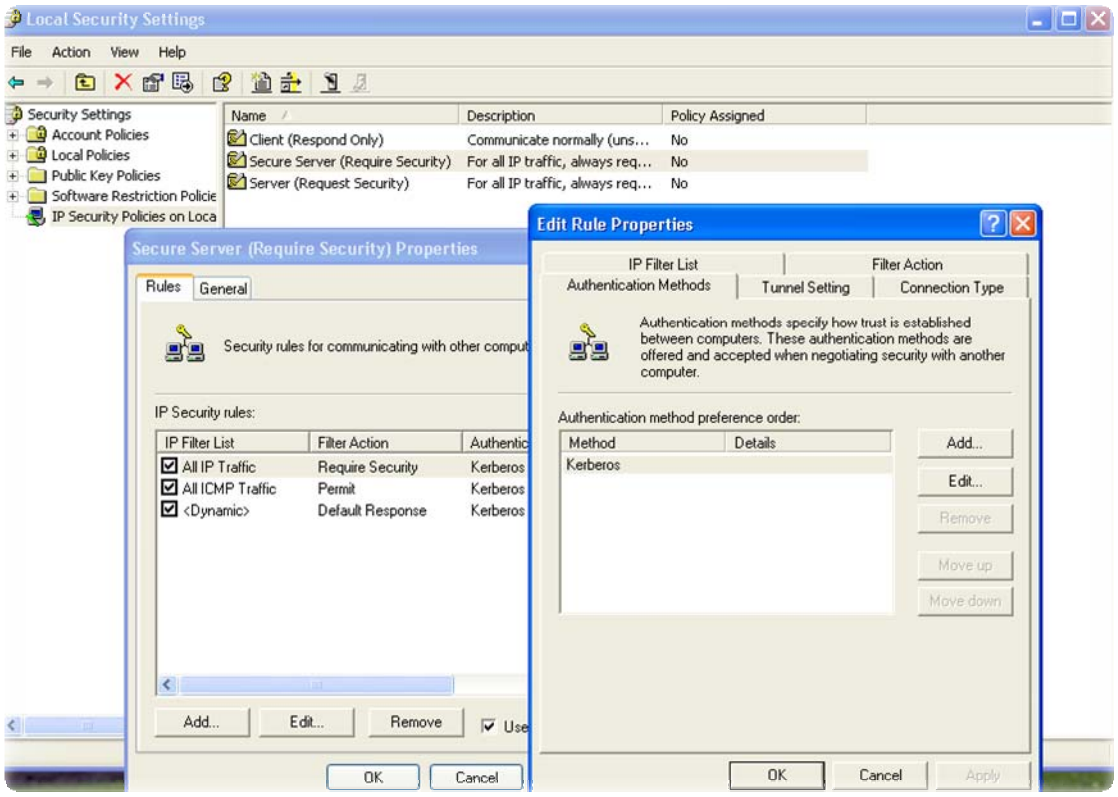
Figure 2.6: Configuring the Windows Script Host TrustPolicy in the registry.

VBScript has been a security concern for well over a decade; for much of that time, the solution was as easy as a registry setting and a code-signing certificate (if you want to sign legitimate scripts so that they'll run in a secured environment). Microsoft's newer administrative shell, Windows PowerShell, uses exactly the same security model (although it defaults to not running any scripts, signed or not, and lets you make the decision about what scripts you'll allow).

Meeting Compliance Requirements

Legal and industry requirements for security are getting progressively more detailed and difficult to meet and maintain. Although certificates aren't the only answer, they are an important component of an overall security system.

For example, one common requirement is that data be protected as it's transmitted across networks. Certificates and the way they enable asymmetric encryption can do this through a technology called IP Security (IPSec). Figure 2.7 shows Windows XP's IPSec policy configuration, which is also present in later versions of Windows. Windows comes with a preconfigured "Secure Server (Required)" policy, which tells servers to communicate only with clients that can establish a secure session.



Name	Description	Policy Assigned
Client (Respond Only)	Communicate normally (uns...	No
Secure Server (Require Security)	For all IP traffic, always req...	No
Server (Request Security)	For all IP traffic, always req...	No

Security rules for communicating with other comput

IP Security rules:

IP Filter List	Filter Action	Authentic
☑ All IP Traffic	Require Security	Kerberos
☑ All ICMP Traffic	Permit	Kerberos
☑ <Dynamic>	Default Response	Kerberos

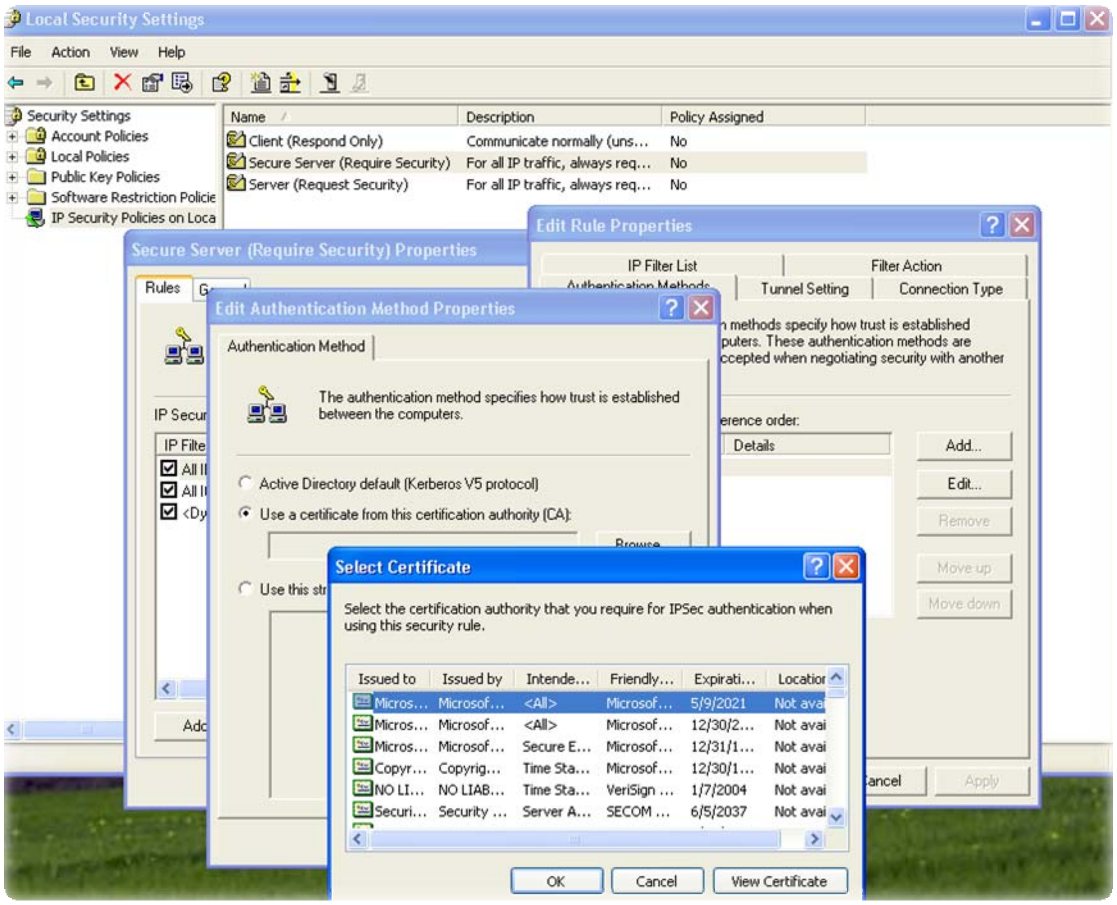
Authentication methods specify how trust is established between computers. These authentication methods are offered and accepted when negotiating security with another computer.

Authentication method preference order:

Method	Details
Kerberos	

Figure 2.7: Windows XP IPSec policies.

As shown, Windows defaults to securing that connection with Kerberos, which is useful when all the clients in the environment are using versions of Windows that can authenticate to AD. However, in mixed environments, or in environments where AD isn't the only authentication mechanism, IPSec can be reconfigured to use certificates for authentication and encryption. Figure 2.8 illustrates this configuration change.



Name	Description	Policy Assigned
Client (Respond Only)	Communicate normally (uns...	No
Secure Server (Require Security)	For all IP traffic, always req...	No
Server (Request Security)	For all IP traffic, always req...	No

The authentication method specifies how trust is established between the computers.

Active Directory default (Kerberos V5 protocol)

Use a certificate from this certification authority (CA):

Use this str

Select the certification authority that you require for IPSec authentication when using this security rule.

Issued to	Issued by	Intende...	Friendly...	Expirati...	Locatior
Micros...	Microsof...	<All>	Microsof...	5/9/2021	Not avai
Micros...	Microsof...	<All>	Microsof...	12/30/2...	Not avai
Micros...	Microsof...	Secure E...	Microsof...	12/31/1...	Not avai
Copyr...	Copyrig...	Time Sta...	Microsof...	12/30/1...	Not avai
NO LI...	NO LIAB...	Time Sta...	VeriSign ...	1/7/2004	Not avai
Securi...	Security ...	Server A...	SECOM ...	6/5/2037	Not avai

Figure 2.8: Configuring IPSec to use certificates for authentication.

Certificate authentication can also be added *in addition to* Kerberos authentication, giving IPSec clients a choice of protocols.

Secure In-Situ

Much of our discussion behind certificates and security has focused on protecting data in-transit: FTPS, HTTPS, IPSec, and so forth are all designed to protect communication channels. But what about protecting that same data when it's sitting on a hard disk?

Most OSs, including Windows, UNIX, Linux, and Mac OS X, provide some support for encrypting files on disk; Windows' feature is called the Encrypting File System (EFS). A concern with this system in most enterprises is the ability for a user's account to become disabled or damaged, losing the user's encryption key and forever blocking access to an encrypted file. Certificates and their supporting infrastructure can address this need by providing users with encryption keys that are private but can still be recovered—if needed—through a highly secure recovery process that may involve the cooperation of multiple administrators (to prevent a single administrator from abusing the process).

Certificates Everywhere!

The fact is that certificates are already all around us: They're used to sign code on our desktops, sign mobile applications, protect Web site communications, secure remote access connections, and much more. Unfortunately, we often don't *use* those certificates as much as we might, meaning that we still struggle to solve IT problems—such as malware—even though certificates offer a straightforward, usable solution. Understanding the many ways in which certificates can help us solve security and identity problems is the first step to solving some of our trickiest troubles.

In the next chapter, we'll explore one of the most important concepts embodied in digital certificates: trust. It's something we've talked about briefly here, as well as in the previous chapter, but it's a complex topic that deserves further explanation.