Covers All Exam Objectives for IINS 640-553

Includes Real-World Scenarios, Hands-On and Written Labs, and Leading-Edge Exam Prep Software Featuring:

- Custom Test Engine
- Hundreds of Sample Questions
- Electronic Flashcards
- Entire Book in PDF

CCNA®

Security

STUDY GUIDE

IINS Exam 640-553

Tim Boyles

SYBEX | SERIOUS SKILLS.

Table of Contents

Chapter 11

Using Asymmetric Encryption and PKI

THE FOLLOWING CCNA SECURITY EXAM TOPICS ARE COVERED IN THIS CHAPTER:

- ✓ **Asymmetric encryption usage**
- ✓ **Asymmetric algorithms and the computations behind them**
- ✓ **The components of a PKI**
- ✓ **Certificate authorities and their structures**
- ✓ **Digital certificates creation, information and usage**
- ✓ **PKI standards**

In this chapter we are going to continue our discussion from previous chapters on encryption. We will discuss asymmetric encryption and the uses of the technology. Once you understand asymmetric encryption, we will take a deep dive into the workings of two of the primary algorithms, including the mathematics behind the algorithms and the benefits and drawbacks for each solution.

We will then move into the Public Key Infrastructure (PKI), looking at the components of a PKI and how these components work together. Our discussion will then shift to digital certificates. We will outline the enrollment, usage, and information fields for digital certificates.

In the last section, we will look at some of the common PKI standards that provide the foundation of the PKI environment.

Asymmetric Encryption

Asymmetric encryption, also called *public key cryptography (PKC),* is an encryption technique that utilizes a key pair. The key pair includes a public key and a private key. The public key is just that—available to the public. It is made available to anyone who needs to use it. The private key is to be used by a single entity and has to remain private to maintain its integrity.

As with our discussions of symmetric encryption and secret key cryptography in previous chapters, we know that one of the drawbacks of symmetric encryption is the difficulty of sharing a secret key. How do you distribute the secret key, how often should you change the secret key, and what happens if the secret key is compromised? Because of these issues, *asymmetric encryption* was created. Asymmetric encryption allows two parties to create a secure communication channel without any prior knowledge of each other; that is, they don't have to share a secret key. It accomplishes this by using the public key cryptography process.

One drawback of asymmetric encryption over symmetric encryption is processing time. Asymmetric encryption is much slower than symmetric encryption because of the massive computational mathematics involved, which is what makes the technology secure. But the slow performance of this encryption method makes asymmetric encryption impractical for many applications. The main use for asymmetric encryption is to provide a means for creating secret keys for symmetric encryption.

With asymmetric encryption, both the public and private keys can be used for encryption and decryption. Within a single message exchange, though, each key can provide only encryption *or* decryption. If the public key is used to encrypt the message, the private key is used to decrypt, and vice-versa. Different security features are achieved depending on which key is used to encrypt the message. These security features include confidentiality and authentication.

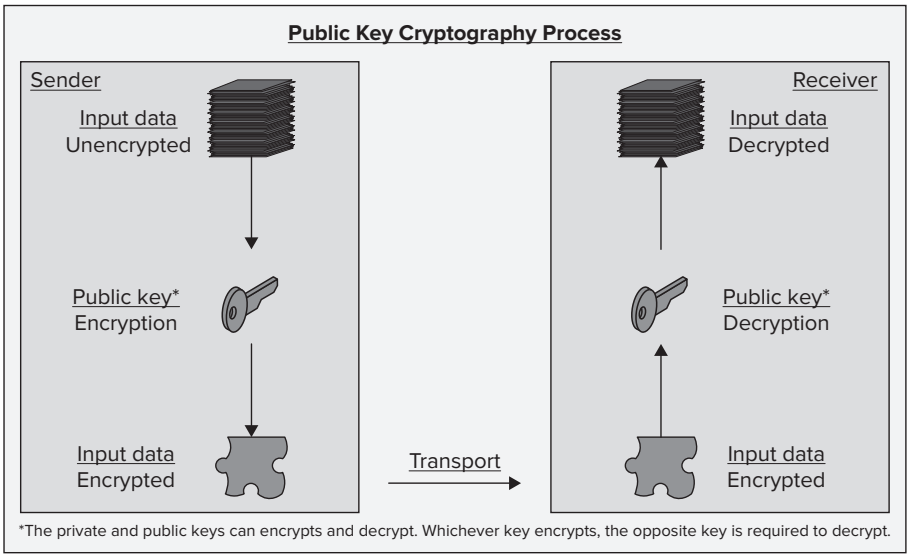
Confidentiality provides a means to keep data private. When information travels over an insecure medium such as the Internet, confidentiality is a necessity. Asymmetric encryption provides confidentiality with the use of the public key.

Authentication provides a means to prove the origin of the data. Knowing where the data comes from is very important. Without authentication, a hacker could impersonate a source and deliver compromised data. Asymmetric encryption utilizes the private key to prove that the source of the data is valid.

Public Key Cryptography Process

In this section we will discuss the components, features, and usage of asymmetric encryption. The components include the public and private key exchange. We will discuss confidentiality and authentication in the features section. Finally, we will provide practical usage information for asymmetric encryption. Figure 11.1 shows the public key cryptography process.

FIGURE 11.1 Public key cryptography process



Key Pairs

The public key cryptography process revolves around two keys, one available to the public and the other private, belonging to a single entity. The public and private keys must have different values to be effective. You cannot take the public key and determine the private key, at least not in any time frame that would be relevant.

Key Size

The key size for asymmetric algorithms varies. For most asymmetric algorithms, the key sizes range from 512 bits and 4096 bits. Typically the key size is a multiple of 512 (for example, 512, 1024, 2048, or 4096). Most asymmetric algorithms can utilize variable-length keys.

As with any encryption technique, the longer the key is, the more secure the output will be. This security, of course, is at a cost of computing time. The most common recommendation is to use an algorithm of 2048 or higher for asymmetric encryption.

Private Key

The private key can be calculated multiple ways depending on the algorithm. Some algorithms determine the private key after the public key process has finished. Other algorithms pick a random number and use that as the private key.

On Windows systems, the private keys can be stored in multiple places. They are typically stored in the Registry but can be stored as files in the file system.

The private key can be used to encrypt or decrypt data. In most cases it is used to decrypt data that has been encrypted with the public key. The reason for this is that many asymmetric algorithms are used on the Internet for secure communication to websites. Because most people do not have their own keys, they will request the public key of the website. They will use the public key to encrypt the traffic, which means that the website will have to use its private key to decrypt the traffic.

Public Key

The public key is meant to be used by anyone. No security is given to the public key. It is presumed to be in the public domain with other eyes on it.

Asymmetric algorithms generate public keys differently. Some algorithms pick random numbers, whereas other algorithms create the private key first and then calculate the public key.

The asymmetric algorithm has to be strong enough that the public key cannot be used to determine the private key.

Features

Asymmetric encryption provides two major features: confidentiality and authentication. It accomplishes each of these features in different ways depending on the use of the public or private key.

Confidentiality is achieved by the use of the public key. When the public key is used for encryption, the private key has to be used for decryption. The private key is on one system. When the data is encrypted with the public key, only the system with the private key can decrypt it. Thus we are confident that the data is being decrypted by the right system and only by that system. This preserves the confidentiality of the data.

Authentication is achieved when the private key is used to encrypt the data. We know that only the public key can decrypt data that is encrypted by the private key. We also assume that the public key is known by everyone. If we receive data and can decrypt the data with the public key, we know that the data came from the owner of the private key and no one else. This is authentication because we proved that the system that sent us the data is what it claims to be.

Drawbacks

The main drawback of asymmetric algorithms is the time it takes the algorithms to calculate the keys. The structure of asymmetric algorithms is to use large numbers, which make it secure. These numbers can be hundreds of bits long. Asymmetric algorithms have complex mathematics behind them to generate these keys. Because of the complex mathematics, it takes the processor more time to compute all of these keys—many seconds in some cases compared to milliseconds for some symmetric algorithms.

Another important drawback is the asymmetric algorithm strength compared to the strength of symmetric algorithms. There is no concrete data that proves the ratio between symmetric and asymmetric strengths, but it is important to know that asymmetric algorithms are considered weaker than their symmetric counterparts. An asymmetric algorithm with a 1024-bit key can be 50–90 percent weaker than the same 1024-bit key for symmetric algorithms. This will come into play when you use an asymmetric algorithm to generate a key for a symmetric algorithm. New symmetric algorithms like Advanced Encryption Standard (AES) have variable length strengths and require large secret keys. To generate a secret key via asymmetric encryption, the algorithm will have to generate a larger key and may have to use an asymmetric algorithm of 6144 bits to match the 256-bit or 512-bit key of the symmetric algorithm.

Usage

The main use of asymmetric algorithms is to generate secret keys for symmetric algorithms. Because of the drawbacks we just discussed, asymmetric algorithms are not suited for large data encryption. But they are the best solution when it comes to creating secret keys between two unknown devices. These two devices can be a web server and a PC or two Virtual Private Networks (VPN) connections with limited knowledge of each other.

Real World Scenario

Securing a Web Server for Email Access

The remote sales office for the XYZ Corporation is hosting a web server for email access. The communication on the web server is not secure. Your boss has tasked you with securing the web server. Currently when you go to the website to check your email, the address is `http://mail.xyzcorporationtest.com`.

You wish to provide a level of encryption to this address. You decided to enable security with digital certificates. Now when you go to the website, you have to enter `https://mail.xyzcorporationtest.com`. The address starts with `https`, meaning it is a secure protocol. The protocol behind `https` is Secure Socket Layer (SSL) or Transport Layer Security (TLS). SSL and TLS are both types of asymmetric encryption that use digital certificates. The web server will have a private key. You will have to obtain the public key from a certificate authority. A key exchange will occur to create the secure channel. Once the secure channel is created, you can browse the website securely. This will ensure that no one can view your private email unless they are peeking over your shoulder.

Hybrid Encryption

A hybrid cryptosystem utilizes both asymmetric and symmetric encryption techniques. There are two main types of encryption with a hybrid encryption solution:

- Key exchange encryption—Encrypt the keys with asymmetric encryption.
- Data encryption—Encrypt the data with symmetric encryption.

Because asymmetric encryption is slow and requires more computing power, it is best suited for small amounts of data. Asymmetric encryption is perfect for encrypting keys. Symmetric encryption is much faster and capable of encrypting large amounts of data efficiently. This makes symmetric encryption perfect for data confidentiality.

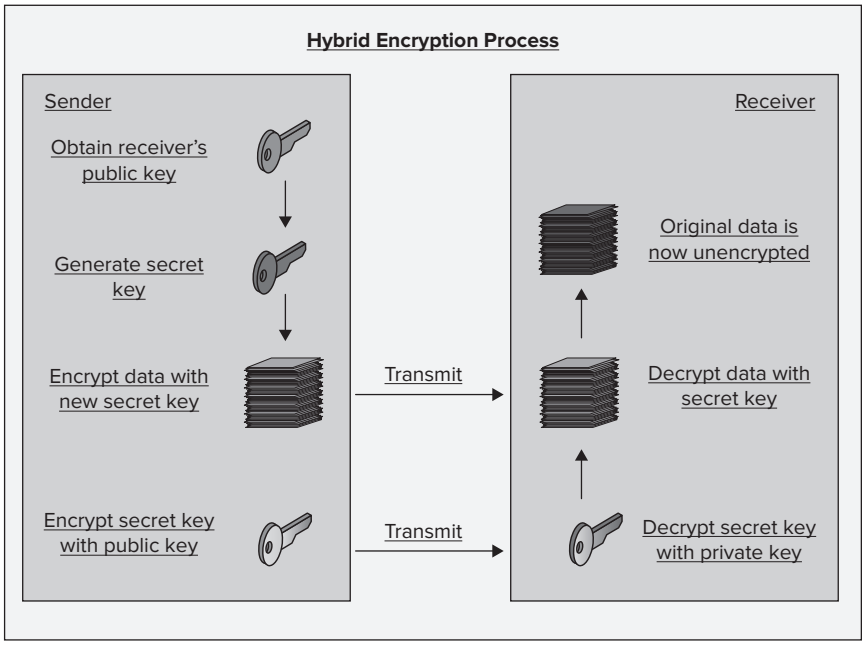
The following process is used in hybrid encryption:

1. Encryption from sender
   a. Sender obtains the public key of the receiver.
   b. Sender creates a symmetric secret key with the public key and asymmetric encryption.
   c. Sender encrypts the data using the newly created secret key and symmetric encryption.
   d. Sender encrypts the symmetric secret key with the public key and asymmetric encryption.
   e. Sender transmits both the encrypted secret key and data to the receiver.

2. Decryption from receiver
   a. Receiver uses their private key to decrypt the symmetric secret key.
   b. Receiver uses the symmetric secret key to decrypt the data.

Figure 11.2 shows the hybrid encryption process.

FIGURE 11.2 Hybrid encryption process



An example of a hybrid encryption solution is*Secure Shell (SSH)* . SSH utilizes PKC to authenticate the remote PC when connecting. If you have ever used SSH, you should have noticed a message about the SSH signature. You have to accept the signature before making a connection to the SSH-enabled device. Most SSH clients require only this during the first connection and save the signature for subsequent connections. Once the connection is authenticated, symmetric encryption is used to secure the data between the PC and the device.

Asymmetric Encryption Algorithms

The asymmetric encryption algorithm is the heart of asymmetric encryption. The algorithm is the mathematical engine that crunches the numbers to compute the public and private keys. There are multiple asymmetric encryption algorithms, and each algorithm determines the keys differently.

Table 11.1 lists a few sample asymmetric algorithms.

TABLE 11.1 Sample Asymmetric Algorithms

Acronym	Full Name and Description
RSA	Rivest, Shamir, and Adleman. Most widely used PKC in e-commerce transactions.
DH	Diffie-Hellman. Creates secret keys for symmetric encryption.
DSA	Digital Signature Algorithm. Used to create digital signatures.
ECC	Elliptic Curve Cryptography. Utilizes an algebraic structure for key creation.
ElGamal	ElGamal Encryption System. Based on the Diffie-Hellman Key Exchange.

We will discuss two of these algorithms in the following section: RSA and Diffe-Hellman.

RSA Algorithm

We started discussing the features of RSA in Chapter 10,"Using Digital Signatures." In this section we will take a deeper look at how RSA works. The RSA encryption algorithm was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman of MIT. A patent was filed in 1977 and issued in 1983. The three inventors of the RSA algorithm formed a company called RSA Security, which released the RSA algorithm into the public domain in 2000, three years before the patent would have expired.

The RSA algorithm is one of the most widely used asymmetric algorithms. It is built into Microsoft Internet Explorer and a host of other browsers. It is also used in SSH, IPSec, PGP, and TLS. The RSA algorithm has been in existence for many years and is a proven technology.

The security of the RSA algorithm is simple to explain and simple to generate. Two prime numbers are used, and multiplying these numbers is easy. But determining the original prime numbers from the total is infeasible because of the time it would take to compute the numbers with modern computers.

The RSA algorithm is capable of using variable key lengths between 512 and 2048 bits. It is recommended that you use 2048 bits when possible. The shorter numbers have proven vulnerable because of increased computing power and the use of distributed computing employing hundreds of computers to guess the key.

The three main functions in the RSA algorithm are key creation, encryption, and decryption. We will look at these three functions in greater detail.

RSA Key Table

Table 11.2 lists the variables we will use in the "RSA Key Creation," "RSA Encryption Example," and "RSA Decryption Example" sections that follow.

TABLE 11.2 RSA Key Chart

Key	Description	Public or Private
p	Prime number	Public
q	Prime number	Public
e	Public key	Public
m	Multiplier to determine *d*	Private
d	Private key	Private
n	Remainder	Public
s	Secret key	Private

RSA Key Creation

In the RSA algorithm, key creation starts with two different large prime numbers, *p* and *q*, and a number *e*, which is not divisible by *p* or *q*. To determine the public key you need to know the values of *p* * *q* and the number *e*. The numbers for *p* and *q* are randomly generated numbers. The number *e* is typically a small integer of 3, 17, or 65537. A small integer is used for *e* to help with computational speed.

- [*e*, *p* * *q*] equals the public key.

To obtain the private key we have to find *d*. The number *d* is a number that when multiplied by *e* equals (*p* -1) * (*q* -1) + 1. Or *d* needs to equal some multiplier *m* of (*p* –1) * (*q* -1) + 1 = *m* * (*p* -1) * (*q* -1)) + 1. The number *d* plus *p* * *q* equals the private key.

- [*d*, *p* * *q*] equals the private key.
- Always keep the value of *d* secret.

Determining the value of *d* is typically accomplished by the Euclidean algorithm. The Euclidean algorithm is an effective way of calculating the *greatest common divisor (GCD)*. Find the GCD using the following equation:

e * *GCD* = *m* * (*p* -1) * (*q* -1)) + 1

In this case, the GCD will equal *d*.

These equations are secure because determining the original prime numbers from the result of the equations is infeasible. You could probably determine the prime numbers and ultimately d or e if you had enough time. But with large enough prime numbers, it would take thousands of years. It is assumed that by the time you calculated the numbers, the decrypted information would be of no use.

Let's assume a person lives 100 years and obtains a social security number when they are born. This would mean that the social security number should remain private for 100 years. If a 50-year-old person transmitted their social security number over the Internet and someone captured the encrypted data, the hacker would have only 50 years to decrypt the data to gain something useful. If the data were encrypted using the RSA algorithm, it would take thousands of years to compute the prime numbers used to encrypt it. Therefore, it is very unlikely that someone could break the encryption and gain access to the encrypted data.

RSA Encryption Example

We described how to compute both the public and private keys. Now let's put some practical numbers around them. We will use small prime numbers for simplicity's sake. With small prime numbers the equation will get large. Just imagine if we were to use large prime numbers.

We will create the public key using the following numbers. These numbers can be picked at random as long as they meet the criteria.

$p = 3$

$q = 5$

$e = 11$

As you can see with these numbers, p and q are prime numbers and e is not divisible by p or q. Now we need to determine the value of d.

$e * \text{GCD} = m * ((p - 1) * (q - 1)) + 1$

Let's start by breaking down the equation with known values.

$((p - 1) * (q - 1)) = ((3 - 1) * (5 - 1)) = ((2) * (4)) = 8$

$11 * \text{GCD} = m * 8 + 1$

We need to figure out what number times 11 will equal some number times 8 + 1. For our purposes this is just trial and error. Typically, the Euclidean algorithm would be used to find the GCD. So we will start with 1.

$11 * 1 = 1 * 8 + 1$

As you can see, 1 did not work. Let's try the number 11 for the GCD and 15 for m.

$11 * 11 = 15 * 8 + 1 = 121$

$m = 15$ = any number to make the equation work, m is not needed after d is determined

$d = 11$

This works. Yes, it took some time to find the right numbers, and we used only single-digit prime numbers for *p* and *q* and a small value for *e*. Now we have all the numbers to compute the public and private keys.

Say we want to share a secret key that we determined by other means. We need to send this secret key to the recipient. For simplicity's sake, let's use the number 3 for the secret key.

First we need to take the secret key of 3 to the power of *e* modulus *p* * *q*.

$3^{11} \bmod p * q = 3^{11} \bmod 3 * 5$

$3^{11} \bmod 15 = 177147 \bmod 15$

Now that we know that $3^{11} = 177147$, we need to divide 15 into 177147.

177147 ÷ 15 = 11809 *remainder* 12

In this equation it is the remainder that we need. In this case the remainder of 12 is sent to the recipient.

RSA Decryption Example

We will continue the equation from the encryption section. We just received the value of 12 from the sender. Being the recipient and the holder of the private key, we already know a few things.

$p * q = 15$

$d = 11$

$n = 12$

To determine the original secret key that was encrypted, we will take *n* to the power of *d* mod 15. Remember that *p* * *q* never changes.

$12^{11} \bmod 15 = 743008370688 \bmod 15$

Now we need to determine how many times the large number is divisible by 15.

743008370688 ÷ 15 = 49533891379 *remainder* 3

15 * 49533891379 + 3 = 3 mod 15

3 = *s* = secret key

If you look at the previous equations you will notice that the remainder is 3. This remainder will be the same value as the original value, encrypted *s*. In our case it is our secret key.

Summary of the RSA Algorithm

Table 11.3 shows the main topics to remember about the RSA algorithm.

TABLE 11.3 Summary of the RSA Algorithm

Topic	Description
Usage	Widespread and the primary algorithm used in e-commerce.
Speed	Very slow and typically used for key creation only.
Patent	Released in the public domain in 2000.
Key length	Variable, between 512 and 2048.
Keys	Public and private keys are used.
Security	Strong when using large keys (e.g., 2048-bit key lengths).

Diffie-Hellman Algorithm

Diffie-Hellman (DH) is an asymmetric algorithm used to create shared secret keys over insecure channels like the Internet. Whitfield Diffie and Martin Hellman invented the algorithm in 1976. It was later uncovered that the same techniques were invented earlier by a government entity but kept classified. Diffie-Hellman is described in RFC 2631.

Diffie-Hellman can utilize variable-length keys for the security. Diffie-Hellman calls these different key lengths *groups* . The original DH group was 1 with a key length of 768 bits. The strength of this key length is weak today because of greater computing power and distributed computing techniques. DH Group 2 is a 1024-bit key and is the standard key length for Cisco VPN solutions. DH Group 5 is becoming the new standard and has a key length of 1536 bits.

RFC 3526 describes the DiffieHellman group types and key lengths. There are larger groups than Group 5 to deal with the larger symmetric key requirements of AES and some of the newer symmetric algorithms.

DH Key Table

Table 11.4 lists the variables we will use in the"DH Key Creation" and "DH Key Exchange Example" sections that follow.

TABLE 11.4 Diffie-Hellman Key Chart

Key	Description	Public or Private
p	Prime number	Public
g	Primitive root	Public
a	Party A's private key	Private
b	Party B's private key	Private
g^a or A	Party A's public key	Public
g^b or B	Party B's public key	Public
As or $(g^b)^a$	Party A's shared secret key	Private
Bs or $(g^a)^b$	Party B's shared secret key	Private

DH Key Creation

The goal of the Diffe-Hellman algorithm is to create a secret key shared between two parties. It accomplishes this by generating a public key and a private key.

Diffe-Hellman requires two numbers to start the process. In Diffie-Hellman only one of the numbers (p) is prime, whereas RSA uses two prime numbers. In DH, the other number g is a primitive root $g \mod p$. Typically, g is a small number, either 2 or 5. Both p and g are public numbers that are agreed on by both parties. Usually one party will generate p and g and then send the numbers to the other party. The receiving party either accepts or refuses the numbers.

Once both parties agree on p and g, each party needs to determine their private key (a and b, for party A and B, respectively). The private key can be any random natural number. As the name suggests, this number needs to remain private and is never transmitted over the insecure network.

Now that each party has chosen a private key, these numbers are used to generate the public keys for each party. If you remember, in RSA the public key generates the private key, but in Diffie-Hellman the private key is used to generate the public key. The equation to accomplish this is $g^a \mod p$ for party A and $g^b \mod p$ for party B. Remember that a and b are each party's chosen private keys. The results from the equation that follows will generate the public key and will be represented with an A or B.

Party A $g^a \mod p = A$

Party B $g^b \mod p = B$

Once the computation is complete, the public keys are known for each party. So far p, g, g^a g^b, A, and B are publicly known and sent to each party in the clear.

Now that the corresponding parties know each other's public key, they can compute the shared secret key. The following equation is used to determine the shared secret key, represented by As or Bs.

Party A: $B^a \mod p = As$

Party B: $A^b \mod p = Bs$

Once the shared secret key is determined, it is typically used with a symmetric algorithm as the secret key. This allows the symmetric algorithm to generate a secret key without prior knowledge and over an insecure network.

All these equations come down to one simple equation to determine the shared secret key.

Party A: $(g^b)^a = As$

Party B: $(g^a)^b = Bs$

Because $(g^b)^a$ and $(g^a)^b$ are capable of generating the shared secret keys, these equations and corresponding numbers need to remain private. These numbers are never transmitted over the insecure network.

DH Key Exchange Example

Parties A and B want to exchange a shared secret key. Parties A and B agree on the following prime and primitive root numbers:

$p = 7$

$g = 2$

So far we know $g \mod p$ is 2 mod 7. Next, each party needs to pick a private key. Party A and B choose the following:

Party A: $a = 4$

Party B: $b = 5$

Once the parties have decided on their private keys, they need to compute the equation and send the results to the other party. The result of this is called the public key:

Party A: $A = g^a \mod p = 2^4 \mod 7 = 2$

Party B: $B = g^b \mod p = 2^5 \mod 7 = 4$

In this example the actual math is not shown. Refer to the RSA section to review the math to determine the final answer, or use a scientific calculator to perform the math.

Now Party A will send Party B the findings of $A = 2$ and Party B will send Party A the findings of $B = 4$. For each party to compute the shared secret key s, they will use the following equation:

Party A: $As = B^a \mod p = 4^4 \mod 7 = 4$

Party B: $Bs = A^b \mod p = 2^5 \mod 7 = 4$

As you can see by these equations, both parties determined the same shared secret key of 4. Normally this algorithm would use much larger numbers to compute the shared secret key.

Summary of the Diffie-Hellman Algorithm

Table 11.5 shows the main topics to remember about the Diffie-Hellman algorithm.

TABLE 11.5 Summary of the Diffie-Hellman Algorithm

Topic	Description
Usage	Establish shared secret key between two parties over an insecure network.
Speed	Much slower than symmetric algorithms.
RFC	RFC 2631 was released in 1999.
Key length	Variable, Group 1 = 768-bit, Group 2 = 1024-bit, Group 5 = 1536-bit.
Keys	Public and private keys are used.
Security	Strong when using large keys like 1024 and 1536.

Public Key Infrastructure

The Public Key Infrastructure (PKI) is a framework that includes servers, workstations, users, networks, corporations, policies, and software. The PKI framework facilitates the means to distribute, administer, verify, and revoke digital identities.

PKI environments are primarily used over insecure networks utilizing public key cryptography. Over these insecure networks a means had to be created to verify the identity of a party and generate a secure channel for communications. PKI is one way to accomplish this.

There are other ways to verify identities and create secure channels. One way is called a web of trust, where there is no centralized authority. This technique is used with Pretty Good Privacy (PGP). We have discussed some other methods in previous chapters. Those methods are not scalable. PKI is typically used for large environments where administration efforts need to be kept to a minimum.

PKI Overview

PKI uses a centralized method to administrate digital identities. Two main technologies allow for this:

- Certificate authorities
- Digital certificates

The first technology involved in administering digital identities is certificate authorities (CA). As with any type of authority, an assumption of power is required. This assumption of power can be either given or taken. For example, a government might take power from the people by using force, or the people might give power to the government by utilizing government health services. The point being made is that the assumption of power can move people, minds, and countries. The assumption of power is the foundation of the PKI framework. Without the assumption of power and authority, the PKI framework would not survive.

PKI is based on power and authority that is given. In other words, the people using the PKI give authority and power to the administrators of the CA. The CA is a company that provides the services needed to manage the PKI. Because the CA is typically a third-party entity, we call this a trusted third party (TTP).

TTPs are entities that people look to as the authority. For the TTP to work, all members within the PKI must accept the authority of the TTP. If the TTP is not authoritative, then the rules, regulations, and punishment are of no value.

Another technology of managing digital identities is digital certificates. A digital certificate is an electronic ID that can be used to identify people, computers, servers, routers, and software, to name a few.

As the Internet has grown over the years, the need to provide virtual identities has grown. There has to be a good and efficient way to prove who is who over this insecure network. A digital certificate does just that. It is proof that you are who you say you are.

Let's look at these two areas more closely. We will first look at TTPs by means of certificate authorities. Then we will look at digital certificates and the IDs they are today.

Certificate Authorities

Certificate authorities are the TTPs that provide the power and authority in a PKI. A CA is similar to a DMV or the Social Security Administration. All these entities have the power and authority to provide you with (or revoke) a form of identification. The DMV provides a driver's license and the SSA provides a social security card. The CA has the power and authority to provide you with a digital certificate, verify a certificate, or revoke one.

What is a CA? Is it a group of people in suits and sunglasses driving Crown Vics? Not exactly. Most CAs are companies that provide a service. This trusted service is to help people identify each other over insecure networks virtually. If you have ever heard of VeriSign, you have heard of a PKI and CA provider. One thing to note about CAs is

that not all CAs are service companies. A CA can be private, associated with a single company, or it can be public. VeriSign is an example of a public CA service. A CA at your workplace that supplies digital certificates only to the Active Directory (AD) domain PCs is considered a private CA.

Now that you have a little background on certificate authorities, let's dig a little deeper. The first thing we should talk about is that a CA is really a service on a server. Although in one of its definitions CA means all the things that encompass the certificate authority, when we get right down to it, the CA is a server or multiple servers.

A CA server provides multiple services. These services include the root public key, certificate requests, creations, revocations, and queries. The first CA created in a CA environment is considered the Root CA and provides the root public key. A CA will process certificate requests by creating new certificates. The CA server can also revoke bad certificates and then provide a list of bad certificates in response to queries from users and computers. Depending on the size of the CA environment, all of these services may be on one server. Most of the time you will see these services spread over multiple servers for redundancy and efficiency.

In a CA environment there is probably more than one CA server unless it is a purpose-built CA. Some VPN implementations install a single CA just to serve the VPN users.

There are four main types of CA server:

- Root CA
- Intermediate CA
- Registration authority (RA)
- Certificate revocation list (CRL)

Root CA As the name suggests, the Root CA is the foundation of the CA environment. The Root CA is the first CA server installed in a CA environment. There can be only one Root CA server. Any other CAs or certificates will ultimately have to be verified against the Root CA. A CA environment can be as simple as a single Root CA. If the environment is small enough, it may be practical to have only one CA server.

Intermediate CA Most of the time you want to have more than one CA server. A CA manages, verifies, and distributes digital certificates. What if the Root CA server is offline? The CA environment does not work, and your organization can't verify its identity to others or verify the identity of entities it communications with. So it is good practice to have more than one CA server. If we add a CA server to an existing environment, we call that CA an intermediate CA. There can be an unlimited number of intermediate CA servers.

Many private CA implementations take the Root CA offline to protect it. If the Root CA private key gets compromised, the entire PKI environment is compromised. As long as there are intermediate CA servers online, the PKI will work. The Root CA server is periodically brought back online to manage the CA environment and certificates. In most cases it's best to keep it offline.

In large CA environments, the intermediate CA servers may be purpose built to take the load off the Root CA server. The CA environment is typically a hierarchy with the Root CA at the top. The main thing to remember about the intermediate CA servers is that they are the workhorses carrying most of the load.

Registration Authority The RA server is a purpose-built server that handles certificate requests. You will see RA servers in large CA environments to take the load off the root and intermediate CA servers. The RA server cannot issue or revoke certificates, but it can verify the user's identity, passwords, and policies and determine the validity of a request. Once the RA has processed the certificate request, it will forward it to the root or intermediate CA. Sometimes the term *RA* also refers to the staff who manages certificates.

Certificate Revocation List When a certificate needs to be revoked, it is done through a CRL server. The CRL server holds a list of revoked certificates. This list is not for expired certificates but for certificates that have been compromised or deemed unnecessary. For the CRL to work, the end device has to receive the list or query the CRL to verify that the certificate is not revoked. The CRL is an important part of a PKI. Without the CRL, compromised certificates could never be removed from the infrastructure. For small environments the Root CA performs double duty and acts as the CRL server. In larger environments a dedicated CRL server is used for performance improvements.

CA Structures

In this section we will discuss and diagram the different CA structures. The two main CA structures are

- Single CA structure
- Hierarchical structure—includes a Root CA and intermediate CAs

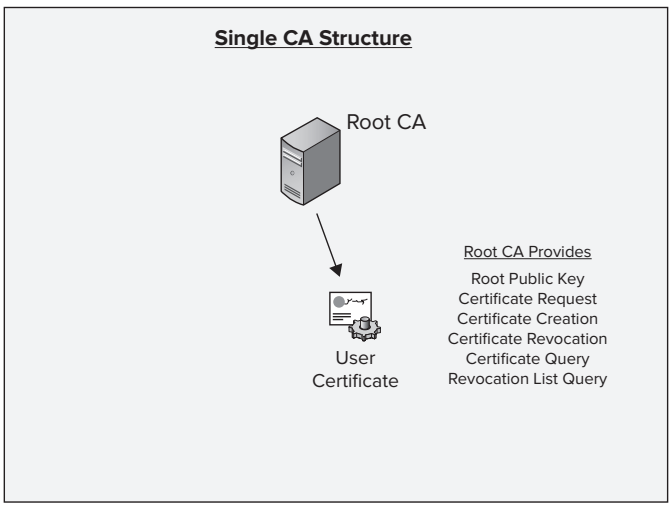
Single CA Structure

The single CA structure consists of one CA server, which is also the Root CA. In this structure, a single server provides all the CA services. We will review some of the benefits and drawbacks to the Single CA structure.

- Benefits
  - Simple to manage
  - Perfect for small environments or purpose-built scenarios
  - Centralized
- Drawbacks
  - Single point of failure
  - Not scalable
  - Affects entire PKI if the single CA is compromised

As you can see, there are some major drawbacks to a single CA structure. A single CA structure should only be used in small environments on applications that are not mission critical. Figure 11.3 displays a single CA structure.

FIGURE 11.3 Single CA structure



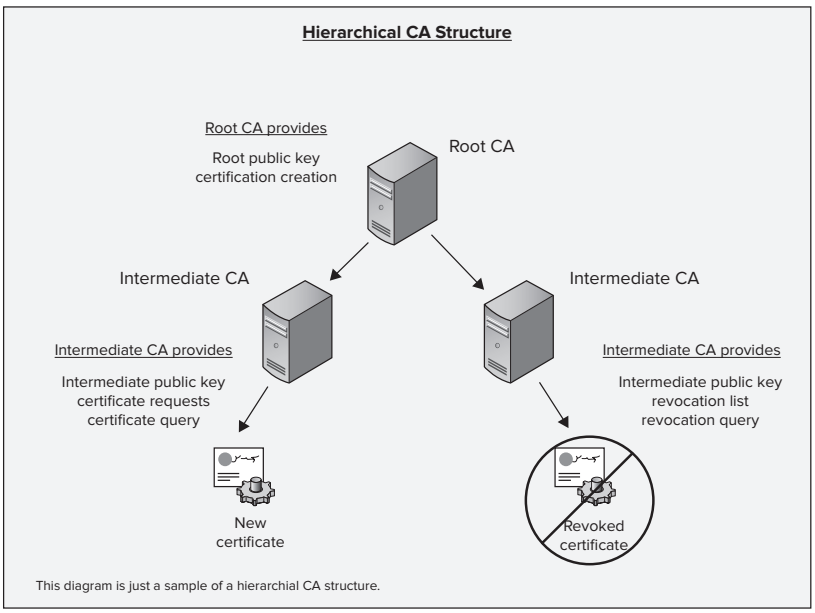
Hierarchical CA Structure

The hierarchical CA structure is the most common. It involves multiple CA servers, with the Root CA server at the top. Each server in the hierarchy specializes in providing certain services. Now we will look at the benefits and drawbacks of the hierarchical CA structure.

- Benefits
  - Distribution of the load
  - No single point of failure
  - Scalability
  - More secure (ability to take the Root CA offline)
- Drawbacks
  - Difficult to implement and manage
  - Certificate chain, determining the path of a certificate

From these benefits and drawbacks you can see that a hierarchical structure is a best practice but requires more effort. Figure 11.4 displays a hierarchical CA structure.

FIGURE 11.4 Hierarchical CA structure



CrossCertify CA Structure

The cross-certify CA structure is a proposed implementation that does not exist yet. The idea is to allow two or more PKIs to share information and verify each other's certificates. This would allow you to obtain one certificate and be verified by other PKIs. Here are a few benefits and drawbacks of cross-certify CA structures.

- Benefits
  - Cross-certify CAs are able to use certificates outside a single PKI. Root CA servers from other PKI structures are able to verify digital certificates that were not generated within their own PKI.
- Drawbacks
  - Cross-certify CA collaboration is limited. For cross-certify CA structures to work, collaboration between PKI owners is required. Many PKI owners have not opened their PKI infrastructure to other PKI owners.
  - Usage between private and public PKIs is administratively difficult. For cross-certify CA structures to work, they have to communicate with each PKI. Many private PKIs do not have a public interface, which is a security best practice. You would have to make the private PKI public for it to communicate with other public PKIs. This would defeat the purpose of a private PKI.

Digital Certificates

A digital certificate is a way to create electronic identities. These certificates are pieces of digital information that provide a way to verify who each entity is. For this to be possible there has to be an authority that controls, distributes, and verifies the identities. One of the most widely used authorities to accomplish this is a certificate authority (CA). There are many CA providers. During a normal day on the Internet you could use a dozen different CAs and not even know it. If you open your web browser options and find your certificates settings, you will see in the Root Certification Authorities section companies providing certificate authority. The only time you really get to choose a CA is when you enroll for a certificate. The CA can be public or private. VeriSign and Thawte are public CAs. Private CAs are those created in private organizations for internal use. A host of products allow for private CAs. Microsoft CA and OpenSSL are types of private CA servers.

Now that you've had a quick refresher on the CA server's role, let's see how to actually get a certificate.

Certificate Enrollment

The process for certificate enrollment depends on the device on which you are creating the certificate request. If the device is a Microsoft workstation, the procedure may be as simple as going to a website and filling in some information. Requesting a certificate on a VPN device may be a command-line request. And perhaps on a load balancer you have to use both a graphical user interface and a command-line interface for the request. You will have to know every command to input to get the request created.

Because there are so many different front ends to a certificate request, we will look at the common factors required for all certificate requests. You will need to input this information into the request no matter what the front end of the device looks like.

Certificate Request

One of the first questions you have to answer is what type of certificate you need. Some devices will not require you to answer this question because the type of certificate needed will be built into the certificate-request front end. But in other cases this is not so. At any rate, it is always good to understand that there are different types of certificates, including

- User certificate
- Server certificate
- CA certificate
- Software signing certificate

Each type of certificate may require some additional information in order to request the certificate. We will use the common server certificate request for the rest of this section.

The main area in a certificate request contains the distinguished name fields. All certificate requests require this information in some format. We will look at it in the standard Distinguished Name (DN) format.

- Organization: O = the legal name of the business. For our running case study this would be XYZ Corporation.
- Organizational Unit: OU = this field is optional. For large companies or any company that has multiple divisions or departments that want to differentiate the certificates, this is the field to use.
- Common Name: CN = fully qualified domain name (FQDN) of the site for this certificate. Remember that mail.xyzcorporationtest.com is different from www.xyzcorporationtest .com. The CN will have to match the FQDN of the site. You do not need to worry about anything after the root like .com or .net; www.xyzcorporationtest.com is the same as www.xyzcorporationtest.com/news from the CN perspective.
- Country: C = twoletter abbreviation for the country the business is registered in.
- State/Providence: ST = the state or providence for the business. Do not abbreviate this field because the certificate may be used around the world and other countries may not know what the abbreviation means.
- City/Locality: L = the city or locality of the business. Once again, do not abbreviate this field.

An important part of every certificate is the key size. In today's world we recommend that the key size be at least 2048 bits. The key size is typically a multiple of 512. With any key, the larger the key, the stronger the security but the longer it takes to process data with the key.

Another field that you may have to fill in is the hash algorithm. The type of application you will be using will determine the hash algorithm. Hash algorithms were discussed extensively in Chapter 10. In most server certificates on the Web, the SHA-1 hash is used.

Finally, you need to decide the format the request will be in. Depending on your CA, you may have to choose a specific format type. A couple of format types are

- Certificate management messages over CMS (CMC), described in RFC 5272
- PKCS #10, part of the PKCS standard, described later in this chapter

Certificate Classes

We have discussed the information needed to generate a certificate request. There may be other information needed that is not included in the digital request. This information may be an email address, government ID, company letterhead, or an in-person verification with the CA. The necessity for further verification of an individual or entity is determined by the certificate class.

There is no real standard for certificate classes yet. VeriSign was the originator of the concept and has created six classes for digital certificates.

- Class 0 = No verification
- Class 1 = Individual Certificate, meant for digital signing of email

- Class 2 = Business Certificate, proves the certificate belongs to the company
- Class 3 = Server and Software Signing, used to verify that a server or software program is provided by a verified source
- Class 4 = Business to Business, specific certificates for communication between two known businesses
- Class 5 = Private or Government, for use in private or government PKI environments

Each of these classes has different verification requirements. For example, Class 1 certificates require email verification to be sent and replied to. A Class 3 certificate may require multiple verifications to prove a company's name, address, and ownership.

Certificate Enrollment Process

Once we have all this information gathered, we can move forward with the actual certificate request. We will take the information and fill in the request from the device. As previously mentioned, the device could be a website, GUI, or command-line interface.

When we request a certificate, we are generating both a private and a public key. The reason we need to perform the request from the device is to protect the private key. If we request the certificate directly from the device, we will not have to copy or move around the private key and expose it to unnecessary risk.

Some devices have a means to automatically send the certificate request to a CA. This is possible, for example, in some AD environments. Other devices have an option to save the certificate request so you can manually import into a CA system. And still other devices such as routers give you the option of copying the request from the command line so you can paste it into a CA system.

Once we have created the request and sent it to a CA, we need to wait for the actual certificate. Depending on the CA, the certificate may be automatically issued or it may require further verification. Some CAs may require physical verification, as previously mentioned in the "Certificate Classes" section.

Once the CA has issued the certificate, it typically sits in a repository until it is retrieved. Most CAs have a website where you can download your certificate. Some devices and protocols offer the ability for the device to occasionally check the repository for the certificate. In this case, if the certificate is available, the device will download it and install it for use.

Digital Certificates Exposed

What does the digital certificate contain, and what does it look like? In this section we will explore the contents of a digital certificate and show you screen shots of a digital certificate. One thing to understand is that we will view the certificate in a few different formats. For this section we are using the Windows built-in viewer to examine the certificates.

The first thing we will look at is the content of a digital certificate. We will explain and provide images of each of these fields:

- Certificate Information
- Details

- Version
- Serial Number
- Signature Algorithm
- Issuer
- Validity
  - Valid From
  - Valid To
- Subject
- Public Key
- Extensions

- Certification Path

Certificate Information

When we open the certificate to view its contents, the first thing we see is the Certificate Information screen of the General tab, as shown in Figure 11.5. Our example certificate is a Root CA certificate built into Microsoft Internet Explorer.

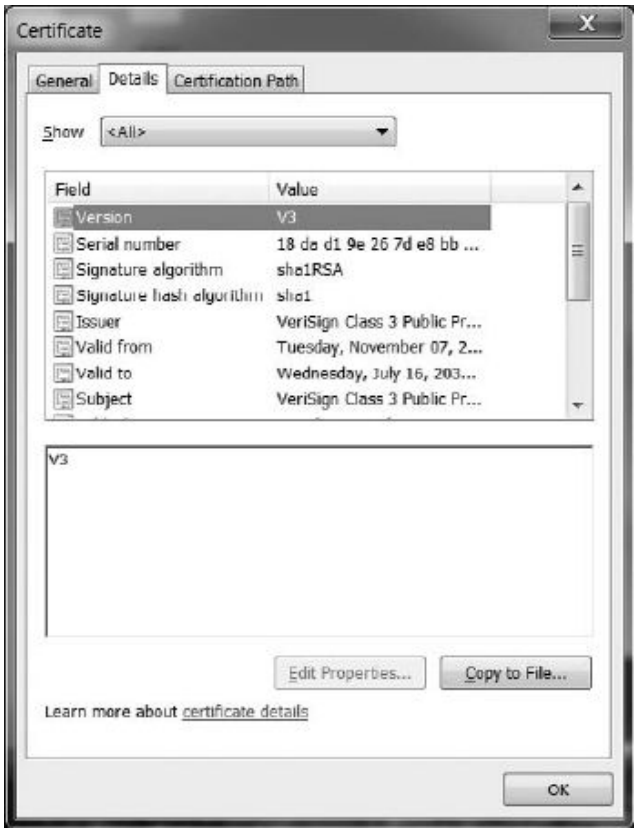
FIGURE 11.5 Certificate General tab

At the top of the screen is the purpose(s) of the certificate. This certificate has multiple purposes, including to ensure identity, protect email, and ensure software. Below this area is the Issued To and Issued By information. Finally, we see the Valid From and To dates. Some of this information is available in the Details section also, but it is provided on the main page for summarization.

Version Field

On the Details tab, look at the Version field (Figure 11.6), which specifies which version of digital certificate formatting was used in the creation of the certificate. Currently there are three versions: 1, 2, and 3. V3, shown in the figure, is the most common.

FIGURE 11.6 Certificate Version field



Serial Number Field

The Serial Number field is a way to identify a specific certificate. If the certificate is in question, you can use the serial number to verify that you have the proper certificate. Figure 11.7 displays the Serial Number field.

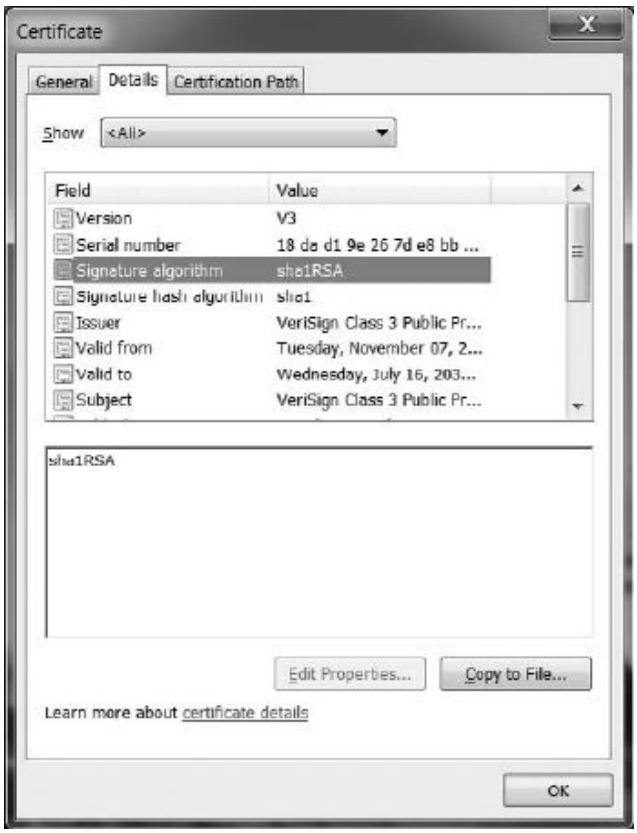
FIGURE 11.7 Certificate Serial Number field

The signature is a hexadecimal value of 32 characters. Each CA uses different values for the serial number. Serial numbers may be as short as 6 hexadecimal characters.

Signature Algorithm and Signature Hash Algorithm Fields

The Signature Algorithm field specifies which algorithm is to be used when using this certificate. Figure 11.8 displays the Signature Algorithm field.

FIGURE 11.8 Certificate Signature Algorithm field



The Signature Algorithm field in our example shows the value sha1RSA. This means that the hash is SHA-1 and the public key cryptography is RSA. Also note that below the Signature Algorithm field is a Signature Hash Algorithm field. This lists sha1.

Issuer Field

The issuer is the creator of the certificate. Figure 11.9 displays the Issuer field.

One thing you will notice in the Issuer field is some letters followed by the = sign. These letters are called distinguished names (DN), and they are used in the creation of digital certificates. It is important that you get to know what the DNs are. DNs are used not only in digital certificates but in LDAP and AD as well.

FIGURE 11.9 Certificate Issuer field

It is good to know that different protocols such as LDAP and AD may have different meanings for DNs. I have seen tools that query with DNs and require different DNs than the corresponding protocol they are querying.

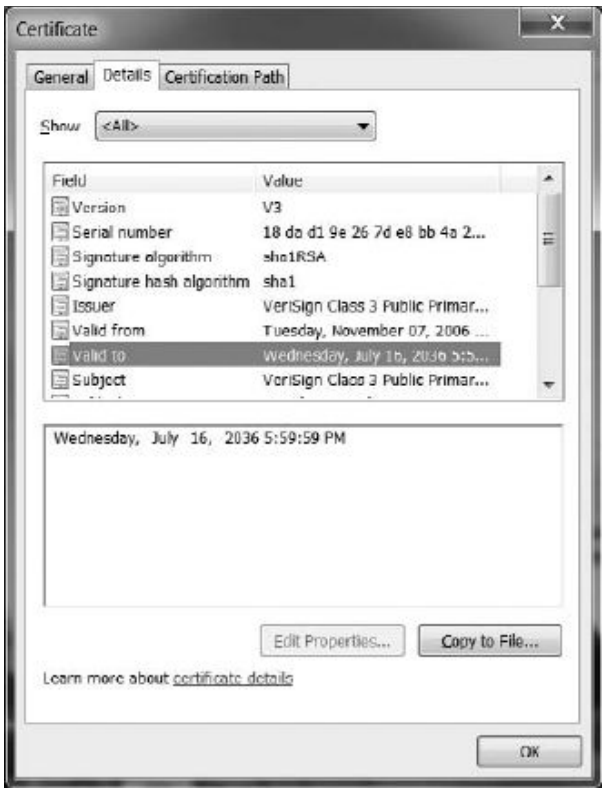
Here are a few of the most common DNs used:

- CN = Common name
- OU = Organizational unit
- O = Organization
- C = Country

Validity Fields

The validity fields determine the time period during which the certificate is valid. The Valid From field displays the start date of the certificate. Most of the time, it will be when the certificate was created. The Valid To field shows how long the certificate can be used. This can be any amount of time from 1 day to 100 years. Most certificates are valid for 1 year to 10 years. Figure 11.10 displays the Valid To field.

FIGURE 11.10 Certificate Valid To field



The certificate displayed is a Root CA certificate. It is valid from Tuesday, November 07, 2006, to Wednesday, July 16, 2036. The example certificate is valid for almost 30 years—a little long if you ask me. (The longer the certificate is valid, the more time a hacker has to compromise the certificate.)

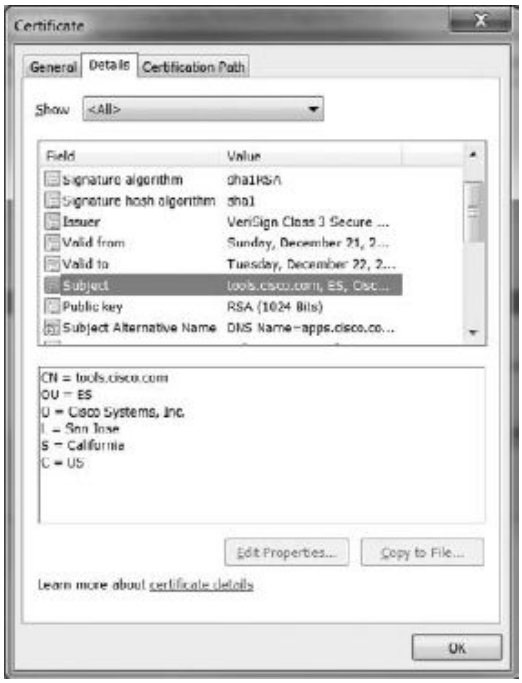
Subject Field

The Subject field describes the entity that is associated with the public key store. Figure 11.11 displays the Subject field.

In our example, the Subject field is identical to the Issuer field. This is because the certificate is a Root CA certificate. With other kinds of certificates, the Subject field will display information specific to the company that requested the certificate. Figure 11.12 displays the Subject field of a certificate requested by Cisco. In this example, the Subject field is different from the Issuer field. Most certificates you come across will be this way.

FIGURE 11.11 Certificate Subject field—Root CA

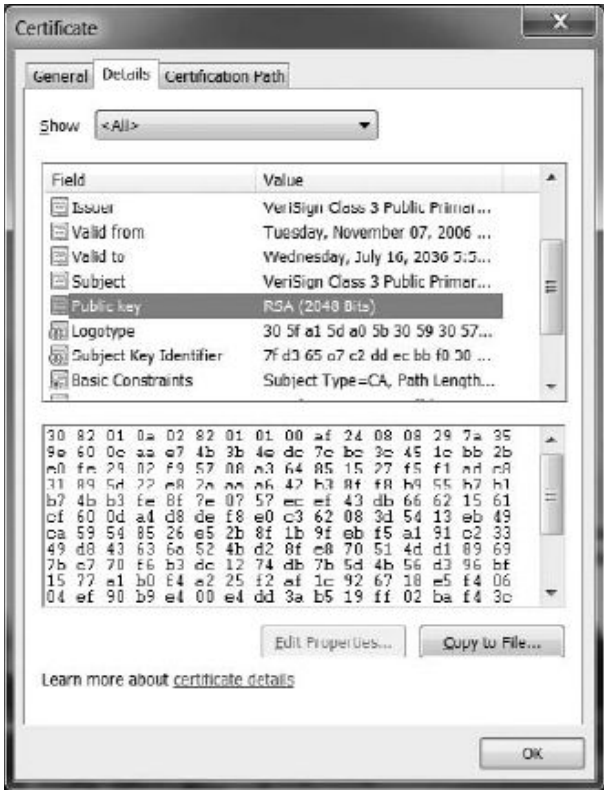
FIGURE 11.12 Certificate Subject field—server CA



Public Key Field

One of the most important fields of a digital certificate is the Public Key field. The Public Key field, as its name suggests, holds the public key. Figure 11.13 displays the Public Key field.

FIGURE 11.13 Certificate Public Key field



The public key for this certificate is a 2048-bit key. You can copy the public key to a file if you wish.

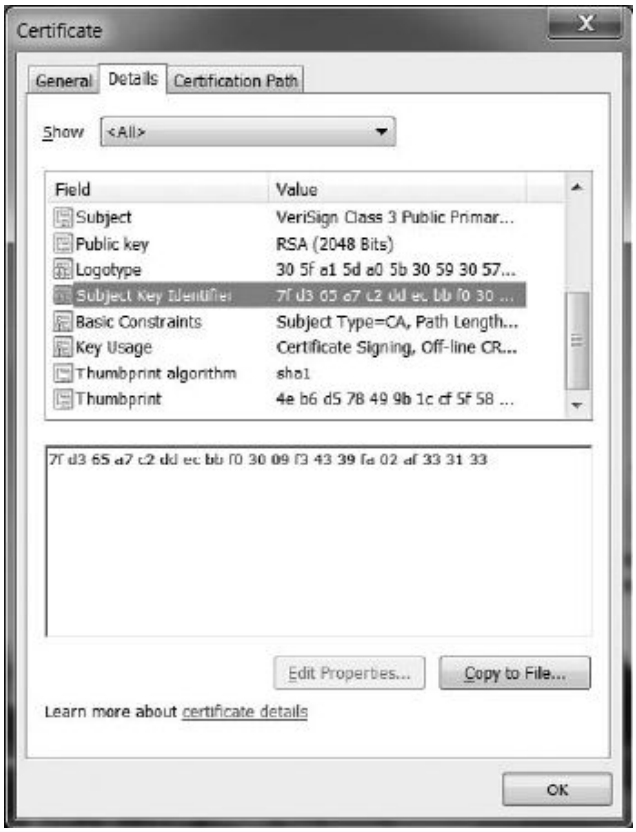
Extension Fields

Extension fields are available only in X.509 version 3. The extension fields provide a way to add more data to the certificates. There are two types of extensions:

- **Standard extensions** include Subject Key Identifier, Key Usage, and Basic Constraints.
- **Private Internet extensions** include the Authority Information Access and Subject Information Access extensions.

Figure 11.14 displays some extension fields. You will notice the Subject Key Identifier, Basic Constraints, and Key Usage extension fields, among others. The icon next to the extension fields is white rather than orange and brown for the primary fields. Details about these fields are outside the scope of this book, but it is good to know of their existence.

FIGURE 11.14 Certificate extension fields



RFC 5280 is a good place to view all the available extension fields and their descriptions.

Certification Path

The Certification Path tab displays the hierarchy of the certificate. Figure 11.15 displays the certification path information for a Root CA certificate.

Notice that a Root CA certificate has only one certificate in its certification path.

Now let's look at the certificate from Cisco again. Figure 11.16 displays the certification path for the tools.cisco.com certificate.

FIGURE 11.15 Certification path of a Root CA

FIGURE 11.16 Certification path from the server

Now we have three certificates in the Certification Path section. The hierarchy is from top to bottom. The Root CA certificate is on top; this means it is the authority. The certificate in the middle is an intermediate CA certificate. Finally, the tools.cisco.com certificate is listed.

These images are taken from a Windows PC. If you click one of the certificates in the Certification Path field, you can view the actual certificate in another window by clicking the View Certificate button.

Certificate Usage

You know what a certificate is and how you can obtain one. Now you need to know what to use it for. Certificates can be used for many things, including secure web transactions, device administration, user authentication, secure email, and even virtual dating.

We will discuss three uses for digital certificates:

- Identity
- Secure communication
- User authentication

Identity

What if you made some copies of your driver's license and carried them around with you? Anytime you wanted to do business with someone or a company, you would give them a copy of your driver's license. You are probably thinking that is crazy. Well is it, really? We do this all the time; the only difference is that we typically do not make the copies ourselves. Let's say you want to withdraw money from your bank and you have mislaid your ATM card. You go inside and fill out a withdrawal slip. You then proceed to a teller. If it is a decent bank, one of the first things the teller will do is ask you for your ID. If there were a good way to make a photocopy tamper proof, we could just give the teller a copy of our ID. When you test-drive a car, they typically make a copy of your license.

A digital certificate is like a tamper-proof photocopy. When we want to do business online with either a corporation or person, we can hand them our certificate. They take our certificate and verify it with the public key that is contained within the certificate and the public keys of the CA servers contained within their certificates.

Thus, we can use a certificate to verify an entity's identity virtually. Think of virtual dating and how people may alter the truth about themselves. A digital certificate that had to be verified against a person's driver's license and credit card before it was issued would be better than no certificate.

Secure Communication

Now that we know we can use a certificate to verify identity, let's take it one step further. We want to buy another Wiley book from our favorite website. We open our web browser and enter the address for the site. We find our next book and add it to our shopping chart. We finish shopping and want to check out. When we click the checkout button we are redirected

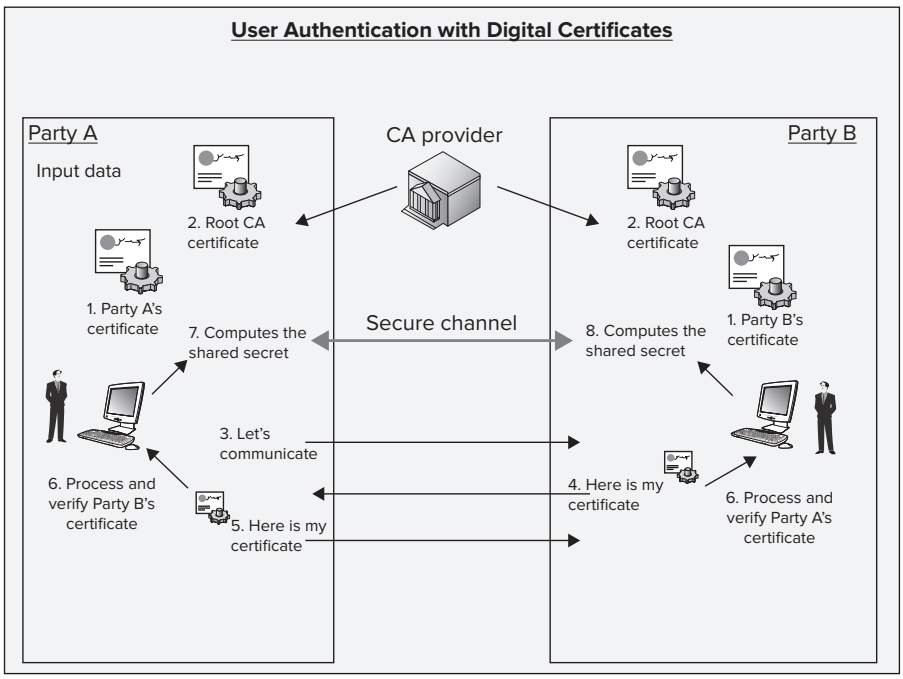
to a secure website. If your favorite website is a reputable one, this should have been somewhat transparent to you. But in the background a digital certificate was acquired and verified. Your browser has verified the identity of the website. Once the verification is completed, the web browser has a trust relationship with the website. Now your browser takes the public key in the certificate and encrypts a message to the website. The communication between the website and your browser is secured using the public and private key exchange we discussed earlier in this chapter. If the certificate was created with the RSA algorithm, then the RSA process will be used for the communication between the two parties. Once you have finished purchasing your book, you can close your web browser and terminate the connection to the website.

User Authentication

We have discussed identity verification and secure communication with certificates. Another use for digital certificates is user authentication. You may see this with VPN solutions, both remote-access and site-to-site VPNs. Without using a digital certificate for authentication, the VPN connection will require a shared secret key. We have discussed the limitations of a manual shared secret key throughout this chapter and in Chapter 10. In a large deployment, a shared secret key is not flexible. If it is compromised, it may take days or weeks to change the shared secret keys on all the devices.

To get around the shared secret key issue, we can use digital certificates. We will use the certificates to verify the identity of each party. We will then use the keys in the certificates to calculate a temporary shared secret key that is used for the symmetric algorithm. Figure 11.17 shows the process for user authentication with digital certificates.

FIGURE 11.17 User authentication process



We will look at this process a little deeper.

1. Party A and Party B both have certificates issued from the same CA.
2. Party A and Party B both have the Root CA certificate installed locally.
3. Party A sends a request for communication to Party B.
4. Party B sends Party A its certificate.
5. Party A sends Party B its certificate.
6. Each party will process the other's certificate with the stored Root CA certificate's public key.
7. Party A computes a shared secret between the two parties.
8. Party B computes a shared secret between the two parties.

Many times two different communication sessions will be created—one channel from Party A to Party B and one channel from Party B to Party A. The reason for the two secure channels is that Party A will use the public key from Party B for encryption, and vice-versa. It is more secure to decrypt with a private key than it is with a public key. Only one entity has the private key. Anyone can obtain the public key. This could allow anyone to feasibly decrypt the information if it was encrypted with the private key.

Real World Scenario

Digital Certificates as Part of a Two-Factor Authentication Solution

The XYZ Corporation has had security and management issues with their remoteaccess VPN solution. Your boss has tasked you with providing a solution for more than 500 remote users. You decide it would be best to implement a private PKI environment dedicated to the VPN users. The XYZ Corporation has a Microsoft infrastructure and is running AD. You decide to install and configure Microsoft CA to provide digital certificates for all the remote VPN users and the VPN appliance they will terminate to. This solution will require the users to authenticate via an individual certificate instead of a secret key shared by all 500 users.

As a security professional, you understand that with any security control, a single layer of protection is never enough. Digital certificates in a remote-access VPN environment provide one type of protection, but two-factor authentication—a way to require at least two means of authentication—should be used with any remote authentication. You might decide to accomplish this with a one-time password (OTP) that expires in a few seconds or a hardware key that is plugged into the device.

Certificate Limitations

You may have already picked up on some of the limitations of digital certificates throughout the chapter. The three main limitations of digital certificates are the following:

- They require a CRL server for revoked certificates.
- The private key is the heart of the security.
- Human interaction is necessary to administer certificate requests.

We will look at these three limitations briefly.

They require a CRL server for revoked certificates. The certificate must have a CRL, and the CA must have a CRL server available. The device must be able to obtain a new CRL periodically to ensure that certificates have not been revoked. Many times you will see PKI implementations without a CRL server or the CRL is offline. Without a CRL server, there is no way to verify that a certificate is still good.

The private key is the heart of the security. If the private key is compromised, all corresponding certificates are compromised. If the compromised private key is from a server certificate, a new certificate is required from the CA. If the compromised private key is from a CA, the CA and all corresponding certificates generated with that private key have to be replaced. You can see the pattern, right? If the private key to the Root CA is compromised—you guessed it—the entire PKI environment is compromised, and all certificates have to be replaced. Wow! Protect those private keys.

Human interaction is necessary to administer certificate requests. When we discussed the certification classes, we stated that some certificates require further verification from an individual or corporation. We also mentioned that some CA servers require an administrator to approve the certificate request and revocation.

A busy IT department may overlook revoking certificates and delay approving certificate requests. A staff member at the CA may have a bad day and process a certification without verifying a company's information. Because humans are involved, the human factor is a limitation of the PKI solution.

PKI Standards

There are many standards, vendors, protocols, and technologies within a PKI. We will look at a few standards that help define some of the most-used technologies in the PKI environment. These standards include the Public Key Cryptography Standards (PKCS), X.509, and the Simple Certificate Enrollment Protocol (SCEP).

Public Key Cryptography Standards

The Public Key Cryptography Standards is a set of standards developed and administered by RSA Security. If you remember from our earlier discussions on the RSA algorithm,

RSA Security was founded by the three inventors of the RSA algorithm. RSA Security was purchased by EMC in 2004 and is now called RSA, The Security Division of EMC Corporation.

The PKCS is a set of standards that was first published in 1991. It was originally developed to deal with the growing technology of public key cryptography. The PKCS was originally comprised 15 different standards. There are currently 13 standards, as two of the standards have been deprecated. Each standard represents a different technology specific to public key cryptography. The names of the 15 standards all start with the letters *PKCS* and end with the corresponding number, such as PKCS #1 and PKCS #3. Table 11.6 lists the 13 active standards. The most important standards are in bold.

TABLE 11.6 PKCS Standards

Standard Number	Standard Name
PKCS #1	RSA Cryptography Standard
PKCS #3	Diffie-Hellman Key Agreement Standard
PKCS #5	Password-Based Cryptography
PKCS #6	Extended-Certificate Syntax Standard
PKCS #7	Cryptographic Message Syntax Standard
PKCS #8	Private-Key Information Syntax Standard
PKCS #9	Selected Attribute Types
PKCS #10	Certification Request Syntax Standard
PKCS #11	Cryptographic Token Interface Standard
PKCS #12	Personal Information Exchange Syntax Standard
PKCS #13	Elliptic Curve Cryptography Standard
PKCS #14	Pseudo-Random Number Generation (under development)
PKCS #15	Cryptographic Token Information Format Standard

PKCS #2 and #4 have been withdrawn. They are incorporated in PKCS #1. If you wish to read more on the PKCS standards, please visit `http://www.rsa.com`.

After looking at the table of PKCS standards, you should be able to identify some of the standards by the descriptions. RSA and Diffie-Hellman, were discussed in Chapter 10 and at the beginning of this chapter. We will take a closer look at some of the widely used PKCS standards—PKCS #1, #10, and #7.

PKCS #1 Standard

The PKCS #1 standard defines RSA cryptography standards that should be followed when the RSA algorithm is used. The standard is also described in RFC 3447 for version 2.1, published in 2003.

The PKCS #1 standard covers four main topics:

- Cryptographic primitives
- Encryption schemes
- Signature schemes with appendix
- ASN.1 syntax for representing keys and for identifying the schemes

We touched on most of these topics at the beginning of this chapter. The main thing we want to look at from these topics is the ASN.1 topic.

You need to first understand what ASN.1 is. ASN.1 is a standard that defines data structures, coding, and transmission of data. It is a standard to follow when creating information, in this case keys for RSA. An example of this is the description of the private key structure taken from RFC 3447.

The following copyright statement is on the document:

> This document and translations of it may be copied and furnished to others provided that the above copyright notice and this paragraph are included on all such copies. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as required to translate it into languages other than English.

An RSA private key should be represented with the ASN.1 type RSAPrivateKey:

```
RSAPrivateKey ::= SEQUENCE {
          version           Version,
          modulus           INTEGER,  -- n
          publicExponent    INTEGER,  -- e
          privateExponent   INTEGER,  -- d
          prime1            INTEGER,  -- p
          prime2            INTEGER,  -- q
          exponent1         INTEGER,  -- d mod (p-1)
          exponent2         INTEGER,  -- d mod (q-1)
          coefficient       INTEGER,  -- (inverse of q) mod p
          otherPrimeInfos   OtherPrimeInfos OPTIONAL
      }
```

You should notice some very similar names and letters in this structure, namely, p, q, and d. If you go back to the "RSA Algorithm" section earlier in this chapter, you will see the same letters. This PKCS standard outlines how to create the data structure so the information for p, q, and d can be transmitted consistently.

PKCS #10 Standard

The PKCS #10 standard defines the Certification Request Syntax Standard. This standard is also specified in RFC 2986. The main purpose of PKCS #10 is to create a standard for digital certificate requests. When a party requests to create a digital certificate, it may be created in the PKCS #10 format. The PKCS #10 standard uses ASN.1 to structure the request.

Most of the time when you deal with a digital certificate request it will be displayed in a Base64 encoding. Following is an example:

```
-----BEGIN CERTIFICATE REQUEST-----
kdaf24928akasdf92axks2kdkoISODK9
etc...
-----END CERTIFICATE REQUEST-----
```

PKCS #10 specifies in the standard what the ANS.1 structure should follow. A sample of that structure comes from RFC 2986.

The following copyright statement is on the document:

> Copyright© The Internet Society 2000. All Rights Reserved.
>
> This document and translations of it may be copied and furnished to others provided that the above copyright notice and this paragraph are included on all such copies. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as required to translate it into languages other than English.

```
Attribute { ATTRIBUTE:IOSet } ::= SEQUENCE {
         type   ATTRIBUTE.&id({IOSet}),
         values SET SIZE(1..MAX) OF ATTRIBUTE.&Type({IOSet}{@type})
    }
```

As you can see, again the structure is very specific. If you took the Base64 certificate request and turned it into the ASN.1 format, you would see the previous data in the output.

PKCS #7 Standard

The PKCS #7 standard defines the Cryptographic Message Syntax Standard. This standard is also specified in RFC 2315. One of the main purposes of PKCS #7 is to describe a standard for transferring certificates. One example of that is that the response to a PKCS #10 request will follow the PKCS #7 standard. The actual PKCS # 10 request will be enveloped in PKCS #7 and sent to the CA.

PKCS #7 was also the basic format used in the Secure/Multipurpose Internet Mail Extensions (S/MIME). S/MIME is a PKC for signing and encapsulating e-mail.

X.509 Standard

The X.509 standard originated in the X.500 standards by the International Telecommunication Union (ITU-T). X.509 describes standards for digital certificates, certificate revocation lists (CRLs), and the certification path validation algorithm.

The current version of X.509 is version 3 and is specified in RFC 5280.

X.509 Specification of Digital Certificates

The first thing we will look at in X.509 is digital certificates. The following are the basic fields for a digital certificate following the X.509 v3 standard:

- Version
- Serial Number
- Signature Algorithm
- Issuer
- Validity
  - Valid From
  - Valid To
- Subject
- Public Key
- Extensions (X.509 v3)

We covered these fields in the "Digital Certificates Exposed" section earlier in this chapter.

X.509 Specification of CRLs

Now we will look at X.509 and how the standard affects CRLs. The main thing to understand about the CRL standard in X.509 is that it outlines the data structure for the CRL communications and messages. This structure uses ASN.1 and is much like the PKCS standard structures discussed earlier.

X.509 Specification of the Certification Path Validation Algorithm

Lastly, we will look at X.509 and the certification path validation algorithm. Certification path validation is a means to verify the certificate hierarchy. Most certificates have a Root CA, intermediate CA, and device/user certificate associated with them. The certification path validation procedures are used to verify that the path between the device/user certificate and the Root CA certificate is valid. Figure 11.18 shows a certification path for the tools.cisco.com certificate.

FIGURE 11.18 Certification path of the server

Certification path validation works by means of a *trust anchor* . The trust anchor is a CA that is part of the CA PKI environment. The top CA that issued the certificate or a dedicated intermediate CA can be used for path validation. The CA's public key, name, and any constraints are used for the validation process. Name, policy, and basic constraint fields are verified to ensure the validation does not violate any of the constraint field requirements.

The certification path validation process is linear. It starts with the device/user certificate and verifies it against the trust anchor. It then moves up to the next certificate in the path until it reaches the certificate at the top of the hierarchy, which should be a Root CA.

X.509 Specification of File Extensions

Another important thing to understand about X.509 is the file extensions. When you deal with digital certificates, you will come across different file extensions. Many times these file extensions are confusing. This section will shed some light on the common file extensions associated with X.509. Table 11.7 describes X.509 file extension types.

TABLE 11.7 X.509 File Extension Types

File Extension	Description
.CER, .DER	Defined in the ITU-T x.690 standard. Formatting uses ASN.1 structures. CER stands for Canonical Encoding Rules, and DER stands for Distinguished Encoding Rules. Both are based on BER (Basic Encoding Rules).

.PEM	Privacy Enhanced Mail (PEM) was invented by the IETF for securing email. It uses Base64 encoding. The PEM file starts and ends with
	`----BEGIN CERTIFICATE-----`
	`-----END CERTIFICATE-----`

.P7B, P7C	Encoding is similar to Base64. Files with these extensions contain PKCS #7 certificates. They do not contain data.
.P12, PFX	These files contain PKCS #12 certificates and keys. PFX is the predecessor to P12.

Figure 11.19 shows a sample certificate request in Base64 formatting. Depending on whether the file is a certificate request or a completed certificate, the beginning and ending tags may be different. In a certificate request the tags will state "`BEGIN NEW CERTIFICATE REQUEST`"and "`END NEW CERTIFICATE REQUEST`."

FIGURE 11.19 Certificate request

SCEP Standard

Simple Certificate Enrollment Protocol (SCEP) is a standard that Cisco invented as a means to issue certificates to network devices. This process is automatic and is intended for a large number of devices. The specifications for SCEP are outlined in IETF draft-nourse-scep-19.

SCEP supports four main functions:

- Certificate enrollment
- Public key distribution
- Certificate query
- CRL query

Certificate Enrollment

As the name implies, the SCEP certificate enrollment process is supposed to be simple. The device creates a certificate request via PKCS #10. The device sends the request in a PKCS #7 package. Depending on how the CA server is set up, it may automatically issue a certificate. If not, an administrator will have to issue the certificate manually. Some SCEP systems require a password. This is configurable in the requesting device.

Once a certificate is issued, it is sent back to the device in a PKCS #7 package. The device will receive the certificate and install it into the system.

Microsoft CA has an add-on for IIS to enable SCEP. Once SCEP is enabled, devices can register for a certificate over HTTP. You point the device to the HTTP address of the CA server. The devices will automatically request a certificate. If a certificate is issued, it will be sent back to the device. Administrators can control whether the certificate is automatically sent or an approval is needed before releasing the certificate.

Public Key Distribution

Public key distribution is used to transfer public keys to devices that have not already received a public key from the CA. This allows the device to gain access to CA and RA root public keys.

Certificate Query

A certificate query is a request for a certificate from either a CA or an LDAP server. If the request is to a CA server, the device sends a request to the CA server with the issuer name and serial number of the certificate. This assumes that the devices previously had the issuer name and serial number. If not, the device is unable to query the CA for the certificate.

CRL Query

The CRL query allows the device to query for a current CRL list. SCEP does not allow for a push of the CRL. The device has to query for a new CRL. The CRL query mechanism does not scale well and is not typically used. CRL Distributed Points (CDP) is typically used when possible.

Summary

In this chapter you learned the foundation of asymmetric encryption. We discussed that asymmetric encryptions have a place in the encryption world. You also learned the limitations of asymmetric algorithms because of the large computations required to process the request. You learned how asymmetric algorithms arrive at a given output via the mathematics behind the algorithm.

After asymmetric encryption you learned about the PKI. You learned that a PKI has to have authority for it to work properly. Components of a PKI include the CA, RA, CRL, and intermediate CA servers. The first CA server is called a Root CA.

You learned that digital certificates are similar to personal IDs such as driver's licenses and social security cards. You learned how to request a certificate and what to do with the certificate once you receive it. You learned the fields within a certificate and why certain fields are important.

Finally, we discussed the PKI standards that govern the PKI and digital certificates. These standards included the Public Key Cryptography Standards, X.509, and the Simple Certificate Enrollment Protocol.

Exam Essentials

Understand the differences between asymmetric and symmetric encryption. Asymmetric encryption utilizes a public key and a private key, and symmetric encryption utilizes a shared secret key. Asymmetric encryption is much slower than symmetric encryption.

Remember the differences between a public key and a private key. The private key is generated during the certificate request and has to remain secure to provide security. The public key is assumed to be available to everyone and has no security.

Understand the two features public key cryptography provides. The two features are confidentiality and authentication. These features are provided by the key used to start the encryption. The public key provides authentication, whereas the private key provides confidentiality.

Understand the RSA and Diffie-Hellman algorithms and how they determine the public and private keys. With the RSA algorithm, the main thing to remember is the makeup of the public and private keys: p and q are the prime numbers, e is public, and d is private. With the Diffie-Hellman algorithm, one prime number is used with one primitive number. The primitive number is typically 2 or 5. Remember that p is prime and g is primitive.

Remember the components of a PKI. The two components of a PKI are certificate authorities and digital certificates. The certificate authorities provide digital certificates for virtual identification.

Understand the certificate enrollment process and different classes available for each certificate. Be sure you understand how to obtain a certificate. Remember that there are automatic ways and manual ways to obtain a certificate. Now that there are different types of classes for each certificate, you should focus on classes 1 and 3.

Understand the different fields present in a digital certificate. Make sure you understand what the main certificate fields consist of. The main fields include issuer, subject, validity, and public key. Understand that the public key is part of the digital certificate.

Remember the PKCS standards and what each standard represents. It would be good to remember all 13 active standards, but the major standards are #1, #7, and #10. #1 is for RSA, #7 is for sending certificates, and # 10 is for requesting certificates.

Remember that the main standard that governs digital certificates is X.509. The X.509 standard governs the digital certificate's process, structure, exchange, certificate path validation, and revocation.

Written Lab

Write the answers to the following questions:

1. What are the key lengths for DH?
2. Can the public key encrypt or decrypt data?
3. What type of number(s) does the RSA algorithm use to start the key process?
4. What type of CA server is needed to guarantee that bad certificates are removed from use?
5. Which PKCS standard is used in the transmission of digital certificates?
6. What does PKI stand for? What is the purpose of PKI?
7. What is the most common digital certificate class?
8. What is the latest version for digital certificates in the X.509 standard?
9. What are the two common values for g in the DH algorithm?
10. Which encryption technique is considered stronger, asymmetric or symmetric?

Hands-on Lab

Handson Lab 11.1: View the Content of Root CA Certificates

1. Open your web browser, either Internet Explorer or Firefox.
2. Find the digital certificates.

   From Internet Explorer:

   A. Find the Internet Options settings: Tools➢ Internet Options.
   B. From the General tab move to the Content tab: General➢ Content.
   C. Click the Certificates tab.
   D. You will notice a few tabs under the Certificates window. We want to look at the Trusted Root Certification Authorities. Click the Trusted Root tab.

   From Firefox:

   A. Open your Firefox web browser.
   B. Find the Options settings: Tools➢ Options.
   C. From the Main tab click the Advanced tab: Main➢ Advanced.
   D. Click the View Certificates tab.
   E. Click the Authorities tab.
3. You should see several Root CA certificates. Click any of the certificates.

   This should look very similar to the figures we discussed in the "Digital Certificates" section. This is a good way to get familiar with digital certificates.
4. Browse through the other Certificate tabs to see if you have other certificates loaded on your computer.

Review Questions

1. What minimum key length is recommended when implementing asymmetric encryption?
   - **A.** 768
   - **B.** 1024
   - **C.** 2048
   - **D.** 4096
2. Who has possession of the private key?
   - **A.** Everyone
   - **B.** Generator of the certificate
   - **C.** CA
   - **D.** Requestor of the certificate
3. What are the drawbacks of asymmetric encryption? (Choose two.)
   - **A.** Speed
   - **B.** Key length weakness
   - **C.** Expense
   - **D.** Lack of use
4. What is the main use for asymmetric encryption?
   - **A.** Encrypting large amounts of data
   - **B.** Generating shared secret keys
   - **C.** Encrypting images
   - **D.** VPN data
5. Who and what possess the public key?
   - **A.** Digital certificate
   - **B.** Holder of the private key
   - **C.** Anyone who requests it
   - **D.** All of the above
6. Which are examples of asymmetric encryption algorithms? (Choose two.)
   - **A.** AES
   - **B.** CIA
   - **C.** DH
   - **D.** RSA

7. What are some uses for digital certificates? (Choose two.)
   - **A.** Provide a degree
   - **B.** Verify identity
   - **C.** User authentication
   - **D.** Generate images

8. What protocol allows for the automatic enrollment of a digital certificate request?
   - **A.** Simple Certificate Enrollment Protocol
   - **B.** Simple Request Enrollment Protocol
   - **C.** Safe Certificate Enrollment Protocol
   - **D.** Same Certificate Enroll Plan

9. What is the most widely used standard for digital certificates?
   - **A.** X.500
   - **B.** PKCS
   - **C.** X.509
   - **D.** SCEP

10. If the private key on the Root CA is compromised, what devices have to have their certificate replaced? (Choose all that apply.)
   - **A.** Root CA server
   - **B.** PC with user certificate
   - **C.** Cross-Certify CA
   - **D.** Intermediate CA
   - **E.** CRL server

11. How many certificates are involved in the user authentication process?
   - **A.** One
   - **B.** Two
   - **C.** Three
   - **D.** Four
   - **E.** It depends

12. How many prime numbers are used in the Diffie-Hellman algorithm?
   - **A.** One
   - **B.** Two
   - **C.** None
   - **D.** Five

13. What is the key length of Diffie-Hellman Group 2?
    - **A.** 1024-bit
    - **B.** 1536-bit
    - **C.** 2048-bit
    - **D.** Variable

14. What components are required for a PKI to be successful? (Choose all that apply.)
    - **A.** Army
    - **B.** Trusted third party
    - **C.** Authority
    - **D.** Secrecy

15. What are some drawbacks to a single server CA structure? (Choose two.)
    - **A.** Single point of failure
    - **B.** Ease of administration
    - **C.** Not scalable
    - **D.** None of the above

16. What two technologies make up a PKI?
    - **A.** Digital certificates
    - **B.** Certificate authorities
    - **C.** Digital signatures
    - **D.** Certificate authenticators

17. What fields in a certificate request should not be abbreviated? (Choose all that apply.)
    - **A.** State
    - **B.** Address
    - **C.** City
    - **D.** Country

18. The Subject field of a certificate contains what information?
    - **A.** Company information
    - **B.** Usage information
    - **C.** Website name
    - **D.** RA information

19. What protocol is considered a hybrid encryption protocol?

A. SSE

B. SNMP

C. SMTP

D. SSH

20. What are some types of CA servers in a PKI environment? (Choose all that apply.)

A. Root CA

B. CRL

C. RA

D. ABL

Answers to Review Questions

1. C. 2048 should be the minimum key length used. 1024 is considered too weak for computing power today.

2. D. The requestor is ultimately the holder of the private key. A CA does have a private key but would have to request it. The best answer is D.

3. A, B. The two main drawbacks to asymmetric encryption are the slowness to compute and the weakness in the key length compared to symmetric encryption.

4. B. Because of its speed, asymmetric encryption is used for small amounts of data for short periods of time. Generating a shared secret key for symmetric encryption is the main use.

5. D. The public key is part of the digital certificate. The holder of the private key also knows the public key, and anyone who requests a certificate is provided with the public key.

6. C, D. Diffe-Hellman (DH) and RSA are two examples of asymmetric encryption.

7. B, C. The three main uses for digital certificates are identity, secure communications, and user authentication.

8. A. Cisco devices use Simple Certificate Enrollment Protocol (SCEP) to request a certificate automatically.

9. C. X.509 is the most widely used standard. It originated in the X.500 class but is not considered part of the class anymore.

10. A, B, D, E. Any device in the actual PKI will have to obtain a new certificate from a new Root CA server certificate. The Root CA will either have to generate a new private key or be rebuilt, depending on how it was compromised. The Cross-Certify CA is not part of the same PKI.

11. E. This answer depends on how many CAs are in the certificate path. If there is only a Root CA in the path, there are three certificates. If there are intermediate CAs in the path, there are more. In the example we presented in the chapter there were three certificates in the path.

12. A. Diffe-Hellman utilizes only one prime number at the beginning process of the algorithm.

13. A. Group 2 is 1024-bit. Group 1 is 768-bit, and Group 3 is 1536-bit.

14. B, C. A trusted third party (TTP) that is given authority is necessary in a PKI. Secrecy is incorrect because PKI stands for Public Key Infrastructure. The public key is assumed to be known to everyone.

15. A, C. The CA structure is unavailable with a single server failure. The computing power of a single server will not scale for a larger environment.

16. A, B. Digital certificates provide the ID and certificate authorities provide the digital certificates. Digital signatures are not directly associated with a PKI but may be used in a PKI.

17. A, C. The city and state should not be abbreviated because certificates are used around the world, and local users may not understand the abbreviation.

18. A. The Subject field will hold the company information of the certificate. This information may be the company of the CA if the certificate is a CA certificate.

19. D. In this chapter we used SSH as the hybrid example.

20. A, B, C. Root CA, intermediate CA, CRL, and RA are the types of servers included in a PKI.

Answers to Written Lab

1. 768, 1024, 1536
2. The public key can encrypt and decrypt.
3. Two prime numbers
4. Certificate Revocation List (CRL) server
5. PKCS #7
6. PKI stands for public key infrastructure. PKI provides a means to generate, administer, and revoke digital certificates.
7. Class 3 is most commonly seen on the Internet.
8. Version 3
9. 2 or 5
10. Symmetric is considered strong when comparing key length to key length.