

Cryptography

This section introduces an essential element of modern information security, that of cryptography.

Cryptography provides an additional layer in securing data. We will first define cryptography, explain different cryptographic algorithms, show how cryptography is implemented, and look at attacks and weaknesses of cryptography.

The next section continues with more advanced cryptography topics such as digital certificates, public key infrastructure (PKI), and secure communication and transport protocols.

Fundamentals of Cryptography

Since the dawn of human communication, there have been efforts to hide private communications from outsiders. This likely started with whispering in a friend's ear or using hand gestures. These were fine when the sender and receiver were standing close to each other but could not be used if they were far apart. And when written communication was developed, it was likewise challenging to hide these messages from outsiders. This led to many different techniques being created and used throughout the centuries to hide written messages. But invariably the outsiders were able to “reverse-engineer” the technique to read the hidden message, so new techniques had to be developed. With the introduction of mechanical devices over 100 years ago, it was possible to increase hiding to a new level so that it was virtually impossible for humans by themselves to uncover the original message.

However, outsiders then turned to using mechanical devices to unmask the messages. The same became true when computers were developed almost 60 years ago: electronic technology raised the bar even higher, but outsiders could also use computers to read the messages.

Using technology to hide a message or mask data is a foundational element today. For example, just using a web browser to perform a simple online search or purchase a product online uses this hiding technology extensively. And it has become the cornerstone of information security: a threat actor may be able to defeat protections to reach the data, but if that data is in a form that cannot be read, then the attacker faces an even higher obstacle to uncover it.

In this section, you learn about the fundamentals of cryptography. You first look at what cryptography is and how it is used. Then you examine cryptographic limitations and attacks on cryptography.

Defining Cryptography

Defining cryptography involves comparing it with steganography to understand what it is and how it works. It also involves knowing the benefits of cryptography.

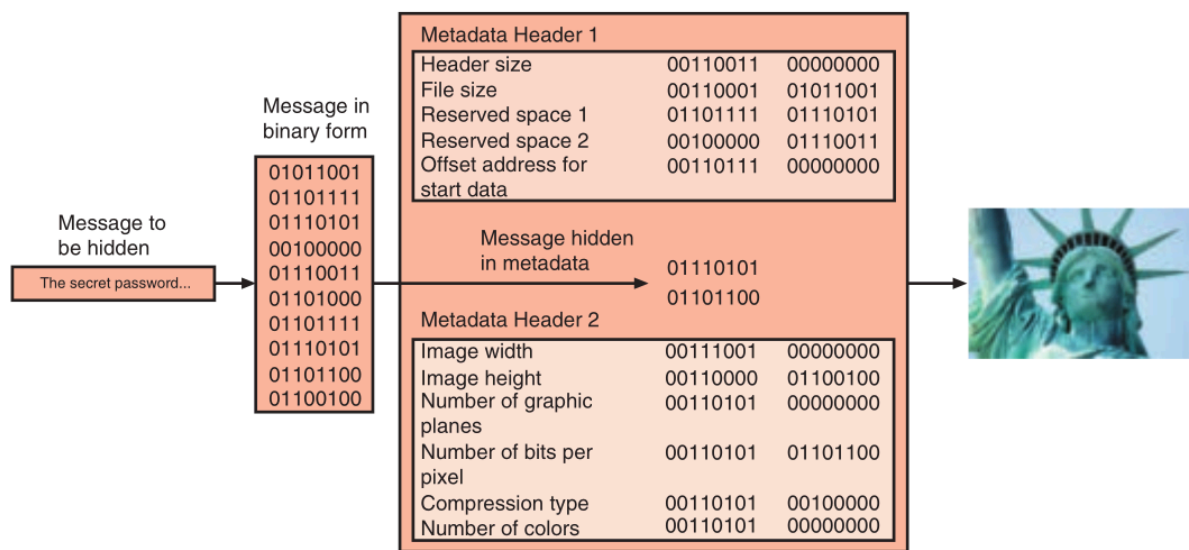
Steganography: Hiding the Message

The word *steganography* comes from Greek:

- “**steganos**” means *covered or concealed*
- “**graphia**” means *writing*

So, steganography literally means “**hidden writing**.” But in modern terms, it's about hiding any kind of data—text, images, audio, or video—inside other files or media so that it's invisible to outsiders. An example of steganography is shown in Figure 3-1.

Figure 3-1 Data hidden by steganography



Imagine you want to send a secret message to a friend, but you don't want anyone to know you're sending a secret at all. Instead of encrypting it (which makes it look suspicious), you **hide it inside something normal**—like a photo (as above), a song, or even a Word document.

Steganography is one form of obfuscation (from Latin meaning “to darken”), or the action of making something obscure. Other forms of obfuscation include data masking, which involves creating a copy of the original data and making it unintelligible, and tokenization, which obfuscates sensitive data elements, such as an account number, into a random string of characters (token).

Cryptography: Hiding the Meaning

Cryptography is the science of protecting information by transforming it into a secret code so that only the intended person can understand it (or decode the original information). It's like turning your message into a puzzle that only someone with the right key can solve.

The word cryptography comes from Greek:

“kryptos” means hidden

“graphia” means writing

So cryptography literally means “hidden writing.” But today, it's used to protect all kinds of digital data—messages, passwords, files, and even money.

Imagine you want to send a private message to a friend, but you don't want anyone else to read it. Instead of hiding the message (like steganography), you **scramble it** using a special method. Only your friend, who has the right “key,” can unscramble it and read the message.

That's cryptography: **scrambling information so it's unreadable to outsiders.**

Cryptography is the practice of transforming (“scrambling”) information so that its meaning cannot be understood by unauthorized parties but can only be understood by approved recipients.

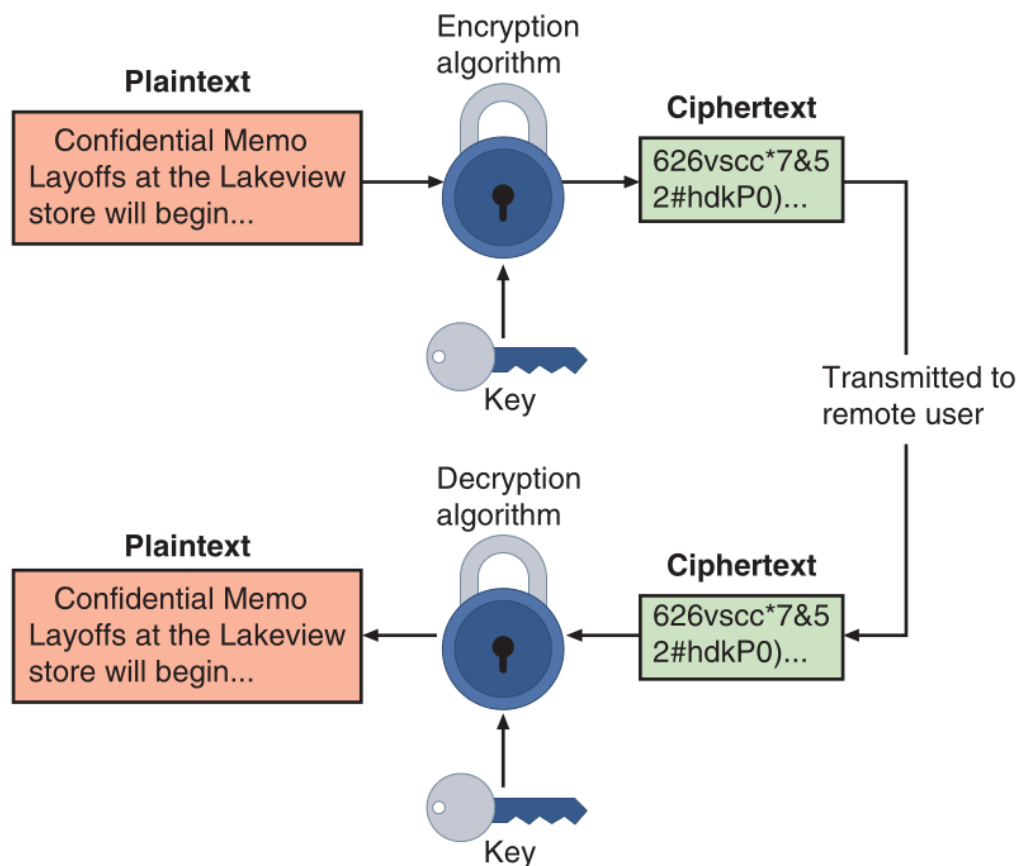
When using cryptography, the process of changing the original text into a scrambled message is known as **encryption**. The reverse process is **decryption** or changing the message back to its original form. In addition, the following terminology applies to cryptography:

- **Plaintext**. Unencrypted data that is input for encryption or is the output of decryption is called plaintext.
- **Ciphertext**. Ciphertext is the scrambled and unreadable output of encryption.
- **Cleartext**. Unencrypted data that is not intended to be encrypted is cleartext (it is “in the clear”).

Note - Steganography is sometimes used together with encryption so that the information is doubly protected. First encrypting the data and then hiding it requires someone seeking the information to first find the data and then decrypt it.

Plaintext data to be encrypted is input into a cryptographic **algorithm** (also called a **cipher**), which consists of procedures based on a **mathematical formula**. A **key** is a mathematical value entered into the algorithm to produce the ciphertext. Just as a key is inserted into a door lock to lock the door, in cryptography a **unique mathematical key** is input into the encryption algorithm to “lock” the data by creating the ciphertext. Once the ciphertext needs to be returned to plaintext so the recipient can view it, the reverse process occurs with the decryption algorithm and key to “unlock” it. The cryptographic process is illustrated in Figure 3-2.

Figure 3-2 Cryptographic process



The critical factor in cryptography is that one or more elements must be kept secret at all costs. Cryptographic algorithms are designed to be public and well known, and how they function is no secret. However, the individualized key for the algorithm that a user possesses must always be kept secret.

Benefits of Cryptography

Why use cryptography? Hiding information through encryption so that threat actors cannot view it is considered one of several different mitigation techniques (mitigation is reducing the vulnerability or severity) that enterprises can use to protect information. It is also considered a hardening technique that makes a system more resilient to attacks.

Cryptography is also unique in that it can provide protections to data in each of its three states: **data at rest, data in use, and data in transit.**

There are several common use cases (situations) for which cryptography can provide security protections. These protections include:

- **Confidentiality.** Cryptography can protect the confidentiality of information by ensuring that only authorized parties can view it. When private information, such as a list of employees to be laid off, is transmitted across the network or stored on a file server, its contents can be encrypted, which allows only authorized individuals who have the key to read it.
- **Integrity.** Cryptography can protect the integrity of information. Integrity ensures that the information is correct and no unauthorized person or malicious software has altered that data. Because ciphertext requires that a key must be used to open the data before it can be changed, cryptography can ensure its integrity. The list of employees to be laid off, for example, can be protected so that no names can be added or deleted by unauthorized personnel.
- **Authentication.** The authentication of the sender can be verified through cryptography. Specific types of cryptography, for example, can prevent a situation such as circulation of a list of employees to be laid off that appears to come from a manager but, in reality, it was sent by an imposter.
- **Nonrepudiation.** Cryptography can enforce nonrepudiation. Repudiation is defined as denial; nonrepudiation is the **inability to deny**. In information technology (IT), nonrepudiation is the process of **proving** that a user performed an action, such as sending an email message. Nonrepudiation prevents an individual from fraudulently reneging on an action. The nonrepudiation features of cryptography can prevent a manager from claiming they never sent the list of employees to be laid off to an unauthorized third party.

Note - A practical example of nonrepudiation is Carla taking her car into a repair shop for service and signing an estimate form of the cost of repairs and authorizing the work. If Carla later returns and claims she never approved a specific repair, the signed form can be used as non-repudiation.

- **Obfuscation.** Obfuscation is making something obscure or unclear. Cryptography can provide a degree of obfuscation by encrypting a list of employees to be laid off so that an unauthorized user cannot read it.

The security protections afforded by cryptography are summarized in Table 3-1 below .

Table 3-1 Information protections by cryptography

Characteristic	Description	Protection
Confidentiality	Ensures that only authorized parties can view the information	Encrypted information can only be viewed by those who have been provided the key.
Integrity	Ensures that the information is correct and no unauthorized person or malicious software has altered the data	Encrypted information cannot be changed except by authorized users who have the key.
Authentication	Provides proof of the genuineness of the user	Proof that the sender was legitimate and not an imposter can be obtained.
Nonrepudiation	Proves that a user performed an action	Individuals are prevented from fraudulently denying that they were involved in a transaction.
Obfuscation	Makes something obscure or unclear	By making it obscure, the original information cannot be determined.

Cryptographic Algorithms

A cryptographic algorithm consists of procedures based on a **mathematical formula**. Different variations of cryptographic algorithms exist. The three broad categories of cryptographic algorithms are **hash algorithms**, **symmetric cryptographic algorithms**, and **asymmetric cryptographic algorithms**.

Hash Algorithms

One type of cryptographic algorithm is a **one-way algorithm**. Its purpose is not to create ciphertext that can later be decrypted back into plaintext; in fact, it is designed so that it cannot be reversed to reveal the original set of data.

A hash algorithm creates a **unique “digital fingerprint”** of a set of data. This process is called **hashing**, and the resulting fingerprint is a **digest** (sometimes called a message digest or simply a **hash**) that represents the contents. Hashing protects the **integrity** of data, proving that no one has altered it, and is used primarily for comparison purposes.

To illustrate how a hash algorithm functions, consider when 12 is multiplied by 34, the result is 408. If a user were asked to determine the two numbers used to create the number 408, it would not be possible to work backward and derive the original numbers with absolute certainty because there are too many mathematical possibilities (1×408 , 2×204 , 3×136 , 4×102 , etc.). Hashing is similar in that it is not possible to determine the plaintext from the digest.

A hashing algorithm is considered secure if it has these characteristics:

- **Fixed size.** A digest of a short set of data should produce the same size as a digest of a long set of data. For example, a digest of the single letter **a** is 86be7afa339d0fc7cfc785e72f578d33, while a digest of **1 million occurrences** of the letter **a** is 4a7f5723f954eba1216c9d8f6320431f, the **same length**.
- **Unique.** Two different sets of data cannot produce the same digest. Changing a single letter in one data set should produce an entirely different digest. For example, a digest of **Sunday** is 0d716e73a2a7910bd4ae63407056d79b while a digest of **sunday** (lowercase s) is 3464eb71bd7a4377967a30da798a1b54.
- **Original.** It should not be possible to produce a data set that has a desired or predefined hash.

- **Secure.** The resulting hash cannot be reversed to determine the original plaintext. There are several common hash algorithms. One of the earliest hash algorithms is a “family” of algorithms known as **Message Digest (MD)**. Serious weaknesses have been identified in **MD5** and it is no longer considered suitable for use. Other secure hash algorithms include:
 - **Secure Hash Algorithm (SHA).** Another family of hashes is the Secure Hash Algorithm (SHA). SHA-2 has six variations, the most common are SHA-256, SHA-384, and SHA-512 (the last number indicates the length in bits of the digest that is generated) and is currently considered to be a secure hash. In 2015, SHA-3 was announced as a new standard. One of the design goals of SHA-3 was for it to be dissimilar to previous hash algorithms to prevent threat actors from building upon any earlier work of compromising these algorithms.
 - **RipeMD.** RipeMD stands for **RACE Integrity Primitives Evaluation Message Digest**. The primary design feature of RipeMD is two different and independent parallel chains of computation, the results of which are then combined at the end of the process. There are several versions of RipeMD, all based on the length of the digest created, including RipeMD-160, RipeMD-256, and RipeMD-320.
 - **Whirlpool.** Whirlpool uses a block cipher and takes a message of any length less than 2256 bits and returns a 512-bit message digest.


Table 3-3 illustrates the digests generated from several one-way hash algorithms of the word *Cengage*.

Table 3-3 Digests generated from one-time hash algorithms

Hash	Digest
RipeMD160	dd52a79bce64a1d145b51ce639e0dadda976516d
SHA-256	f6c8a86bf6a5128cbaf2ad251b0beaa3604c11c51587de518737537800098d76
SHA3-512	3a82d58e17f3991413c5f4e9811930b69513bba02a860eed82070f892ab381f9fd926a88cf68745565f51a93b97a1317ae8b84e2dfb798e4a2aa331187dc9e34
Whirlpool	1428cacf499fce9ef439f95a27f0efdc518e97edf9714e234cffe0c22dcb0e2c4bd3f22a975670f4a229452062b8bd0c6c9244e4c986f22c61ce951de31c505

Hashing is often used as a check to verify that the original contents of an item have not been changed. For example, digests are often calculated and then posted on websites for files that can be downloaded, as seen in Figure 3-4.

Figure 3-4 Verify downloads with digests

 **Hashes**

KeePass 2.53

KeePass-2.53.zip:

SHA-1: 092CC353 1A46B600 968636B5 6851E23F FEE5505F

SHA-256: DCA1B970 9A87BA67 ECEF8905 80C0B2AD 6E3ACF38 546A642C AFE36D70 62E40AD4

Size: 3225609 B

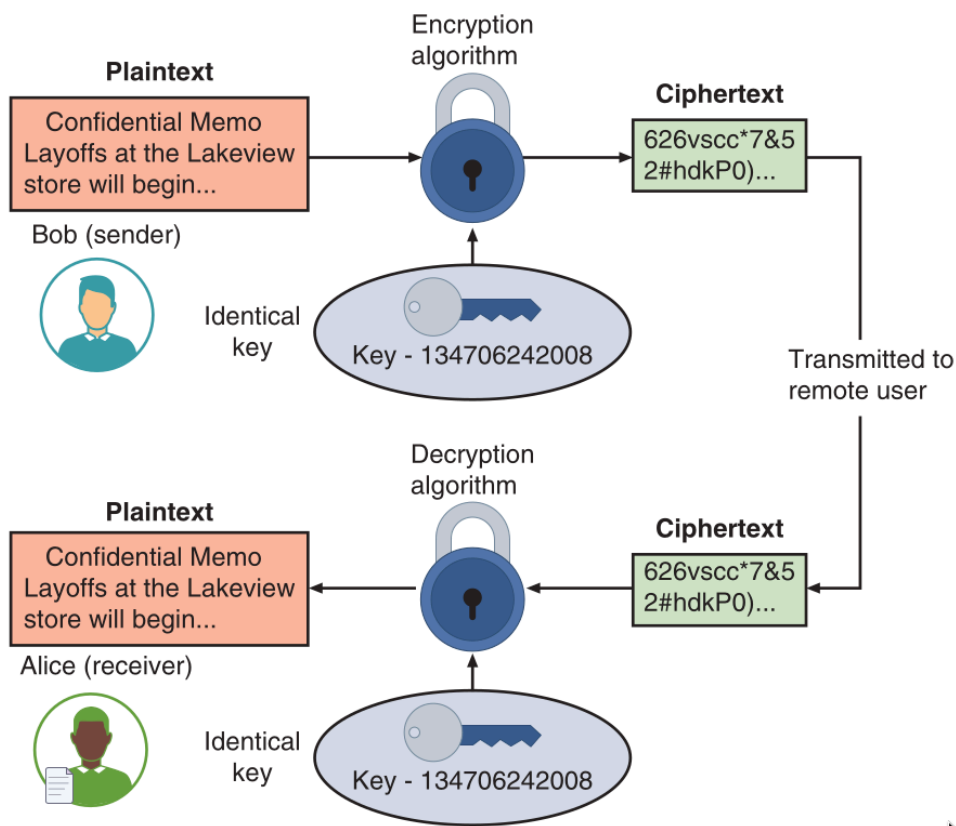
Sig.: [\[OpenPGP ASC\]](#)

After downloading the file, a user can create their own digest on the file and then compare it with the digest value posted on the website; a match indicates that there has been no change to the original file while being downloaded.

Symmetric Cryptographic Algorithms

The original cryptographic algorithms for encrypting and decrypting data are symmetric cryptographic algorithms. **Symmetric cryptographic algorithms** use the same key to encrypt and decrypt the data. Data encrypted by Bob with a key can only be decrypted by Alice using that same key. Thus, it is essential that the key be kept private (**secret**) so, for this reason, symmetric encryption is also called **private key cryptography**. Symmetric encryption, which protects the confidentiality of data, is illustrated in Figure 3-5, where identical keys are used to encrypt and decrypt a document.

Figure 3-5 Symmetric (private key) cryptography



Caution - The element that must be kept secret in symmetric cryptography is the key.

Symmetric cryptography can provide strong encryption - if the key is kept secure between the sender and all the recipients. There are several strong symmetric cryptographic algorithms. These include:

- **Advanced Encryption Standard (AES).** The Advanced Encryption Standard (AES) is a symmetric algorithm that performs three steps on every block (128 bits) of plaintext. Within step 2, multiple rounds are performed depending on the key size: a 128-bit key performs 9 rounds, a 192-bit key performs 11 rounds, and a 256-bit key, known as AES-256, uses 13 rounds. Within each round, bytes are substituted and rearranged, and then **special multiplication** is performed based on the new arrangement. To date, no attacks have been successful against AES.

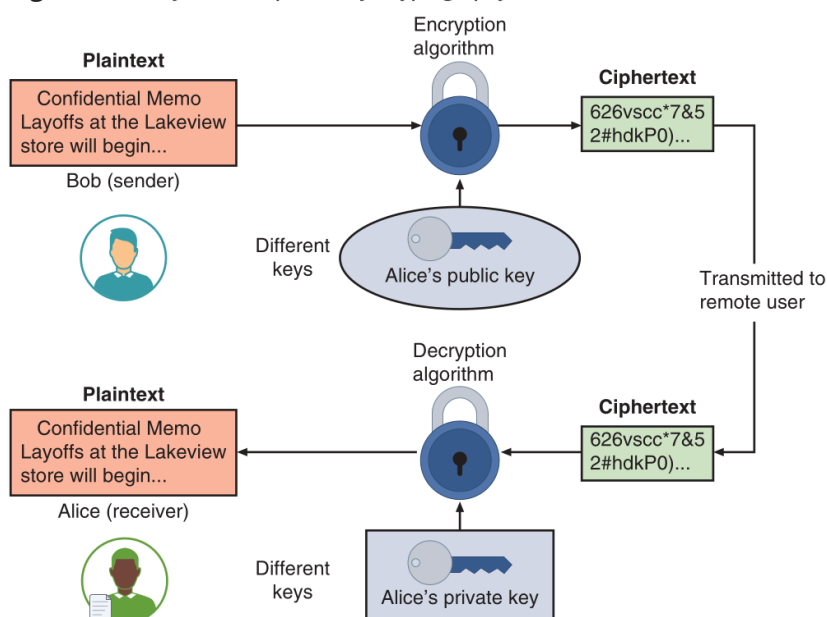
- **Blowfish and Twofish.** Blowfish is a block cipher algorithm that operates on 64-bit blocks and can have a key length from 32 to 448 bits. To date, no significant weaknesses have been identified. A later derivation of Blowfish known as Twofish is also considered to be a strong algorithm, although it has not been used as widely as Blowfish.

Asymmetric Cryptographic Algorithms

If Bob wants to send an encrypted message to Alice using symmetric encryption, he must be sure that she has the key to decrypt the message. Yet how should Bob get the key to Alice? He cannot send it electronically as an email attachment because that would make it vulnerable to interception by attackers. Nor can he encrypt the key and send it because Alice would not have a way to decrypt the encrypted key. This illustrates the primary weakness of symmetric encryption algorithms: distributing and maintaining a secure single key among multiple users, who are often scattered geographically, poses significant challenges.

A completely different approach is **asymmetric cryptographic algorithms**, also known as **public key cryptography**. Asymmetric encryption, which protects the confidentiality of data, uses two keys instead of only one. These keys are **mathematically related** and are known as the **public key** and the **private key**. The public key is known to everyone and can be freely distributed, while the private key is known only to the individual to whom it belongs. When Bob wants to send a secure message to Alice, he uses Alice's public key to encrypt the message. Alice then uses her private key to decrypt it. Asymmetric cryptography is illustrated in Figure 3-6.

Figure 3-6 Asymmetric (public key) cryptography



Caution - The element that must be kept secret in asymmetric cryptography is the private key.

Several important principles regarding asymmetric cryptography are:

- **Key pairs.** Unlike symmetric cryptography that uses only one key, asymmetric cryptography requires a pair of keys.
- **Public key.** Public keys by their nature are designed to be public and do not need to be protected. They can be freely given to anyone or even posted on the Internet.

- **Private key.** The private key should be kept confidential and never shared.
- **Both directions.** Asymmetric cryptography keys can work in both directions. A document encrypted with a public key can be decrypted with the corresponding private key. In the same way, a document encrypted with a private key can be decrypted with its public key.

There are different asymmetric algorithms and variations, discussed next. Also, there are issues surrounding key management.

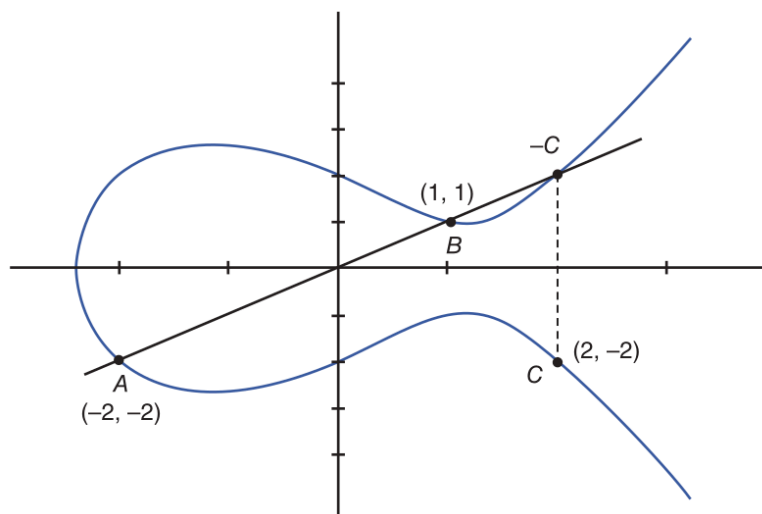
RSA: The RSA algorithm multiplies two large **prime numbers** (a prime number is a number divisible only by itself and 1), **p** and **q**, to compute their product ($n = pq$). Next, a number **e** is chosen that is less than **n** and a prime factor to $(p - 1)(q - 1)$. Another number **d** is determined, so that $(ed - 1)$ is divisible by $(p - 1)(q - 1)$. The values of **e** and **d** are the **public** and **private exponents**. The **public key** is the pair (n, e) while the **private key** is (n, d) . The numbers **p** and **q** can be discarded. An illustration of the RSA algorithm using very small numbers is as follows:

1. Select two prime numbers, **p** and **q** (in this example, $p = 7$ and $q = 19$).
2. Multiply **p** and **q** together to create **n** ($7 * 19 = 133$).
3. Calculate **m** as $p - 1 * q - 1$ ($[7 - 1] * [19 - 1]$ or $6 * 18 = 108$).
4. Find a number **e** so that it and **m** have no common positive divisor other than 1 ($e = 5$).
5. Find a number **d** so that $d = (1 + n * m) / e$ or $(1 + 133 * 108) / 5$ or $14,364 / 5 = 2875$.

For this example, the **public key n** is 133 and **e** is 5, while the **private key n** is 133 and **d** is 2873.

Elliptic Curve Cryptography (ECC): An elliptic curve is a set of points that satisfy a specific mathematical equation. This paved the way for a different form of asymmetric encryption not based on factoring. Instead of using large prime numbers as with RSA, elliptic curve cryptography (ECC) uses sloping curves. An **elliptic curve** is a function drawn on an **x- or y-axis** as a gently curved line. By adding the values of two points on the curve, a third point on the curve can be derived, of which the inverse is used, as illustrated in Figure 3-7.

Figure 3-7 Elliptic curve cryptography



With ECC, users share one elliptic curve and one point on the curve. One user chooses a secret random number and computes a public key based on a point on the curve; the other user does the same. They can now exchange messages because the shared public keys can generate a private key on an elliptic curve.

The difference in size necessary to produce the same level of security between RSA and ECC keys is significant. Table 3-4 compares the key length (in bits) of RSA and ECC keys that have the same level of security.

Table 3-4 RSA versus ECC key length for same security level

RSA key length	ECC key length
1024	160
2048	224
3072	256
7680	384
15360	521

Despite a slow start, ECC has gained wide popularity. It is used by the **U.S. government** to protect internal communications, by the **Tor project** to help ensure anonymity, and as the mechanism to prove ownership of bitcoins. All modern **operating systems (OSs)** and **web browsers** rely on ECC. Because mobile devices are limited in terms of computing power due to their smaller size, ECC offers security that is comparable to other asymmetric cryptography algorithms but with smaller key sizes, resulting in faster computations and lower power consumption.

Digital Signature Algorithm (DSA): Asymmetric cryptography can also be used to provide proofs. Suppose Alice receives an encrypted document that says it came from Bob. Although Alice can be sure that the encrypted message was not viewed or altered by someone else while being transmitted, how can she know for certain that Bob was the sender ? Because Alice’s public key is widely available, anyone could use it to encrypt the document. Another individual could have created a fictitious document, encrypted it with Alice’s public key, and then sent it to Alice while pretending to be Bob. Alice’s key can verify that no one read or changed the document in transport, but it cannot verify the sender.

Proof can be provided with asymmetric cryptography, however, by creating a **digital signature**, which is an **electronic verification** of the sender. A handwritten signature on a paper document serves as proof that the signer has read and agreed to the document. A digital signature is much the same but can provide additional benefits. A digital signature can:

- **Verify the sender.** A digital signature serves to confirm the identity of the person from whom the electronic message originated.
- **Prevent the sender from disowning the message.** The signer cannot later attempt to disown the message by claiming the signature was forged (nonrepudiation).
- **Prove the integrity of the message.** A digital signature can prove that the message has not been altered since it was signed.

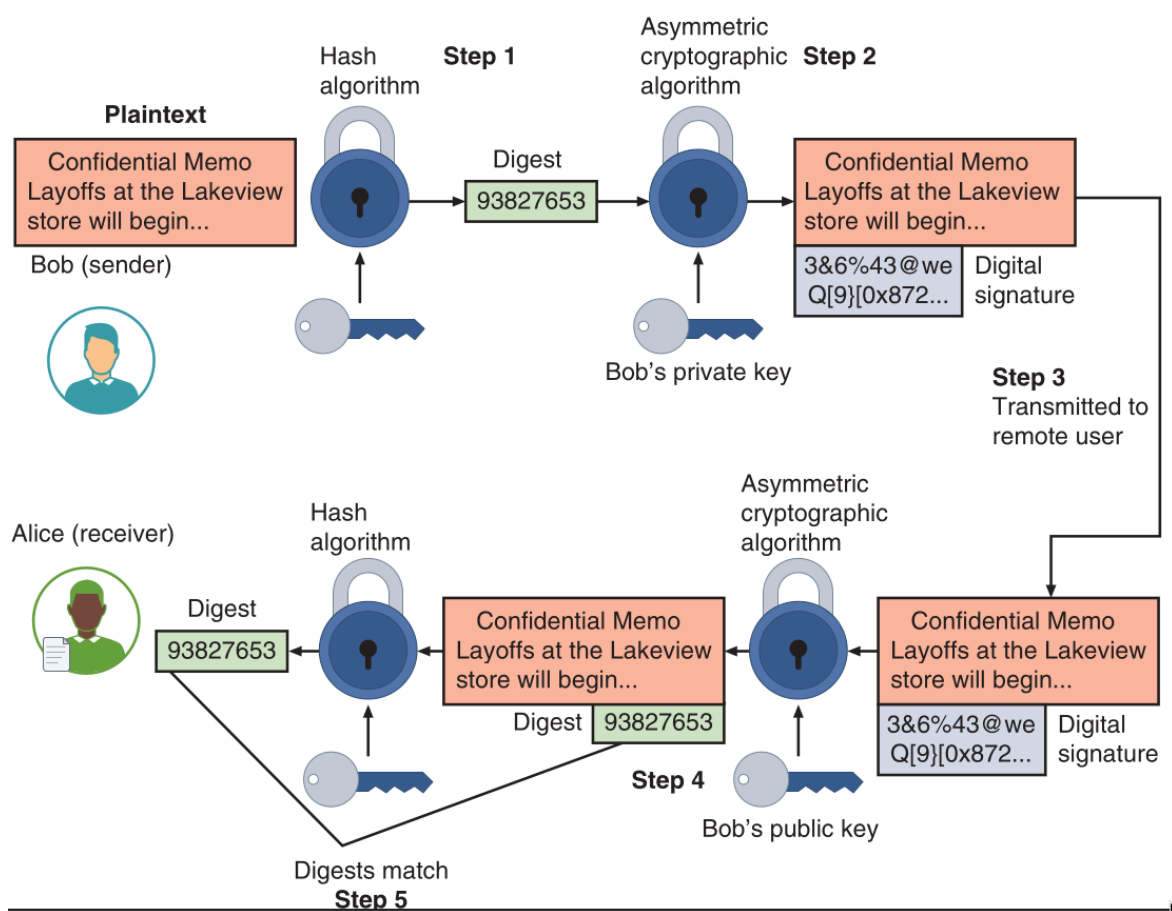
The basis for a digital signature rests on the ability of asymmetric keys to work in both directions (a public key can encrypt a document that can be decrypted with a private key, and the private key can encrypt a document that can be decrypted by the public key).

The steps for Bob to send a digitally signed message to Alice are:

1. After creating a memo, Bob generates a digest on it.
2. Bob then encrypts the digest with his private key. This encrypted digest is the digital signature for the memo.
3. Bob sends both the memo and the digital signature to Alice.
4. When Alice receives them, she decrypts the digital signature using Bob's public key, revealing the digest. If she cannot decrypt the digital signature, then she knows that it did not come from Bob (because only Bob's public key can decrypt the digest generated with his private key).
5. Alice then hashes the memo with the same hash algorithm Bob used and compares the result to the digest she received from Bob. If they are equal, Alice can be confident that the message has not changed since he signed it. If the digests are not equal, Alice will know the message has changed since it was signed.

These steps are illustrated in Figure 3-8.

Figure 3-8 Digital signature



Caution - Using a digital signature does not encrypt the message itself. In the example, if Bob wanted to ensure the privacy of the message, he also would have to encrypt it using Alice's public key.

Key Exchange: Public and private keys may result in confusion regarding whose key to use and which key should be used. Table 3-5 lists the practices to follow when using asymmetric cryptography.

Table 3-5 Asymmetric cryptography practices

Action	Whose key to use	Which key to use	Explanation
Bob wants to send Alice an encrypted message.	Alice's key	Public key	When an encrypted message is to be sent, the recipient's, and not the sender's, key is used.
Alice wants to read an encrypted message sent by Bob.	Alice's key	Private key	An encrypted message can be read only by using the recipient's private key.
Bob wants to send a copy to himself of the encrypted message that he sent to Alice.	Bob's key	Public key to encrypt Private key to decrypt	An encrypted message can be read only by the recipient's private key. Bob would need to encrypt it with his public key and then use his private key to decrypt it.
Bob receives an encrypted reply message from Alice.	Bob's key	Private key	The recipient's private key is used to decrypt received messages.
Bob wants Susan to read Alice's reply message that he received.	Susan's key	Public key	The message should be encrypted with Susan's key for her to decrypt and read with her private key.
Bob wants to send Alice a message with a digital signature.	Bob's key	Private key	Bob's private key is used to encrypt the hash.
Alice wants to see Bob's digital signature.	Bob's key	Public key	Because Bob's public and private keys work in both directions, Alice can use his public key to decrypt the hash.

In addition to confusion regarding which key to use, there are also issues with sending and receiving keys (key exchange) such as exchanging a symmetric private key. One solution is to make the exchange outside of the normal communication channels (for example, Alice could hire Carol to deliver a USB flash drive containing the key directly to Bob).

There are different solutions for a key exchange that occurs within the normal communications channel of cryptography, including:

- **Diffie-Hellman (DH).** The DH key exchange requires Alice and Bob to each agree upon a large prime number and related integer. Those two numbers can be made public, yet Alice and Bob, through mathematical computations and exchanges of intermediate values, can separately create the same key.
- **Diffie-Hellman Ephemeral (DHE).** Whereas DH uses the same keys each time, DHE uses different keys. Ephemeral keys are temporary keys that are used only once and then discarded.
- **Elliptic Curve Diffie-Hellman (ECDH).** ECDH uses elliptic curve cryptography instead of prime numbers in its computation.
- **Perfect forward secrecy.** Public key systems that generate random public keys that are different for each session are called perfect forward secrecy. The value of this is that if the secret key is compromised, it cannot reveal the contents of more than one message.

Using Cryptography

Cryptography can be applied through either software or hardware. Also, a relatively new technology known as **blockchain** uses cryptography as its basis.

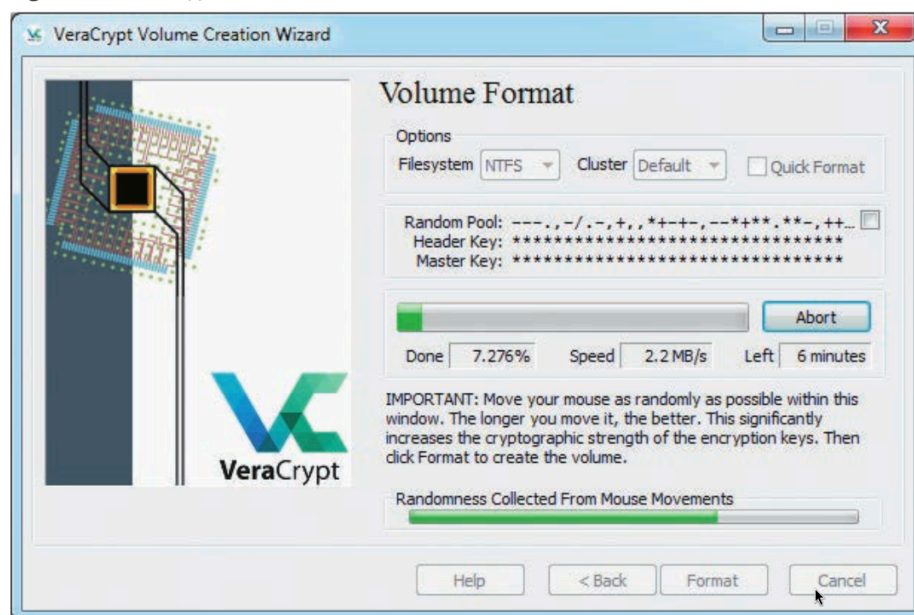
Encryption through Software

Software-based cryptography can be implemented at the file level or disk level. It can also be implemented across an entire database using software.

File and File System Cryptography: Cryptographic software can be used to encrypt or decrypt files one by one (file-level encryption). However, this can be a cumbersome process. Instead, protecting groups of files, such as all files in a specific folder, can take advantage of the operating system's (OS's) file system. A **file system** is a method used by OSs to store, retrieve, and organize files. Protecting individual files or multiple files through file system cryptography can be performed using third-party software or native OS cryptographic features.

Third-Party Software. A wide variety of third-party software tools are available for performing encryption. This encryption is essentially file-level encryption. These tools include **GNU Privacy Guard** (which is abbreviated GnuPG), **AxCrypt**, **Folder Lock**, and **VeraCrypt**, which is seen in Figure 3-9.

Figure 3-9 VeraCrypt



Operating System Encryption. Modern OSs provide encryption support natively. Microsoft's Encrypting File System (**EFS**) is a cryptography system for the Windows operating systems that use the Windows **NTFS** file system, while Apple's **FileVault** performs a similar function. Because these are tightly integrated with the file system, file encryption and decryption are transparent to the user. Any file created in an encrypted folder or added to an encrypted folder is automatically encrypted. When an authorized user opens a file, it is decrypted as data is read from a disk; when a file is saved, the OS encrypts the data as it is written to the disk.

Disk Encryption: Instead of protecting individual files or groups of files, cryptography can also be applied to entire disks as well, either in part or in full. The different types of a disk encryption include:

- **Disk-level encryption.** Protecting the entire drive using cryptography is known as **full-disk encryption (FDE)** and protects all data on a drive, including the installed OS. One example of FDE software is included in Microsoft Windows and is known as **BitLocker** drive encryption software. BitLocker encrypts the entire disk, including the Windows Registry and any temporary files that might hold confidential information. BitLocker prevents attackers from accessing data by booting from another OS or placing the drive in another computer.

- **Volume-level encryption.** A volume is a section of a drive that is accessible by a user and has a file system associated with it. That is, a volume is a single accessible storage area with a single file system. A partition is not the same as a volume. A partition is a logical division of a hard drive and is essentially a “chunk” of a disk that does not necessarily contain a file system or is even formatted to store data. Software can apply cryptography to a volume, known as volume-level encryption. Partition-level encryption may be applied depending on if the partition is formatted and contains a file system.

Database Encryption: A database is an organized collection of structured information stored electronically on a computer. A database has an engine that stores and retrieves the data, and that engine is manipulated by a user through a **database management system (DBMS)** or a programming language. (Most databases use the **Structured Query Language [SQL]** to interact with a database.) A relational database, the most common type of database, is organized into rows (horizontal) and columns (vertical) in a series of tables.

Applying cryptography to a database (**database-level encryption**) typically occurs in one of two ways. The **plugin method** requires attaching an encryption module (package) onto the DBMS. This method can be applied to both commercial and open-source databases. A more widely used approach is **transparent data encryption (TDE)**, which executes encryption and decryption within the database engine itself. TDE does not require any additional packages or code modification of the database, engine, or DBMS and is easier to manage.

The encryption level for database cryptography can vary. Typically, encryption is performed at the database file level (encrypting all the database files), at the table level, or at the column level. It is possible to encrypt each row (record-level encryption) or even each individual data element (cell), but this high level of “granularity” requires careful planning and strict key management.

Hardware Encryption: Software encryption suffers from the same fate as any application program: it can be subject to attacks to exploit its vulnerabilities. As a more secure option, cryptography can be embedded in hardware. Hardware encryption cannot be exploited like software encryption. Hardware encryption can be incorporated into a USB device or drive. The hardware security model and trusted execution environment can provide an even higher level of security.

USB Devices - Many instances of data leakage are the result of USB flash drives being lost or stolen. Cryptographic features can be built into the hardware of a USB device instead of installing separate software. This allows for all data written to the USB device to be automatically encrypted. Special-purpose USB devices have additional features such as the ability for administrators to remotely control and track the activity on the devices and can remotely disable a compromised or stolen drive. One hardware-based USB device allows administrators to remotely prohibit accessing the data on a device until it can verify its status, to lock out the user completely the next time the device connects, or even to instruct the drive to initiate a self-destruct sequence to destroy all data.

Self-Encrypting Drives (SEDs) - Just as a cryptographic hardware-based USB device can automatically encrypt any data stored on it, **self-encrypting drives (SEDs)** are drives that can protect all the data written to them. When the computer or other device with an SED is initially powered up, the drive and the host device perform an authentication process. If the authentication process fails, the drive can be configured to deny any access to the drive or even perform a cryptographic erase on specified blocks of data, deleting even the decryption keys so that no data can be recovered. This also makes it impossible to install the drive on another computer to read its contents.

Hardware Security Module (HSM) - A **hardware security module (HSM)** is a removable external cryptographic device. An HSM can be a USB device, an expansion card, a device that connects directly to a computer through a port, or a secure network server. It includes an onboard random number generator and key storage facility, as well as accelerated symmetric and asymmetric encryption, and can even back up sensitive material in encrypted form. Because all of this is done in hardware and not through software, it cannot be compromised by malware.

Some financial banking software comes with a specialized USB HSM hardware key, also called a “security dongle.” This device is paired with a specific financial account and cannot be cloned or compromised. Figure 3-10 shows a USB HSM.

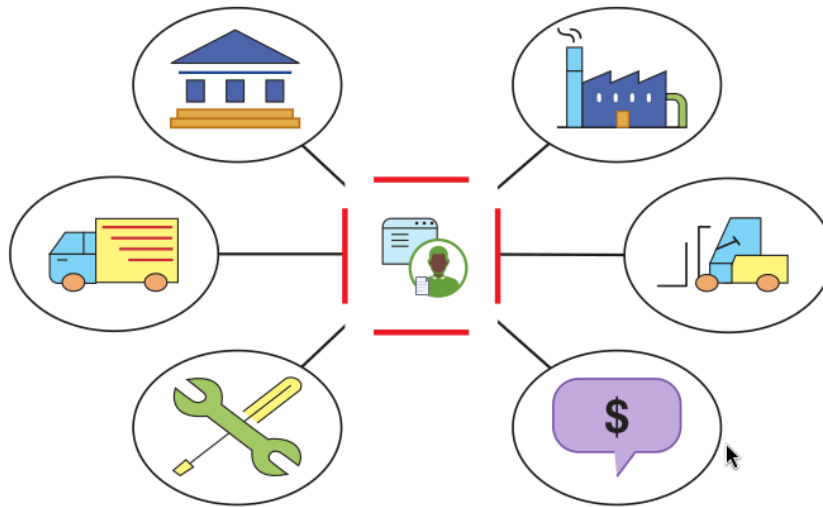
Figure 3-10 USB HSM



Trusted Execution Environment (TEE) - Instead of relying on vulnerable software or an external device to be connected to a computer, a **trusted execution environment (TEE)** is a secure cryptoprocessor (a motherboard chip) that is internal to the computer itself. Because it is hardware that cannot be tampered with, its contents are secure. A TEE protects the confidentiality and integrity of the code and data stored on it.

Blockchain - A blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. A single shared ledger is seen in Figure 3-12.

Figure 3-12 Multiple organizations using single ledger



At a high level, blockchain technology allows a network of computers to agree at regular intervals on the true state of a distributed ledger. It is a system in which a record of transactions made are maintained across several computers that are linked in a peer-to-peer network.

Note - Blockchain relies heavily on cryptographic hash algorithms, most notably the SHA-256, to record its transactions. This makes it computationally infeasible to try to replace a block or insert a new block of information without the approval of all entities involved.

There are different types of blockchains. A **public blockchain**, also called an **open public ledger**, is a blockchain network that anyone can join and become part of it. A common use of public blockchains is for exchanging cryptocurrencies and crypto mining.

A **private blockchain** operates in a closed network and is best suited for enterprises and businesses that implement a blockchain only for internal uses.

A **federated blockchain**, also called a **consortium blockchain**, is typically used when organizations need both a public and private blockchain. In this type, there is more than one organization involved who provides access to preselected nodes for reading, writing, and auditing.

Cryptographic Limitations and Attacks

As with all information security defenses, cryptography is not an absolute protection. There are limitations to cryptography, and it continually faces attacks.

Limitations of Cryptography

Despite providing widespread protections, cryptography faces constraints (limitations) that can impact its effectiveness. In recent years, the number of small electronic devices that consume very small amounts of power (low-power devices) has grown significantly. These devices range from tiny sensors that control office heating and lighting to consumer devices such as thermostats and lightbulbs. These devices need to be protected from threat actors who could accumulate their data and use it in nefarious ways.

In addition, many applications require extremely fast response times, most notably communication applications (like collecting car toll road payments), high-speed optical networking links, and secure storage devices such

as **solid-state disks**. Cryptography is viewed as a necessary feature to add to protect low-power devices and applications with fast response times to make them secure.

However, adding cryptography to low-power devices or those that have near instantaneous response times can be difficult. To perform their computations, cryptographic algorithms require both time and energy, each of which are typically in short supply for low-power devices and applications needing ultra-fast response times. This results in a resource versus security constraint, or a limitation in providing strong cryptography due to the tug-of-war between the available resources (time and energy) and the security provided by cryptography. Table 3-6 lists additional cryptographic constraints.

Table 3-6 Cryptographic Constraints

Limitation	Explanation
Speed	The speed at which data can be encrypted or decrypted depends on several hardware and software factors and, in some instances, a slower speed is unacceptable.
Size	The resulting size of an encrypted file can be as much as one-third larger than the plaintext version.
Weak keys	Some ciphers can produce a weak key that causes the cipher to behave in unpredictable ways or may compromise overall security.
Key length	Some ciphers have a short key length , or the number of bits in a key, that results in weaker security.
Longevity	As computers continue to become more powerful and can “crack” keys, the longevity or useful lifetime of service of ciphers may diminish.
Predictability	A weak random number generator of the cipher may create predictable output.
Reuse	If someone reuses the same key for each encryption, then it provides a larger data footprint for an attacker to use in attempting to break the encryption.
Entropy	Entropy is the measure of randomness of a data-generating function, and ciphers with low entropy give the ability to predict future-generated values.
Computational overhead	Sensors and Internet of Things (IoT) devices often lack the capacity to accommodate the computational overhead for cryptography.

Attacks on Cryptography

There are several different types of attacks on cryptography. Two of the most common include **algorithm attacks** and **collision attacks**. In addition, a new type of computing, called **quantum computing**, may have a profound impact on cryptography.

• **Algorithm Attacks.** Modern cryptographic algorithms are reviewed, tested, and vetted by specialists in cryptography over several years before they are released to the public for use. Very few threat actors have the advanced skills needed to even attempt to break an algorithm. However, there are other methods by which attackers can focus on circumventing strong algorithms. These include known **ciphertext attacks**, **downgrade attacks**, and **taking advantage of misconfigurations**.

Known Ciphertext Attacks - When properly implemented, cryptography prevents the threat actor from knowing the plaintext or the key; the only item they can see is the ciphertext itself. Yet there are sophisticated statistical tools that can be used to perform an analysis of the ciphertext in an attempt to discover a pattern in the ciphertexts, which then may be useful in revealing the plaintext or key. This is called a known ciphertext attack or ciphertext-only attack because all that is known is the ciphertext - but it can still reveal clues that may be mined.

Note - Wireless data networks are particularly susceptible to known ciphertext attacks. This is because threat actors can capture large sets of ciphertexts to be analyzed, and the attackers may be able to inject their own frames into the wireless transmissions.

Downgrade Attacks - Because of the frequent introduction of new hardware and software, often they include backward compatibility so that a newer version can still function with the older version. However, in most instances, this means that the newer version must revert to the older and less secure version. In a **downgrade attack**, an attacker forces the system to abandon the current higher security mode of operation and instead “fall back” to implementing an older and less secure mode. This then allows the threat actor to attack the weaker mode.

Attacks Based on Misconfigurations - Most breaches of cryptography are the result of incorrect choices or misconfigurations of the cryptography options, known as misconfiguration implementation. Selecting weak algorithms, like DES or SHA-1, should be avoided since these are no longer secure. Many cryptographic algorithms have several configuration options and, unless careful consideration is given to these options, the cryptography may be improperly implemented. Also, careless users can weaken cryptography if they choose SHA-256 when a much stronger algorithm like SHA3-512 is available through a simple menu choice, for example.

- **Collision Attacks.** One of the foundations of a hash algorithm is that each digest must be unique. If it were not unique, then a threat actor could trick users into performing an action that was assumed to be safe but in reality was not. For example, digests are calculated and then posted on websites for files that can be downloaded. Suppose an attacker could infiltrate a website and post their own malicious file for download, but when the digest was generated for this malicious file, it created the same as that posted for the legitimate file. Two files having the **same digest** is known as a **collision**. A collision attack is an attempt to find two input strings of a hash function that produce the same hash result.

Note - While hash algorithms that produce long digests, like SHA3-512, have very low odds of such a collision, for hash algorithms that produce shorter digests, such as MD5, the odds increase.

Typically, a threat actor would be forced to try all possible combinations until a collision was found. However, a statistical phenomenon called the **birthday attack** makes it easier. It is based on the **birthday paradox**, which says that for there to be a 50 percent chance that someone in a given room shares your birthday, 253 people would need to be in the room. If, however, you are looking for a greater than 50 percent chance that any two people in the room have the same birthday, you only need 23 people. That’s because the matches are based on pairs. If you choose yourself as one side of the pair, then you will need 253 people to have 253 pairs (in other words, it is you combined with 253 other people to make up all 253 sets). But if you are only concerned with matches and not concerned with matching someone with you specifically, then you only need 23 people in the room because it only takes 23 people to form 253 pairs when cross-matched with each other. This applies to hashing collisions in that it is much harder to find something that collides with a specific hash than it is to find any two inputs that hash to the same value.

Note - With the birthday paradox, the question is whether each person must link with every other person. If so, only 23 people are needed; if not, when comparing only your single birthday to everyone else’s, 253 people are needed.

- **Quantum Computing.** The foundation of modern technology is the bit (**binary digit**) that can either be off (0) or on (1). However, the development of a revolutionary different type of computer has been underway

for several years called a quantum computer. Quantum computing relies on **quantum physics** using **atomic-scale units (qubits)** that can be both **0 and 1 at the same time**. As a result, it is possible for one qubit to carry out two separate streams of calculations simultaneously, so that quantum computers will be much faster and more efficient than today's computers.

However, quantum computing poses a risk for cryptography. Asymmetric cryptography begins by multiplying two prime numbers, and what makes this method strong is that it is difficult for today's computers to determine the prime numbers that make up the value (factoring). A single quantum computer could perform factoring by using hundreds of atoms in parallel to quickly factor huge numbers. This would result in virtually all current asymmetric cryptographic algorithms being rendered useless.

Summary

- Steganography hides the existence of information, while cryptography hides the meaning of the information. Cryptography masks information so that it cannot be read.
- When using cryptography, the original data, called plaintext, is input into a cryptographic encryption algorithm that has a mathematical value (a key) used to create ciphertext. Cryptographic algorithms are designed to be public, while the individualized key for the algorithm that a user possesses must always be kept secret. Cryptography can provide confidentiality, integrity, authentication, nonrepudiation, and obfuscation. It can also protect data as it resides in any of three states: data in use, data in transit, and data at rest.
- There are different variations of cryptographic algorithms. One variation is based on the device (if any) that is used in the cryptographic process. Another variation is the amount of data that is processed at a time. A stream cipher takes one character and replaces it with one character, while a block cipher manipulates an entire block of plaintext at one time. A sponge function takes as input a string of any length and returns a string of any requested variable length. This function repeatedly applies a process on the input that has been padded with additional characters until all characters are used.
- Hashing creates a unique digital fingerprint called a digest, which represents the contents of the original material. Hashing is not designed for encrypting material that will be later decrypted. Instead, it protects the integrity of data, proving that no one has altered it, and is used primarily for comparison purposes. If a hash algorithm produces a unique fixed-size hash and the original contents of the material cannot be determined from the hash, the hash is considered secure. Common hashing algorithms are the Secure Hash Algorithm, RACE Integrity Primitives Evaluation Message Digest, and Whirlpool.
- Symmetric cryptography, also called private key cryptography, uses a single key to encrypt and decrypt a message. Symmetric cryptography can provide strong protections against attacks if the key is kept secure. Common symmetric cryptographic algorithms include Advanced Encryption Standard, Blowfish, and Twofish.
- Asymmetric cryptography, also known as public key cryptography, uses two keys instead of one. These keys are mathematically related and are known as the public key and the private key. The public key is widely available and can be freely distributed, while the private key is known only to the recipient of the message and must be kept secure. Common asymmetric cryptographic algorithms include RSA, Elliptic Curve Cryptography, and Digital Signature Algorithm. There are also algorithms relating to key exchange.

- Cryptography can be applied through either software or hardware. Software-based cryptography can protect large numbers of files on a system or an entire disk. There are several third-party software tools and modern OSs provide encryption support natively. Cryptography can also be applied to entire disks, known as full-disk encryption (FDE), as well as to volumes and partitions if the partition is properly structured. Applying cryptography to a database (database-level encryption) typically occurs using transparent data encryption (TDE) that executes encryption and decryption within the database engine itself.

- Hardware encryption cannot be exploited like software cryptography. Hardware encryption devices can protect USB devices and standard hard drives. More sophisticated hardware encryption options include self-encrypting drives (SEDs), the hardware security model (HSM), and the Trusted Platform Module (TPM) and secure enclave.

- A blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a business network. At a high level, blockchain technology allows a network of computers to agree at regular intervals on the true state of a distributed ledger, and it is a system in which a record of transactions made are maintained across several computers that are linked in a peer-to-peer network.

- Despite providing these protections, cryptography faces constraints that can impact its effectiveness. Adding cryptography to low-power devices or those that have near instantaneous response times can be a problem because the algorithms require both time and energy, which are typically in short supply for low-power devices and applications needing ultra-fast response times. This results in a resource versus security constraint. Other constraints are speed, size, weak keys, key length, longevity, predictability, reuse, entropy, and computational overhead. Due to the importance of incorporating cryptography in low-power devices, a new subfield of cryptography is being developed called lightweight cryptography.

- There are several types of attacks on cryptography. A known ciphertext attack uses statistical tools to attempt to discover a pattern in the ciphertexts, which then may be useful in revealing the plaintext or key. In a downgrade attack, a threat actor forces the system to abandon the current higher security mode of operation and instead fall back to implementing an older and less secure mode. Many breaches of cryptography are the result not of weak algorithms but instead of misconfigurations. When two files have the same digest this is known as a collision. A collision attack is an attempt to find two input strings of a hash function that produce the same hash result.

- Quantum computing relies on quantum physics using atomic-scale units that can be both 0 and 1 at the same time. As a result, it is possible for one qubit to carry out two separate streams of calculations simultaneously, so that quantum computers will be much faster and more efficient than today's computers. However, quantum computing poses a risk for cryptography. Due to its speed and efficiency, quantum computing may be able to break the foundations of cryptography.

Hands-On Projects

Note - If you are concerned about installing any of the software in these projects on your regular computer, you can instead install the software using the **Windows Sandbox**.

Project 3-1: Using OpenPuff Steganography

Objective: Explore steganography.

Description: Unlike cryptography, which scrambles a message so that it cannot be viewed, steganography hides the existence of the data. In this project, you will use OpenPuff to create a hidden message.

1. Use your web browser to go to **embeddedsw.net/OpenPuff_Steganography_Home.html**. (It is not unusual for websites to change the location of where files are stored. If the URL above no longer works, open a search engine and search for "OpenPuff.")
2. Click **Manual** to open the OpenPuff manual. Save this file to your computer. Read through the manual to see the different features available. Return to the home page when finished.
3. Click **Download binary for Windows/Linux** to download the program. A page will appear asking for payments; however, click the .ZIP link to download the program for evaluation without submitting a payment.
4. Click **Screenshot** to view a screen capture of OpenPuff. Right-click this image and save it as **OpenPuff_Screenshot.jpg** on your computer. This will be the carrier file that will contain the secret message.

Note - For added security, OpenPuff allows a message to be spread across several carrier files.

5. Navigate to the location of the download and uncompress the OpenPuff zip file on your computer.
6. Now create the secret message to be hidden. Open Notepad and enter **This is a secret message**.
7. Save this file as **Message.txt** and close Notepad.
8. Create a zip file from the **Message** file. Navigate to the location of this file through File Explorer and right-click it.
9. Click **Compress to ZIP file** (Windows 11) or click **Send to** and select **Compressed (zipped) folder** (Windows 10) to create the zip file.
10. Navigate to the OpenPuff folder and double-click **OpenPuff.exe**.
11. Click Hide.

Note - Under Bit selection options, note the wide variety of file types that can be used to hide a message.

12. Under (1), create three unrelated passwords and enter them into **Cryptography (A)**, **(B)**, and **(C)**. Be sure that the **Scrambling (C)** password is long enough to turn the **Password check** bar from red to green.
13. Under (2), locate the message to be hidden. Click **Browse** and navigate to the file **Message.zip**. Click **Open**.
14. Under (3), select the carrier file. Click Add and navigate to **OpenPuff_Screenshot.jpg**.
15. Click **Hide Data!**.
16. Navigate to a different location from that of the carrier files and click OK.
17. After the processing has completed, navigate to the location of the carrier file that contains the message and open the file. Can you detect anything different with the file now that it contains the message?
18. Now uncover the message. Close the OpenPuff Data Hiding screen to return to the main menu.

19. Click **Unhide!**.
20. Enter the three passwords.
21. Click **Add Carriers** and navigate to the location of **Carrier1** that contains the hidden message.
22. Click **Unhide!** and navigate to a location to deposit the hidden message. When it has finished processing, click **OK**.
23. Click **Done** after reading the report.
24. Go to the location of the hidden message and you will see **Message.zip**.
25. Close OpenPuff and close all windows.

Project 3-2: Running an RSA Cipher Demonstration

Objective: Observe how RSA creates public and private keys.

Description: The steps for encryption using RSA can be illustrated in a Java applet on a website. In this project, you will observe how RSA encrypts and decrypts.

Note - It is recommended that you review the section earlier regarding the steps in the RSA function.

1. Use your web browser to go to **people.cs.pitt.edu/~kirk/cs1501/notes/rsademo**. (It is not unusual for websites to change the location of where files are stored. If the URL above no longer works, open a search engine and search for "RSA Cipher Demonstration.")
2. Read the information about the demonstration.
3. Click **key generation page**.
4. Change the first prime number (P) to **7**.
5. Change the second prime number (Q) to **5**.
6. Click **Proceed**.
7. Read the information in the pop-up screen and record the necessary numbers. Close the screen when finished.
8. Click **Encryption Page**.
9. Next to **Enter Alice's Exponent key, E**: enter **5** as the key value from the previous screen.
10. Next to **Enter Alice's N Value**: enter **35**.
11. Click **Encrypt**. Read the message and record the values. Close the screen when finished.
12. Click **Decryption Page**.
13. Next to **Enter the encrypted message**: enter **1**.
14. Next to **Enter your N value**: enter **35**.
15. Next to **Enter your private key, D**: enter **5**.
16. Click **Proceed**. Note that **1** has been decrypted to **A**.
17. Close all windows.

Project 3-3: Using 7-Zip Cryptography

Objective: Use file level cryptography.

Description: There are a wide variety of third-party software tools available for performing file-level encryption and decryption. In this project, you will download and use 7-Zip.

1. Use your web browser to go to **www.7-zip.org/index.html**. (If you are no longer able to access the site through the web address, use a search engine and search for "7-zip.")

2. Click the appropriate version and click **Download**.
3. Follow the instructions to install the program.
4. Locate a file or a folder on the computer to use to create an encrypted archive file.
5. Right-click the file. Note that there is now a **7-Zip** option. (In Windows 11, click **Show more options** on the shortcut menu to display the 7-Zip option.)
6. Click 7-Zip.
7. Click **Add to Archive**.
8. The **Add to Archive** dialog box appears. Be sure the Archive format: is **7z**.
9. Under **Encryption**, enter a strong password.
10. Enter the password again under **Reenter password**.
11. Click **OK**.
12. A new encrypted file is created with the extension 7z. Now open this encrypted file by right-clicking it and then clicking **7-Zip**. (In Windows 11, click **Show more options** on the shortcut menu to display the 7-Zip option.)
13. Click **Extract files**.
14. Under **Password**, enter the password and then click **OK**. The encrypted file will be extracted and available for use.
15. Close all windows.

Advanced Cryptography

Despite the clear benefits of cryptography in protecting data, most end-users do not implement it other than what occurs by default. Yet for enterprises, the encrypting (locking | hiding) and decrypting (unlocking | revealing) of data is not only considered essential but in many instances is required by regulations.

However, when cryptography is utilized in the enterprise, a much higher degree of complexity is involved, particularly regarding **keys** (secret codes). What happens if the employee Alice has encrypted an important proposal but suddenly falls ill and cannot return to work? Does only she know the key? Or is there a copy of her key that could be retrieved? Where is this copy of her key stored? Can Bob acquire access to Alice's key? And what happens if Bob is away at a conference? Should multiple employees have key access, or will this weaken security? And how can the keys of hundreds or even thousands of employees be managed?

These and other advanced cryptography issues, particularly in the enterprise, are the topics of this section. First you learn about the **authentication** and **distribution** of **public keys** through digital certificates. Next, you study the **management of keys** through **public key infrastructure**. Finally, you look at different secure communication and transport **protocols** to see the role of cryptography on **data in transit** and how to implement cryptography.

Digital Certificates

One of the common applications of **public key cryptography** is digital certificates. Using digital certificates involves understanding their purpose, knowing how they are managed, and determining which type of digital certificate is appropriate for different situations.

Defining Digital Certificates

Asymmetric cryptography uses a **pair of related keys**. Also known as **public key cryptography**, the **public key** can be distributed and shared with anyone, while the corresponding **private key** must be kept confidential by the owner. Asymmetric cryptography protects the confidentiality of data to ensure that only authorized parties can view it.

Suppose that Alice receives an encrypted document that claims it is from Bob. Because it is encrypted, Alice knows that the document was not viewed by someone else. But how can she *know for certain* that Bob was the sender? Because Alice's public key is widely available, anyone could use it to encrypt a document. Suppose that Mallory created a fictitious document, encrypted it with Alice's public key, and then sent it to Alice while pretending it came from Bob. Alice can use her private key to decrypt the document to read it, but she does not know with absolute certainty who sent it.

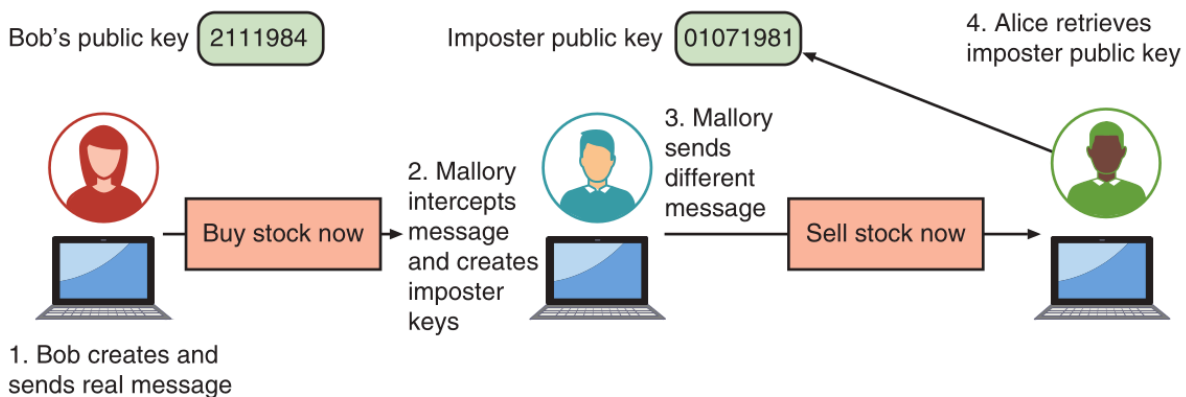
A degree of **proof** can be provided with asymmetric cryptography by creating a digital signature, which is an electronic verification of the sender. After creating a document, Bob generates a **digest (hash)** on it and then **encrypts** the digest with his **private key**, which serves as the **digital signature** (a **special code** attached to a digital document or message that confirms who sent it and that it's still in its original form). Bob sends both the encrypted document and the digital signature to Alice, who **decrypts** the **digital signature**

using Bob's public key, revealing the digest. If she cannot decrypt the digital signature, then she knows that it did not come from Bob (because only Bob's public key can decrypt the digest generated with his private key).

However, there is a weakness with a digital signature: it can only prove the owner of the private key. But that owner could pretend to be someone else (Mallory could pretend to be Bob). It does not necessarily confirm the true identity of the sender because it cannot definitively prove who was the sender of that key. If Alice receives a message with a digital signature claiming to be from Bob, she cannot know for certain that it is the "real" Bob.

For example, suppose Bob created a message along with a digital signature and sent it to Alice. However, Mallory intercepted the message. Mallory then created her own set of public and private keys using Bob's identity. Mallory could create a new message and digital signature (with the imposter private key) and send them to Alice. Upon receiving the message and digital signature, Alice would unknowingly retrieve Mallory's imposter public key (thinking it belonged to Bob) and decrypt it. Alice would be tricked into thinking Bob had sent it when it came from Mallory. This interception and imposter public key are illustrated in Figure 4-1.

Figure 4-1 Imposter public key



Suppose that Bob wants to ensure that Alice receives his real public key and not an imposter public key. He could travel to Alice's city, knock on her front door, and say, "I'm Bob and here's my key."

Yet how would Alice even know this was the real Bob and not Mallory in disguise? For verification, she could ask to see Bob's passport. A passport is a document that is provided by a trusted **third party** who is different from either the sender or receiver. Although Alice may not initially trust Bob because she does not know him, she will trust the government agency that required Bob to provide proof of his identity when he applied for the passport. Using a trusted third party who has verified Bob - and who Alice also trusts - would solve the problem.

This is the concept behind a **digital certificate**. A digital certificate is a technology used to associate a **user's identity** to a **public key** and has been **digitally signed** by a trusted third party. This third party verifies the owner and that the public key belongs to that owner.

A digital certificate is basically a **verified container for a public key**. Typically, a digital certificate contains information such as the owner's name or alias, the owner's public key, the name of the issuer, the digital signature of the issuer, the serial number of the digital certificate, and the expiration date of the public key that has been digitally signed. It can contain other user-supplied information, such as an email address, postal address, and basic registration information.

Note - Digital certificates can be used to identify more than just users; they can authenticate the identity of a **website, organization, device, server, email, and application**.

When Bob sends a message to Alice, he does not ask her to retrieve his public key from a central site. Instead, Bob attaches the digital certificate to the message. When Alice receives the message with the digital certificate, she can check the signature of the trusted third party on the certificate. If the signature was signed by a party that she trusts, then Alice can safely assume that the public key - contained in the digital certificate - is actually from Bob.

Digital certificates make it possible for Alice to verify Bob's claim that the key belongs to him and prevent an attack by Mallory who impersonates the owner of the public key.

Managing Digital Certificates

Several entities and technologies are used to manage digital certificates. These include the certificate authorities and tools for managing certificates.

Certificate Authorities

If a user wants a digital certificate, they must, after creating the public and private keys to be used, complete a request with information such as name, address, and email address. The user electronically signs it by affixing their public key and then sends it to a registration authority that is responsible for verifying the credentials of the applicant. Once verified, it is transferred to an **intermediate certificate authority (intermediate CA)**. The intermediate CA, of which there are many, processes the request and issues the digital certificates. These intermediate CAs perform functions on behalf of a **certificate authority (CA)** that is responsible for digital certificates. This entire process is known as **certificate signing request (CSR) generation**.

Intermediate CAs are subordinate entities designed to handle specific CA tasks such as **processing certificate requests** and **verifying the identity of the individual**. Depending on the type of digital certificate, the person requesting a digital certificate can be authenticated by the following:

- **Email**. In the simplest form, the owner might be identified only by an email address. Although this type of digital certificate might be sufficient for basic email communication, it is insufficient for other verifications.
- **Documents**. A registration authority can confirm the authenticity of the person requesting the digital certificate by requiring specific documentation such as a birth certificate or a copy of an employee badge that contains a photograph.
- **In person**. In some instances, the registration authority might require the applicant to apply in person to prove their existence and identity by providing a government-issued passport or driver's license.

Note - Although the registration function could be implemented directly with the CA, there are advantages to using separate intermediate CAs. If many entities require a digital certificate, or if these are spread out across geographical areas, using a single centralized CA could create bottlenecks or inconveniences. Using multiple intermediate CAs, who can "off-load" these registration functions, can create an improved workflow. This process functions because the CAs trust the intermediate CAs.

Certificate Management

Multiple entities make up strong certificate management. These include a **certificate repository** and a means for **certificate revocation**.

Certificate Repository (CR) A certificate repository (CR) is a publicly accessible centralized directory of digital certificates that can be used to **view the status of a digital certificate**. This directory can be managed locally by setting it up as a **storage area that is connected to the CA server**.

Certificate Revocation Digital certificates normally have an expiration date. However, some circumstances might cause the certificate to be **revoked before it expires**. Some reasons might be benign, such as when the certificate is no longer used or the details of the certificate, such as the user's address, have changed. Other circumstances could be more dangerous. For example, if Mallory were to steal Alice's private key, she could impersonate Alice through using digital certificates without Alice being aware of it.

In addition, what would happen if digital certificates were stolen from a CA? The thieves could then issue certificates to themselves that would be trusted by unsuspecting users. It is important that the CA publishes lists of approved certificates as well as revoked certificates in a timely fashion; otherwise, it could lead to a situation in which security may be compromised.

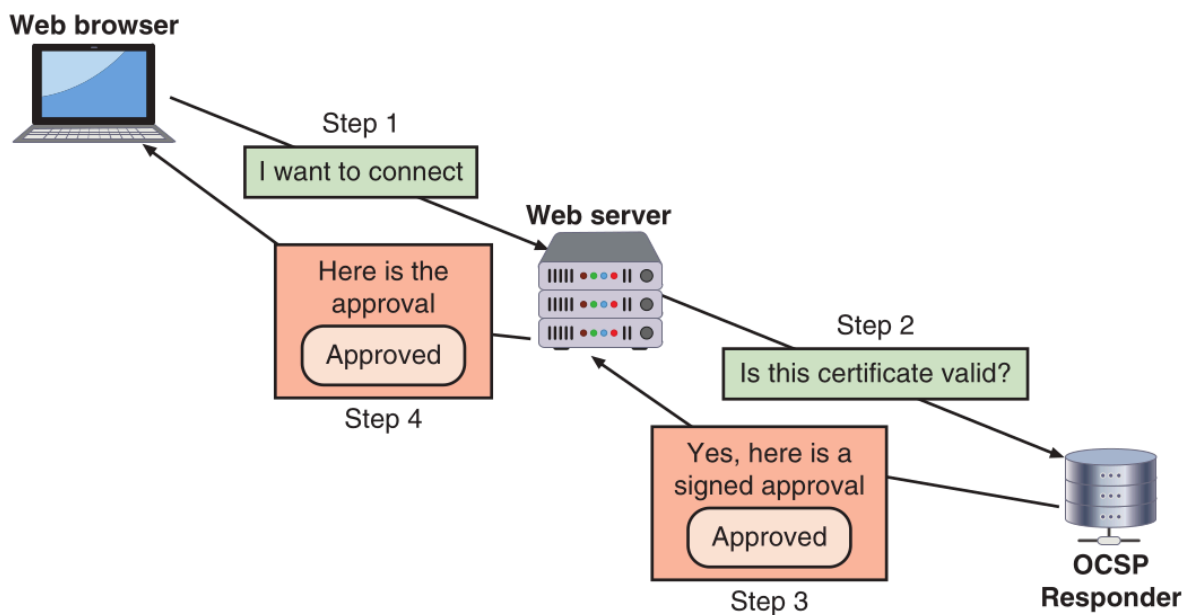
Caution - There have been several incidences of digital certificates being stolen from CAs or intermediate CAs. The thieves can then trick unsuspecting users into connecting with an imposter site, thinking it is a legitimate site. There have also been charges that nation-state actors have stolen digital certificates to trick their own citizens into connecting with fraudulent sites. This allows the actors to monitor the citizen's activities so as to locate dissidents.

There are two means by which the status of a certificate can be checked to see if it has been revoked. The first is to use a **certificate revocation list (CRL)**, which is a list of certificate serial numbers that have been revoked. Many CAs maintain an **online CRL** that can be queried by entering the certificate's serial number. In addition, a local device receives updates to the operating system (OS) or web browser software on the status of certificates and maintains a local CRL.

The second method is the **Online Certificate Status Protocol (OCSP)**, which performs a **real-time lookup of a certificate's status**. OCSP is called a "**request-response**" protocol. The web browser sends the certificate's information to a trusted entity like the CA, known as an **OCSP Responder**. The OCSP Responder then provides revocation information on that one specific certificate.

A variation of OCSP is called **OCSP stapling**. OCSP requires the OCSP Responder to provide responses to every web client of a certificate in real time, which may create a high volume of traffic. With OCSP stapling, web servers send queries to the Responder OCSP server at regular intervals to receive a signed, time-stamped OCSP response. When a client's web browser attempts to connect to the web server, the server can include (staple) in the handshake with the web browser the previously received OCSP response. The browser then can evaluate the OCSP response to determine if it is trustworthy. OCSP stapling is illustrated in Figure 4-2 below.

Figure 4-2 OCSP stapling



Determining the revocation status of certificates presented by websites is an ongoing problem in web security. Initially, web browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera) used OCSP. However, if the web browser cannot reach the OCSP Responder server, such as when the server is down, then the browser receives the message that there is a network error (called a “soft-fail”) and the revocation check is simply ignored. Also, online revocation checking by web browsers can be slow. For these reasons, modern web browsers have implemented a range of solutions to reduce or eliminate the need for online revocation checking by instead “harvesting” lists of revoked certificates from CAs and then pushing them to the user’s browser. Table 4-2 lists the solutions used by selected web browsers for determining the revocation status of certificates.

Table 4-2 Web browser certificate revocation procedures

Browser	Procedure name	Description	Resources
Google Chrome	CRLSet	CRLSet is a list of revoked certificates determined by searching CRLs published by CAs that is pushed to the browser as a software update.	Users can view the CRLSet version and manually check for updates at chrome://components .
Mozilla Firefox	OneCRL	OneCRL is list of intermediate certificates that have been revoked by CAs and then pushed to Firefox in application updates; the browser can also be configured to query OCSP responders.	Lists of OneCRL can be viewed as webpages at https://crt.sh/mozilla-onecrl .
Apple Safari	No name	Apple collects revoked certificates from CAs that are then periodically retrieved by Apple devices; when a revoked certificate request occurs, Safari performs an OCSP check to confirm.	Users can view blocked certificates at https://support.apple.com/en-us/HT209144#blocked .
Microsoft Edge	CRLSet	Windows OS checks for server certificate revocation and Edge relies on this listing.	Can be viewed through Windows or in the Edge browser at edge://settings/privacy .

Types of Digital Certificates

There are several types of digital certificates. These can be grouped into the broad categories of **root certificates**, **domain certificates**, and **hardware and software certificates**. In addition, there are standardized certificate formats and attributes.

Root Digital Certificates - Suppose that Alice is making a purchase at an online ecommerce site and needs to enter her credit card number. How can she be certain that she is at the authentic website of the retailer and not an imposter's look-alike site that will steal her credit card number? The solution is a **digital certificate**. The online retailer's web server issues to Alice's web browser a digital certificate that has been signed by a trusted third party. In this way, Alice can rest assured that she is at the authentic online retailer's site.

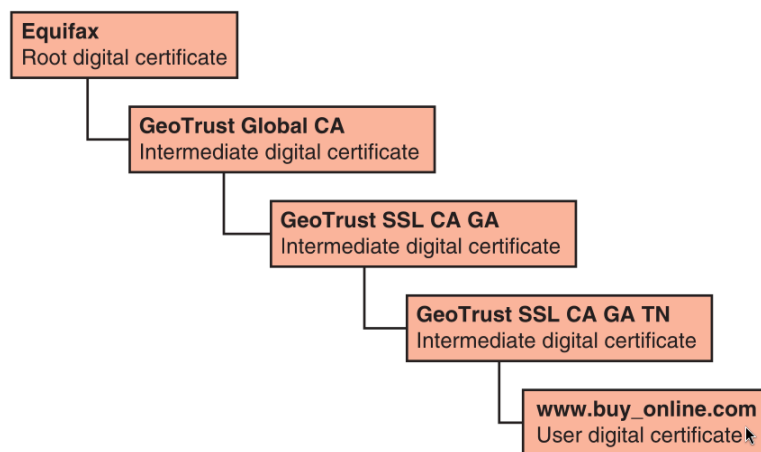
However, by some estimates, there are 24 million ecommerce sites. In addition, there are websites for banks, credit card companies, schools, and workplaces, just to name a few of the other websites that need to provide protection to its customers and clients. How can each of these digital certificates be verified as being authentic and not expired or revoked, and how should they be organized?

Note - In 2016, the nonprofit **Internet Security Research Group (ISRG)** created "Let's Encrypt," which is advertised as a free, automated, and open CA run for the public's benefit. Their goal is to provide free digital certificates to any website with the stated reason "because we want to create a more secure and privacy-respecting web." Of the over 309 million domains protected by Let's Encrypt, the "daily issuance" (how many digital certificates are sent to users) averages 2.5 million with over 3 billion total digital certificates sent since it started.

Grouping and verifying digital certificates relies on **certificate chaining**. Certificate chaining creates a **path** between the trusted "root" **CAs (of which there are a few) and the intermediate CAs (of which there are many)** with the digital certificates that have been issued. Every certificate is signed by the entity that is identified by the next higher certified entity in the chain. In this way, the trust of a certificate can be traced back to the highest level of CA. This is known as the **root of trust**.

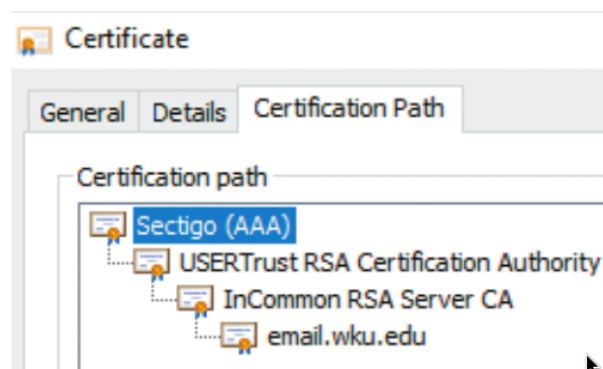
The beginning point of the chain is a specific type of digital certificate known as a root digital certificate. A root digital certificate is created and verified by a CA. Because there is no higher-level authority than a CA, root digital certificates are **self-signed** and do not depend on any higher-level authority for authentication. The next level down from the root digital certificate is one or more **intermediate certificates** that have been issued by intermediate CAs. The root digital certificate (verified by a CA) trusts the intermediate certificate (verified by an intermediate CA), which may in turn validate more lower-level intermediate CAs until it reaches the end of the chain. The endpoint of the chain is the **user digital certificate** itself. The path of certificate chaining is illustrated in Figure 4-3 below.

Figure 4-3 Path of certificate chaining



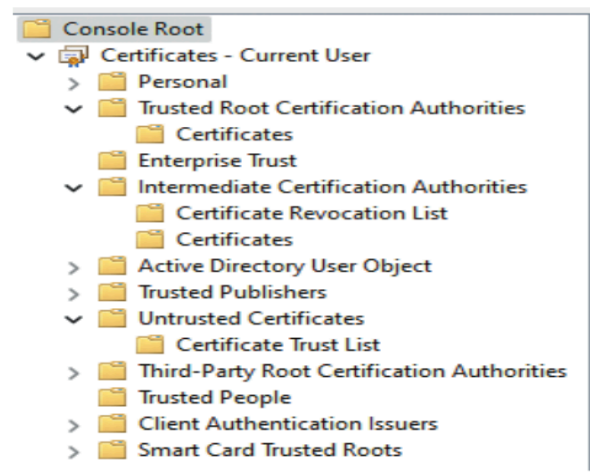
An example of certificate chaining is shown in Figure 4-4 below.

Figure 4-4 Example of certificate chaining



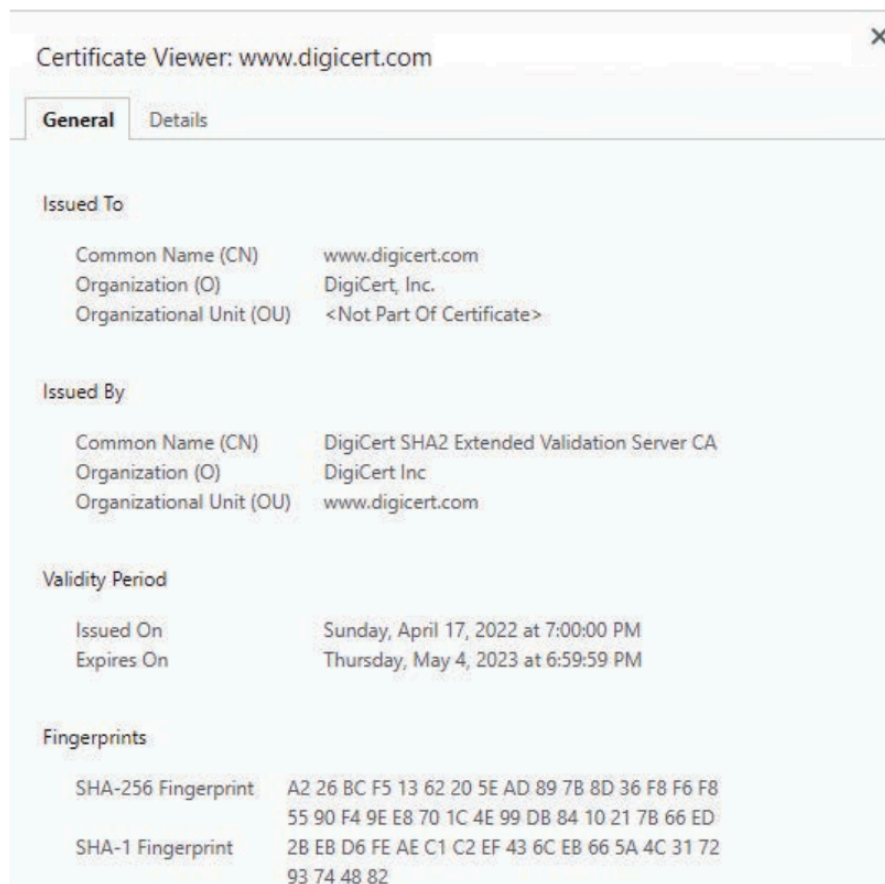
Approved root digital certificates and intermediate certificates are distributed in one of three ways. **First**, they can be distributed through updates to the OS. Trusted root CAs, intermediate CAs, and untrusted certificates can be viewed through the OS. Figure 4-5 below illustrates these certificates in Microsoft Windows 11, which includes trusted root CAs, intermediate CAs, CRLs, and untrusted certificates.

Figure 4-5 Microsoft Windows certificates



A **second** means is that certificates can be distributed through **updates** to the web browser. At one time, browsers relied on the underlying OS-approved list, but today many rely on their own browser updates. Information pertaining to a digital certificate can be seen in the browsers and is illustrated in Figure 4-6 below.

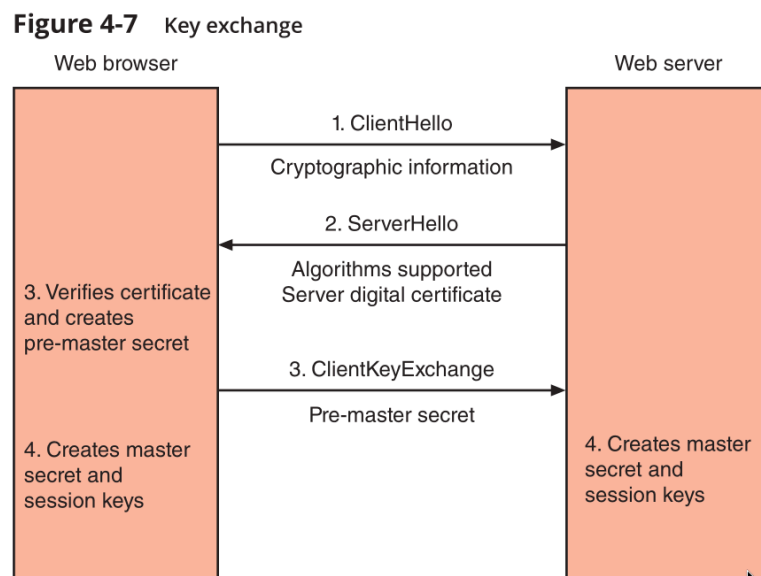
Figure 4-6 Web browser digital certificate information



A **third** option is **pinning**, in which a digital certificate is **hard-coded (pinned)** within the app or program that is using the certificate. Pinning is common for securing mobile messaging apps and for certain web-based services and browsers.

Note - The consequences of a compromised root CA are very significant because a breach could likewise taint all its intermediate CAs along with all the digital certificates that they issued. This makes it essential that all CAs must be kept safe from unauthorized access. A common method to ensure the security and integrity of a root CA is to keep it in an **offline state** from the network (offline CA) and even “powered down.” It is only brought online when needed for specific and infrequent tasks, typically limited to the issuance or re-issuance of certificates authorizing intermediate CAs.

Domain Digital Certificates - Most digital certificates are web server digital certificates that are issued from a web server to a device. These web server digital certificates perform **two primary functions**: they ensure the **authenticity** of the web server to the client and the **authenticity** of the **cryptographic connection to the web server**. Web servers can set up secure cryptographic connections by providing the server’s public key with a digital certificate to the client. This handshake setup between web browser and web server, also called a key exchange, is illustrated in Figure 4-7:



1. The web browser sends a message (“ClientHello”) to the server that contains information including the list of cryptographic algorithms that the client supports.
2. The web server responds (“ServerHello”) by indicating which cryptographic algorithm will be used. It then sends the server digital certificate to the browser.
3. The web browser verifies the server certificate (such as making sure it has not expired) and extracts the server’s public key. The browser generates a random value (called the pre-master secret), encrypts it with the server’s public key, and sends it back to the server (“ClientKeyExchange”).
4. The server decrypts the message and obtains the browser’s pre-master secret. Because both the browser and server now have the same pre-master secret, they can each create the same master secret. The master secret is used to create session keys, which are symmetric keys to encrypt and decrypt information exchanged during the session and to verify its integrity.

Several types of domain digital certificates address the security of web server digital certificates. These include domain validation digital certificates, extended validation digital certificates, wildcard digital certificates, and subject alternative names digital certificates.

Domain Validation. Some entry-level certificates provide domain-only validation to authenticate that only a specific organization has the right to use a particular domain name. A domain validation digital certificate verifies the identity of the entity that has control over the domain name. These certificates indicate nothing regarding the trustworthiness of the individuals behind the site; they simply verify who has control of that domain.

Note - Because domain validation digital certificates are not verifying the identity of a person but only the control over a site, they often can be generated automatically and are very inexpensive or even free.

Extended Validation (EV). An enhanced type of domain digital certificate is the Extended Validation (EV) certificate. This type of certificate requires more extensive verification of the legitimacy of the business. Requirements include:

- The intermediate CA must pass an independent audit verifying that it follows the EV standards.
- The existence and identity of the website owner, including its legal existence, physical address, and operational presence, must be verified by the intermediate CA.
- The intermediate CA must verify that the website is the registered holder and has exclusive control of the domain name.
- The authorization of the individual(s) applying for the certificate must be verified by the intermediate CA, and a valid signature from an officer of the company must accompany the application.

Wildcard. A wildcard digital certificate is used to validate a main domain along with all subdomains. For example, a domain validation digital certificate for `www.example.com` would only cover that specific site. A wildcard digital certificate for `*.example.com` would cover `www.example.com`, `mail.example.com`, `ftp.example.com`, and any other subdomains.

Subject Alternative Name (SAN). A limitation of wildcard digital certificates is that while they can protect all first-level subdomains on an entire domain, they cannot apply to different domains, such as `www.example.com` and `www.example.org`. And hosting multiple websites on a single server typically requires a unique Internet Protocol (IP) address per site.

The **Subject Alternative Name (SAN)** solves these problems. This certificate allows different values to be associated with a single certificate. A SAN allows a single digital certificate to specify additional host names (sites, common names, etc.) to be protected by that one certificate. It also permits a certificate to cover multiple IP addresses. This can greatly simplify a server's domain name certificate configuration: instead of configuring multiple IP addresses on a server and then "binding" each IP address to a different certificate, a single SAN can instead cover all the addresses.

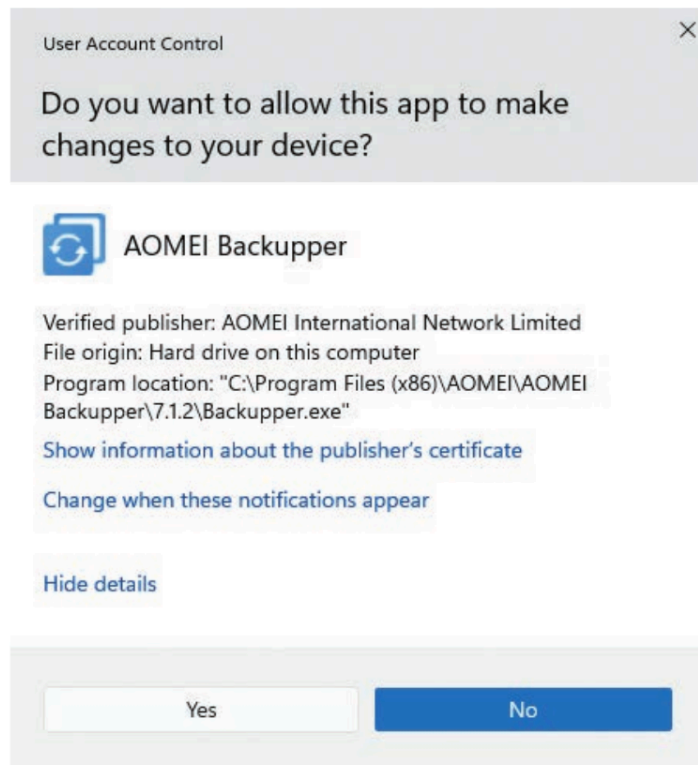
Hardware and Software Digital Certificates - In addition to root digital certificates and domain digital certificates, more specific digital certificates relate to hardware and software. These include the following:

- **Machine/computer digital certificate.** A machine/computer digital certificate is used to verify the identity of a device in a network transaction. For example, a printer may use a machine digital certificate to verify to the endpoint that it is an authentic and authorized device on the network.

Note - Many network devices can create their own self-signed machine digital certificates.

- Code signing digital certificate. Digital certificates are used by software developers to digitally sign a program to prove that the software comes from the entity that signed it and no unauthorized third party has altered or compromised it. This is known as code signing, and it produces a code signing digital certificate. When the installation program is launched that contains a code signed digital certificate, a pop-up window appears. Clicking the Show more details link will display Verified publisher as seen in Figure 4-8 below.

Figure 4-8 Verified publisher message



An installation program that lacks a code digital certificate will display a window with the warning **Publisher: Unknown**.

- Email digital certificate. An email digital certificate allows a user to digitally sign and encrypt mail messages. Typically, only the user's name and email address are required to receive this certificate.

Note - In addition to email messages, digital certificates can also be used to authenticate the authors of documents. For example, a user can create a Microsoft Word or Adobe Portable Document Format (PDF) document and then use a digital certificate to create a digital signature.

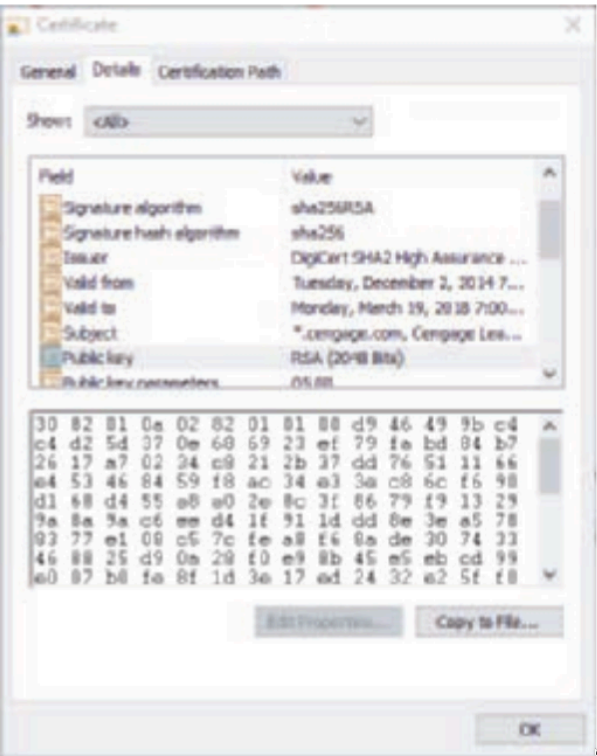
Digital Certificate Attributes and Formats

Hardware devices require that digital certificates contain specific attributes (fields) and are presented in a specific format. This allows the device to read and process the digital certificate. The standard format for digital certificates is **X.509 Version 3**. Digital certificates following this standard can be read or written by any hardware device or application that follows the X.509 format.

Several certificate attributes make up an X.509 digital certificate. These attributes are used when the parties negotiate a secure connection. Attributes that must be included are the **certificate validity period**,

end-host identity information, encryption keys that will be used for secure communications, the signature of the issuing CA, and the **common name (CN)**. CN is the name of the device protected by the digital certificate. The CN can reference a single device (www.example.com) or multiple devices with a wildcard certificate (*.example.com) but is not the URL (https://example.com). Other optional attributes may also be included. Figure 4-9 illustrates several attributes in a digital certificate.

Figure 4-9 Digital certificate attributes



Note - X.509 certificates can either be contained in a binary file with a .cer extension or in a Base64 file, which is a binary-to-text encoding scheme that presents binary data in ASCII string format.

Public Key Infrastructure (PKI)

One of the important management tools for the use of digital certificates and asymmetric cryptography is public key infrastructure. This involves defining public key infrastructure, and knowing its trust models, how it is managed, and the features of key management.

What Is Public Key Infrastructure (PKI)?

Suppose that Alice wants to obtain a digital certificate for her personal use. She must go through the entire CSR generation process that involves multiple steps and interacting with multiple entities. She must use asymmetric cryptography to create her public and private keys, a registration authority must verify the request, an intermediate CA must process the request, the digital certificate must be placed in a CR, and then the CR finally moved to a CRL when it expires. For one user creating a digital certificate through the CSR generation process, this involves multiple steps interacting with multiple entities.

Public key infrastructure (PKI) is what you might expect from its name: it is the underlying infrastructure that serves as a **key management system** for controlling public keys, private keys, and digital certificates. Strictly speaking, PKI is the set of software, hardware, processes, procedures, and policies that are needed to create, manage, distribute, use, store, and revoke digital certificates across large user populations. The goal of a PKI is to **establish the identity** of people, devices, and services in order to **control** access to resources, **protect** data, and provide **accountability**. In short, PKI is digital certificate management at scale.

Trust Models

Trust is defined as **confidence** in or **reliance** on another person or entity. One of the principal foundations of PKI is that of **trust**: Alice must trust that the public key in Bob's digital certificate belongs to him.

A **trust model** refers to the type of **trust relationship** that can exist between individuals or entities. In one type of trust model, **direct trust**, a relationship exists between two individuals because one person knows the other person. Because Alice knows Bob - she has seen him, she can recognize him in a crowd, she has spoken with him - she can trust that the digital certificate that Bob personally gives her contains his public key.

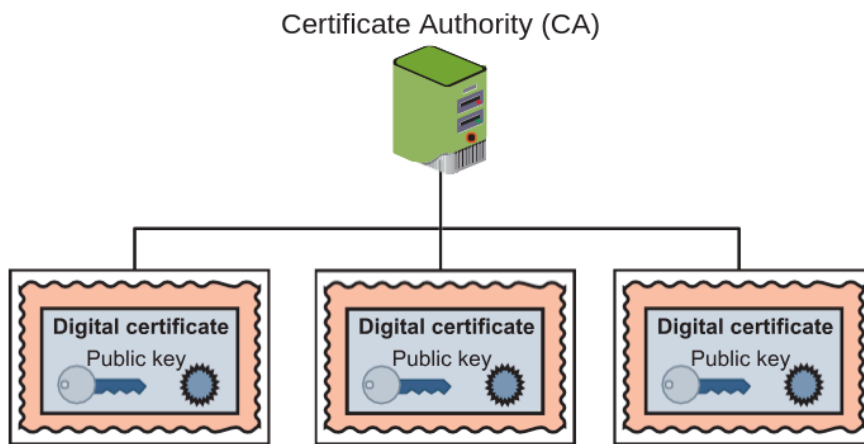
A **third-party trust** refers to a situation in which two individuals trust each other because each **trusts a common third party**. An example of a third-party trust is a courtroom. Although the defendant and prosecutor may not trust one another, they both can trust the judge (a third party) to be fair and impartial. In that case, they implicitly trust each other because they share a common relationship with the judge. In terms of PKI, if Alice does not know Bob, this does not mean that she can never trust his digital certificate. Instead, if she trusts a third-party entity who knows Bob, then she can trust that the digital certificate with the public key is Bob's.

A less secure trust model that uses no third party is called the **web of trust** model and is based on **direct trust**. Each user signs a digital certificate and then exchanges certificates with all other users. Because all users trust each other, each user can sign the certificate of all other users.

Essentially three PKI trust models use a CA. These are the **hierarchical** trust model, the **distributed** trust model, and the **bridge** trust model.

Hierarchical Trust Model: The hierarchical trust model assigns a single hierarchy with one master CA called the **root**. This root signs all digital certificate authorities with a single key. A hierarchical trust model is illustrated in Figure 4-10 below.

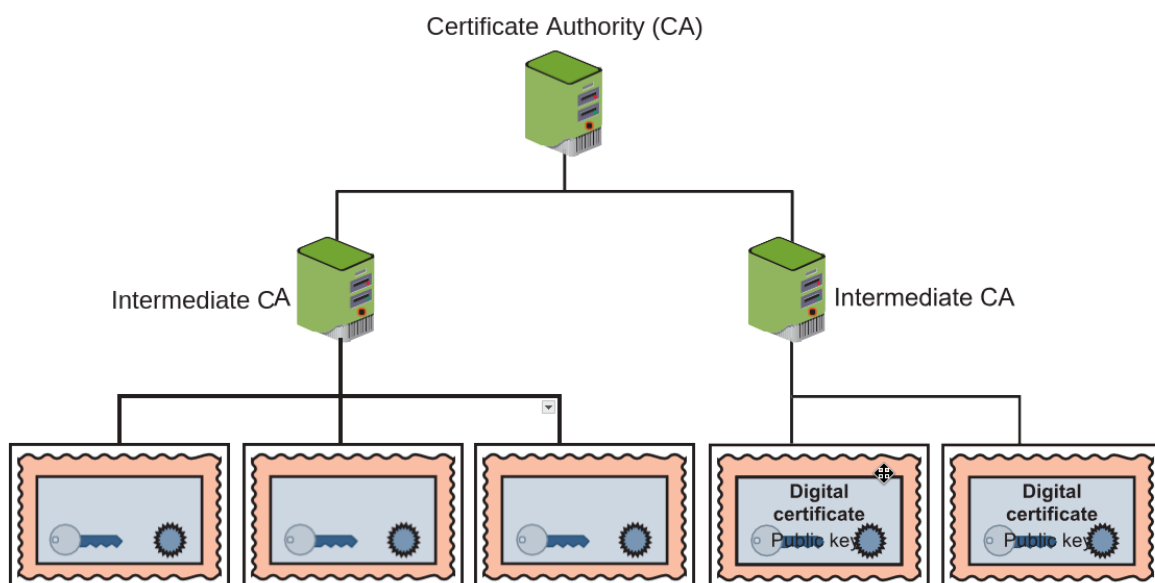
Figure 4-10 Hierarchical trust model



A hierarchical trust model can be used in an organization where one CA is responsible for only the digital certificates for that organization. However, on a larger scale, a hierarchical trust model has several limitations. First, if the CA's single private key were to be compromised, then all digital certificates would be worthless. Also, having a single CA that must verify and sign all digital certificates may create a significant backlog.

Distributed Trust Model: Instead of having a single CA, as in the hierarchical trust model, the distributed trust model has **multiple CAs** that sign digital certificates. This essentially eliminates the imitations of a hierarchical trust model. The loss of a CA's private key would compromise only those digital certificates it had signed, and the workload of verifying and signing digital certificates can be distributed. In addition, these CAs can delegate authority to other intermediate CAs to sign digital certificates. The distributed trust model is the basis for most digital certificates used on the **Internet**. A distributed trust model is illustrated in Figure 4-11.

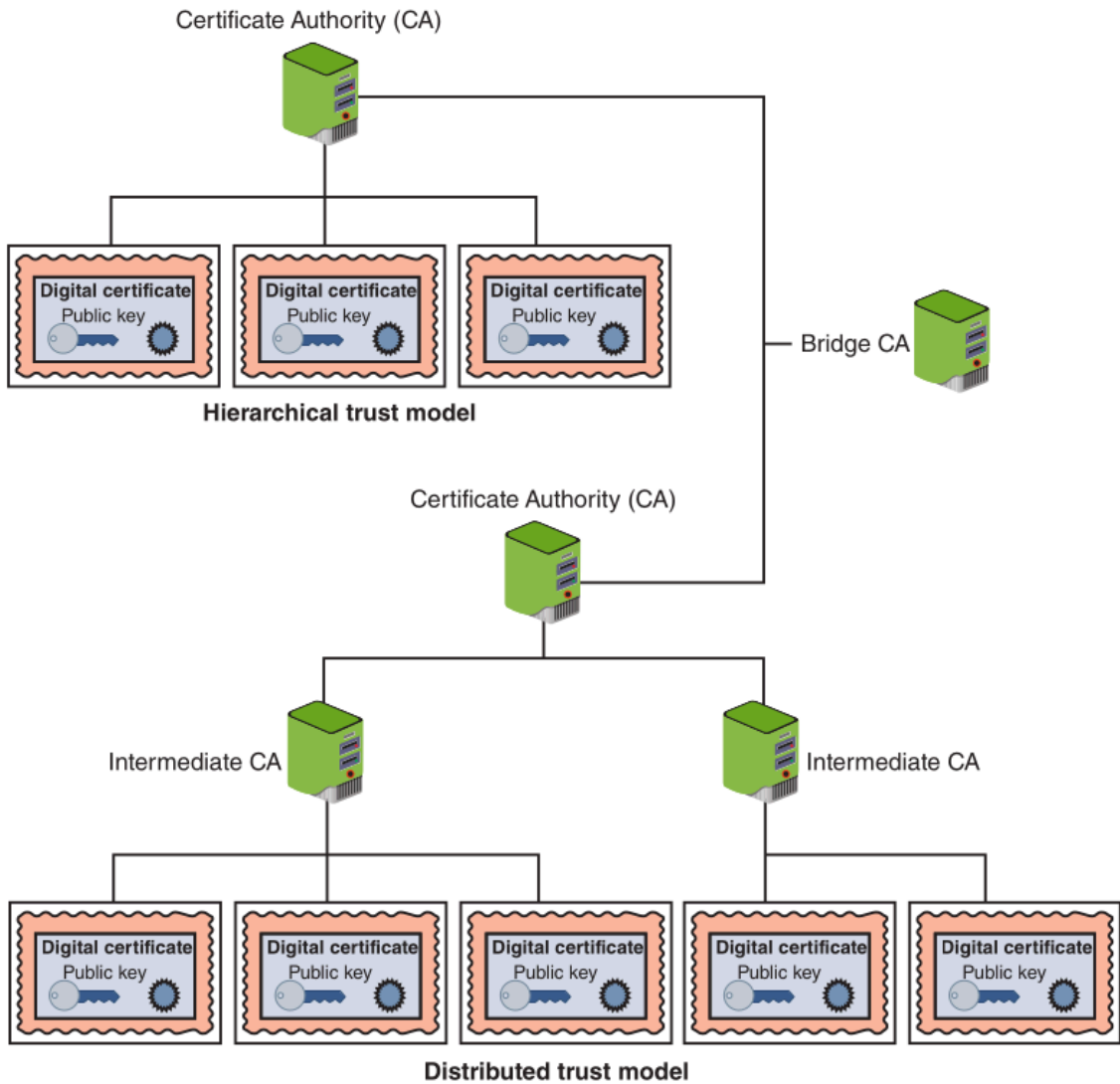
Figure 4-11 Distributed trust model



Bridge Trust Model: The bridge trust model is similar to the distributed trust model in that no single CA signs digital certificates. However, with the bridge trust model, one CA acts as a facilitator to interconnect all other

CAs. This facilitator CA does not issue digital certificates; instead, it acts as the hub between hierarchical trust models and distributed trust models. This allows the different models to be linked together. The bridge trust model is shown in Figure 4-12 below.

Figure 4-12 Bridge trust model



Managing PKI

An organization that uses multiple digital certificates on a regular basis will need to properly manage those digital certificates. This includes **establishing policies and practices** and determining the life cycle of a digital certificate.

Certificate Policy (CP): A **certificate policy (CP)** is a published set of rules that govern the operation of a PKI. The CP provides **recommended baseline security requirements** for the use and operation of CA, intermediate CA, and other PKI components. A CP should cover such topics as CA or intermediate CA obligations, user obligations, confidentiality, operational requirements, and training.

Certificate Practice Statement (CPS): A **certificate practice statement (CPS)** is a more technical document than a CP. A CPS describes in detail how the CA uses and manages certificates. Additional topics for a CPS include how end-users register for a digital certificate, how to issue digital certificates, when to revoke digital certificates, procedural controls, key pair generation and installation, and private key protection.

Certificate Life Cycle

Digital certificates do not last forever: employees leave, new hardware is installed, applications are updated, and cryptographic standards evolve. Each of these changes affects the usefulness of a digital certificate. The life cycle of a certificate is typically divided into four parts:

- 1. Creation.** At this stage, the certificate is created and issued to the user. Before the digital certificate is generated, the user must be positively identified. The extent to which the user's identification must be confirmed can vary, depending on the type of certificate and any existing security policies. Once the user's identification has been verified, the request is sent to the CA for a digital certificate. The CA can then apply its appropriate signing key to the certificate, effectively signing the public key. The relevant fields can be updated by the CA, and the certificate is then forwarded to the registration authority. The CA can also keep a local copy of the certificate it generated.
- 2. Suspension.** This stage could occur once or multiple times throughout the life of a digital certificate if the certificate's validity must be temporarily suspended. This may occur, for example, when an employee is on a leave of absence. During this time, it may be important that the user's digital certificate not be used for any reason until they return. Upon the user's return, the suspension can be withdrawn or the certificate can be revoked.
- 3. Revocation.** At this stage, the certificate is no longer valid. Under certain situations a certificate may be revoked before its normal expiration date, such as when a user's private key is lost or compromised. When a digital certificate is revoked, the CA updates its internal records and any CRL with the required certificate information and timestamp (a revoked certificate is identified in a CRL by its certificate serial number). The CA signs the CRL and places it in a public repository so that other applications using certificates can access this repository to determine the status of a certificate.
- 4. Expiration.** At the expiration stage, the certificate can no longer be used. Every certificate issued by a CA must have an expiration date. Once it has expired, the certificate may not be used any longer for any type of authentication and the user will be required to follow a process to be issued a new certificate with a new expiration date.

Key Management

One common vulnerability that allows threat actors to compromise a PKI is improper certificate and key management. Because keys form the foundation of PKI systems, they must be carefully managed. Proper key management includes **key storage, key usage, and key handling** procedures.

Key Storage

The means of storing keys in a PKI system is important. Public keys can be stored by embedding them within digital certificates, while private keys can be stored on the user's local system. The drawback to software-based storage is that it can leave keys open to attacks; vulnerabilities in the client operating system, for example, can expose keys to attackers.

Storing keys in hardware is an alternative to software-based storage. For storing public keys, special CA root and intermediate CA hardware devices can be used. Private keys can be stored on smart cards or in tokens.

Caution - Whether private keys are stored in hardware or software, they must be adequately protected. To ensure basic protection, never share the key in plaintext, always store keys in files or folders that are themselves password protected or encrypted, do not make copies of keys, and destroy expired keys.

Key Usage

If more security is needed than a single set of public and private keys, multiple pairs of dual keys can be created. One pair of keys may be used to encrypt information, and the public key can be backed up to another location. The second pair would be used only for digital signatures, and the public key in that pair would never be backed up.

Key Handling Procedures

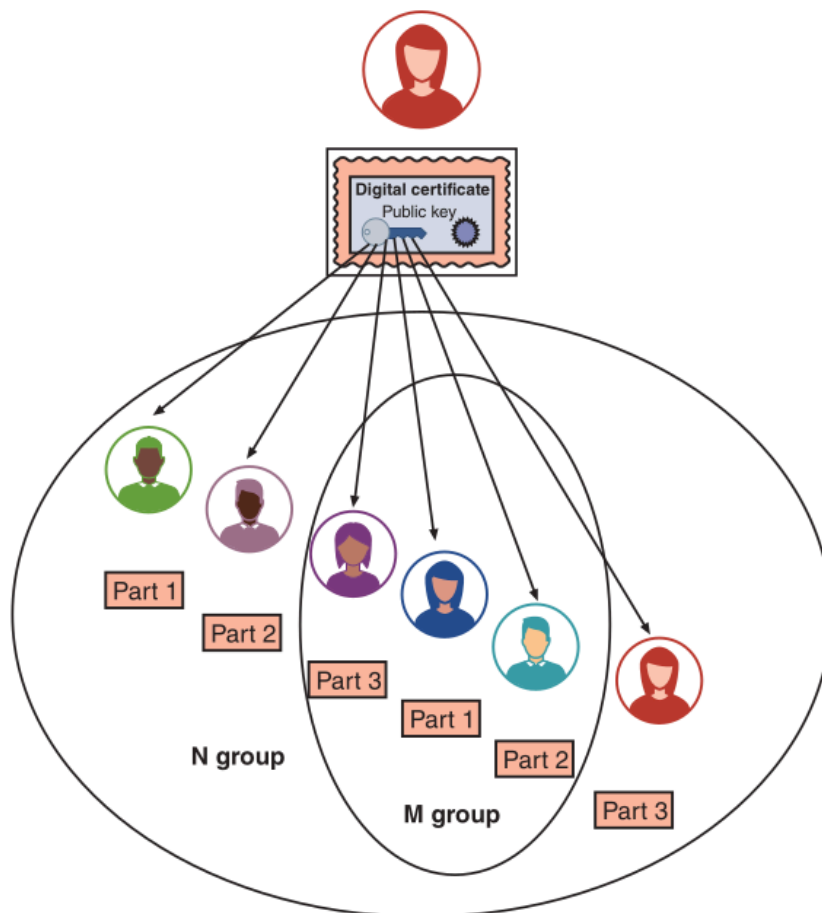
Certain procedures can help ensure that keys are properly handled. These procedures include:

- **Escrow. Key escrow** refers to a process in which keys are managed by a third party, such as a trusted CA. In key escrow, the private key is split and each half is encrypted. The two halves are registered and sent to the third party, which stores each half in a separate location. A user can then retrieve the two halves, combine them, and use this new copy of the private key for decryption. Key escrow relieves the end-user from the worry of losing their private key. The drawback to this system is that after the user has retrieved the two halves of the key and combined them to create a copy of the key, that copy of the key can be vulnerable to attacks.
- **Expiration.** Keys have an expiration date after which they cease to function. This prevents an attacker who may have stolen a private key from being able to decrypt messages for an indefinite period. Some systems set keys to expire after a set period by default.
- **Renewal.** Instead of letting a key expire and then creating a new key, an existing key can be renewed. With renewal, the original public and private keys can continue to be used and new keys do not have to be generated. However, continually renewing keys makes them more vulnerable to theft or misuse.
- **Revocation.** Whereas all keys should expire after a set period, a key may need to be revoked prior to its expiration date. For example, the need for revoking a key may be the result of an employee being terminated from their position. Revoked keys cannot be

reinstated. The CA should be immediately notified when a key is revoked and then the status of that key should be entered on the CRL.

- **Recovery.** What happens if an employee is hospitalized, yet their organization needs to transact business using their keys? Different techniques may be used. Some CA systems have an embedded key recovery system in which a key recovery agent (KRA) is designated, a highly trusted person responsible for recovering lost or damaged digital certificates. Digital certificates can then be archived along with the user's private key. If the user is unavailable or if the certificate is lost, the certificate with the private key can be recovered. Another technique is known as **M-of-N control**. A user's private key is encrypted and divided into a specific number of parts, such as three. The parts are distributed to other individuals, with an overlap so that multiple individuals have the same part. For example, the three parts could be distributed to six people, with two people each having the same part. This is known as the N group. If it is necessary to recover the key, a smaller subset of the N group, known as the M group, must meet and agree that the key should be recovered. If a majority of the M group can agree, they can then piece the key together. M-of-N control is illustrated in Figure 4-13 below.

Figure 4-13 M-of-N control



Note - The reason for distributing parts of the key to multiple users is that the absence of one member would not prevent the key from being recovered.

- **Suspension.** The revocation of a key is permanent; key suspension is for a set period. For example, if an employee is on an extended medical leave, it may be necessary to suspend the use of their key for security reasons. A suspended key can be later reinstated. As with revocation, the CA should be immediately notified when a key is suspended, and the status of that key should be checked on the CRL to verify that it is no longer valid.
- **Destruction.** Key destruction removes all private and public keys along with the user's identification information in the CA. When a key is revoked or expires, the user's information remains on the CA for audit purposes.

Secure Communication and Transport Protocols

In addition to protecting data in use and data at rest, cryptographic algorithms are used to protect data in transit (**transport/communication encryption**). There are different secure communication and transport protocols based on cryptographic algorithms for protecting data in transit. These protocols typically rely on “encapsulating” or enveloping the data to be transmitted inside something else.

Consider a road that must be built between two cities. However, a tall mountain separates the cities, and it is not practical or economically feasible to build the road over the mountain. As an alternative, a tunnel can be bored through the mountain to create a road, as seen in Figure 4-14.

Figure 4-14 Tunnel through a mountain



In a similar fashion, private data can be sent through a public network in which it normally could be visible to prying eyes. A private tunnel can “bore through” a public network to **hide its contents**. Conceptually,

first the data is **encrypted** and then it is **encapsulated**. Encapsulation is the process of enclosing the encrypted data with an additional header so that the data can be directed to its destination. If the data were to be completely encrypted, including its header, then network routers would not be able to forward the packet to its destination since they do not have the decryption key to view the header. By encapsulating the encrypted information with unencrypted information, the data can be protected and travel across the networks as normal. This technique for transporting data securely across a network is called **tunneling**.

Note - Tunneling refers to the entire process of encapsulation, transmission, and “de-encapsulation.” Encapsulation itself is only a single step within the entire process.

Two primary secure communication and transport protocols use tunneling. These are **Transport Layer Security (TLS)** and **IP Security (IPSec)**. In addition, other secure protocols can provide protection.

Transport Layer Security (TLS)

One of the early and most widespread secure communication and transport protocols was **Secure Sockets Layer (SSL)**. This protocol was developed by Netscape in 1994 in response to the growing concern over Internet security. The design goal of SSL was to create an encrypted data path between a client and a server that could be used on any platform or operating system. However, the way SSL functioned left it vulnerable to attack.

As a replacement for SSL, the **Transport Layer Security (TLS)** provides a higher degree of protection. The **current version of TLS, v1.3**, is considered a significant upgrade over TLS v1.2.

Caution - Although the algorithms SSL and TLS are sometimes listed as being interchangeable or even in conjunction with each other (“TLS/SSL”), this is not correct. They are different, and SSL is deprecated and is being replaced by TLS.

A **cipher suite** is a named combination of the **encryption, authentication, and message authentication code (MAC) algorithms** that is used with **TLS**. These are negotiated between the web browser and web server during the initial connection handshake. **Cipher suites** typically use descriptive names to indicate their components. For example, **TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256** specifies that TLS is the protocol, during the handshake keys will be exchanged via the ephemeral Elliptic Curve Diffie Hellman (ECDHE), AES running the Galois Counter Mode with 128-bit key size is the encryption algorithm, and SHA-256 is the hashing algorithm.

IP Security (IPSec)

A more comprehensive communication and transport security protocol is IP Security. It is important to know what IP Security is and how it can be implemented.

Definition

Internet Protocol Security (IPSec) is a group or suite of protocols for securing Internet Protocol (IP) communications. IPSec encrypts and authenticates each IP packet of a session between devices.

Note - IPSec is not a single protocol but a framework of multiple protocols, encryption methods, authentication processes, and cryptographic algorithms.

IPSec is considered to be a transparent security protocol. It is transparent to the following entities:

- **Applications.** Programs do not have to be modified to run under IPSec.

- **Users.** Unlike some security tools, users do not need to be trained on specific security procedures, such as encrypting with a specific application.
- **Software.** Using IPsec does not require that software changes to an application must be made on the local device.

IPsec provides three areas of protection that correspond to three IPsec protocols:

- 1- Authentication.** IPsec authenticates that packets received were sent from the source. This is identified in the header of the packet to ensure that no specific attacks took place to **alter the contents** of the packet. This is accomplished by the **Authentication Header (AH)** protocol.
- 2- Confidentiality.** By encrypting the packets, IPsec ensures that no other parties could view the contents. Confidentiality is achieved through the **Encapsulating Security Payload (ESP)** protocol. ESP supports authentication of the sender and encryption of data.
- 3- Key management.** IPsec manages the keys to ensure that they are not intercepted or used by unauthorized parties. For IPsec to work, the sending and receiving devices must share a key. This is accomplished through a protocol known as **Internet Security Association and Key Management Protocol/Oakley (ISAKMP/Oakley)**, which generates the key and authenticates the user using techniques such as digital certificates.

IPsec supports two encryption modes: **transport** and **tunnel**. **Transport mode** encrypts only the data portion (**payload**) of each packet yet leaves the header unencrypted. The more secure **tunnel mode** encrypts both the header and the data portion. IPsec accomplishes transport and tunnel modes by adding new headers to the IP packet. The entire original packet (header and payload) is then treated as the data portion of the new packet.

Note - Because tunnel mode protects the entire packet, it is generally used in a network-to-network communication, while transport mode is used when a device must see the source and destination addresses to route the packet.

IPsec is considered a more robust protocol than TLS. This is because it provides security to IP, which is the basis for all other TCP/IP protocols. In protecting IP, IPsec is essentially protecting everything else in TCP/IP as well. Table 4-3 lists some of the differences between IPsec and TLS.

Table 4-3 IPSec versus TLS

	IPSec	TLS
Definition	A set of protocols that provide security for all IP traffic by directly encrypting IP packets	A secure protocol developed for protecting specific information over the Internet
OSI layer	Layer 3	Layers 4–7
Installation	Generic and requires full client installed on device	Vendor specific and capabilities incorporated within all web browsers, so no separate client needed
Configuration	Complex	Basic
Changes to protocol	Depends on implementation but no need to change application	Must change application but not OS
Protections	All IP-based applications	Web-enabled applications, file sharing, and email

Architectures and Implementation of IPSec

How exactly can IPSec be integrated into IP? The following are different methods for deploying IPSec:

- **Devices.** Installing IPSec into all devices provides the highest security because it provides “end-to-end” security between all devices on the network. However, installing it on a network with a large number of devices requires more work. Another option is to install it only on routers through which network data passes. While this gives protection only for the portion of the route that the data takes outside the enterprise and may be sufficient for certain applications, the connections between routers and local devices would not be secured.
- **Integration into TCP/IP.** Ideally IPSec would be infused directly into IP itself so that all IPSec security modes and capabilities would be transparently provided and no additional hardware or software is needed. IPv6 has integrated the features of IPSec, making it the best option. With IPv4, however, integration requires making changes to the IP implementation on each device.

Other Protocols

There are other secure communication and transport protocols. These include Hypertext Transport Protocol Secure (**HTTPS**), Secure Shell (**SSH**), Secure/Multipurpose Internet Mail Extensions (**S/MIME**), and Secure Real-time Transport Protocol (**SRTP**).

Hypertext Transport Protocol Secure (HTTPS)

One common use of TLS is to secure Hypertext Transport Protocol (HTTP) communications between a browser and a web server. This secure version is “plain” HTTP sent over TLS or SSL and is called **Hypertext Transport Protocol Secure (HTTPS)**. HTTPS uses port 443 instead of HTTP’s port 80 and is indicated by *https://* instead of *http://* in the URL.

At one time, web browsers prominently displayed a colored visual indicator to alert users that the connection between the browser and the web server was using HTTPS. Most browsers displayed a green padlock to indicate the connection was encrypted and secure. However, as more websites transitioned to HTTPS, web

browsers changed from displaying a color indicator that the connection was secure to only a warning that the connection was *not* secure. **Today most browsers display a padlock, but it is gray in color.**

Secure Shell (SSH)

Secure Shell (SSH) is an encrypted alternative to the Telnet protocol used to access remote computers. SSH is a Linux/ UNIX-based command interface and protocol for securely accessing a remote computer. Both the client and server ends of the connection are authenticated using a digital certificate, and passwords are protected by being encrypted. SSH can even be used as a tool for secure network backups.

Secure/Multipurpose Internet Mail Extensions (S/MIME)

Secure/Multipurpose Internet Mail Extensions (S/MIME) is a protocol for securing email messages. MIME is a standard for how an electronic message will be organized, so S/MIME describes how encryption information and a digital certificate can be included as part of the message body. It allows users to send encrypted messages that are also digitally signed.

Secure Real-time Transport Protocol (SRTP)

The Secure Real-time Transport Protocol (SRTP) has several similarities to S/MIME. Just as S/MIME is intended to protect MIME communications, SRTP is a secure extension protecting transmissions using the Real-Time Transport Protocol (RTP). Also, as S/MIME is designed to protect only email communications, SRTP provides protection for Voice over IP (VoIP) communications. SRTP adds security features, such as message authentication and confidentiality, for VoIP communications.

Implementing Cryptography

Cryptography that is improperly applied can lead to vulnerabilities that threat actors will exploit. There are different options and configurations that relate to cryptography that must be implemented correctly. Implementing cryptography includes understanding **key strength**, **secret algorithms**, and **block cipher modes of operation**.

Key Strength

A cryptographic **key** is a value that serves as **input** to an **algorithm**, which then transforms plaintext into ciphertext (and vice versa for decryption). A **key**, which is essentially a **random string of bits**, serves as an input parameter for **hash**, **symmetric encryption**, and **asymmetric cryptographic algorithms**.

Caution - A key is different from a password. Passwords are designed to be created and **remembered** by humans so that the passwords can be reproduced when necessary. A key is used by hardware or software that is running the cryptographic algorithm; as such, human readability is not required.

The following three primary characteristics determine the resiliency of the key to attacks (called key strength):

- **Randomness.** For a key to be considered strong, it must be random with no predictable pattern. This thwarts an attacker from attempting to uncover the key.
- **Cryptoperiod.** Another characteristic that determines key strength is its **cryptoperiod**, or the

length of time for which a key is authorized for use. Having a limited cryptoperiod helps protect the ciphertext from extended cryptanalysis and limits the exposure time if a key is compromised. Different cryptoperiods are recommended for different types of keys.

- **Key length.** The final characteristic is the length of the key. Shorter keys can be more easily broken than longer keys. All the possible values for a specific key make up its **key space**. The formula for determining a given key space for symmetric algorithms is $\text{character-set}^{\text{key-length}}$. For example, suppose a key has a length of 3 and is using a 26-character alphabet. The list of possible keys (*aaa, aab, aac*, etc.) would be 26^3 or **17,576** possible outcomes. Thus, the **key length** in this example is 3 and the **key space** is 17,576.

Note - On average, half the key space must be searched to uncover the key. A key with a length of only 3 that has a key space of 17,576 requires only 8,788 keys to be searched (on average) until the correct key is discovered. However, if the key length was increased by just one character to 4, the key space increases to 456,976 requiring on average 228,488 attempts. Just increasing the key length can have a significant impact on security.

Secret Algorithms

Although keys need to be kept secret (except for public keys), does the same apply to algorithms? That is, should an enterprise invest in hiring a cryptographer to create a new cryptographic algorithm and then hide the existence of that algorithm from everyone? Wouldn't such a secret algorithm enhance security in the same way as keeping a key or password secret?

The answer is no. In the past, cryptographers have often attempted to keep their algorithms or the workings of devices that encrypted and decrypted documents a secret. However, this approach has always failed. One reason is because for cryptography to be useful, it needs to be widespread: a military force that uses cryptography must by nature allow many users to know of its existence to use it. The more users who know about it, the more difficult it is to keep it a secret. In contrast, a password only requires one person - the user - to keep it confidential.

Block Cipher Modes of Operation

One variation in cryptographic algorithms is the amount of data that is processed at a time. Some algorithms use a stream cipher, while other algorithms make use of a block cipher. Whereas a **stream cipher** works on **one character at a time**, a **block cipher** manipulates an **entire block** of plaintext at one time. Because the size of the plaintext is usually larger than the block size itself, the plaintext is divided into separate blocks of specific lengths, and then each block is encrypted independently.

A **block cipher mode of operation** specifies how block ciphers should handle these blocks. It uses a symmetric key block cipher algorithm to provide an information service. This service could be **authentication mode of operation** that provides a credentialing service or **unauthentication mode of operation** that provides a service such as confidentiality. Some of the most common modes are:

- **Electronic Code Book (ECB).** The ECB mode is the most basic approach: the plaintext is divided into blocks, and each block is then encrypted separately. However, this can result in two identical plaintext blocks being encrypted into two identical ciphertext blocks. Attackers can use this repetition to their advantage. They could modify the encrypted message by modifying a block or even reshuffle the order of the blocks of ciphertext. ECB is not considered suitable for use.

- **Cipher Block Chaining (CBC).** CBC is a common cipher mode. After being encrypted, each ciphertext block gets “fed back” into the encryption process to encrypt the next plaintext block. Using CBC, each block of plaintext is XORed with the previous block of ciphertext before being encrypted. Unlike ECB in which the ciphertext depends only on the plaintext and the key, CBC is also dependent on the previous ciphertext block, making it much more difficult to break.
- **Counter (CTR).** CTR mode requires that both the message sender and receiver access a counter, which computes a new value each time a ciphertext block is exchanged. The weakness of CTR is that it requires a synchronous counter for both the sender and receiver.
- **Galois/Counter (GCM).** GCM mode both encrypts plaintext and computes a message authentication code (MAC) to ensure that the message was created by the sender and that it was not tampered with during transmission. Like CTR, GCM uses a counter. It adds a plaintext string called additional authentication data (AAD) to the transmission. The AAD may contain the addresses and parameters of a network protocol that is being used.

Summary

- A digital certificate is the user’s public key that has been digitally signed by a trusted third party who verifies the owner and that the public key belongs to that owner. It also binds the public key to the certificate. A user who wants a digital certificate must generate the public and private keys to be used and then complete a process known as a certificate signing request (CSR) generation. The user electronically signs the CSR by affixing their public key and then sending it to a registration authority, who verifies the authenticity of the user. The CSR is then sent to an intermediate certificate authority (CA), who processes the CSR. The intermediate CAs perform functions on behalf of a certificate authority (CA) that is responsible for digital certificates.
- A Certificate Repository (CR) is a list of approved digital certificates. Revoked digital certificates are listed in a Certificate Revocation List (CRL), which can be accessed to check the certificate status of other users. The status can also be checked through the Online Certificate Status Protocol (OCSP). When using OCSP stapling, web servers send queries to the Responder OCSP server at regular intervals to receive a signed, time-stamped OCSP response. Because digital certificates are used extensively on the Internet, modern web browsers are configured with a default list of CAs and the ability to automatically update certificate information.
- The process of verifying that a digital certificate is genuine depends on certificate chaining, or linking several certificates together to establish trust between all the certificates involved. The beginning point of the chain is a specific type of digital certificate known as a root digital certificate, which is created and verified by a CA and also self-signed. Between the root digital certificate and the user certificate can be one or more intermediate certificates that have been issued by intermediate CAs. Approved root digital certificates and intermediate certificates are distributed in one of three ways: through updates to the OS, through updates to the web browser, or by pinning (hard-coding within the program). The endpoint of the chain is the user digital certificate itself.
- Domain validation digital certificates verify the identity of the entity that has control over the domain name but indicate nothing regarding the trustworthiness of the individuals behind the site. Extended Validation (EV) certificates require more extensive verification of the legitimacy of the business. A wildcard digital certificate is

used to validate a main domain along with all subdomains. The limitation of wildcard digital certificates is addressed by the Subject Alternative Name (SAN), which allows different values to be associated with a single certificate and also permits a certificate to cover multiple IP addresses.

- A machine/computer digital certificate is used to verify the identity of a device in a network transaction. Digital certificates are used by software developers to digitally sign a program to prove that the software comes from the entity that signed it and no unauthorized third party has altered or compromised it are called code signing digital certificates. The most widely accepted format for digital certificates is the X.509 standard. There are several different certificate attributes that make up an X.509 digital certificate.
- Public key infrastructure (PKI) is the underlying foundation that serves as a key management system for controlling public keys, private keys, and digital certificates. PKI is the set of software, hardware, processes, procedures, and policies that are needed to create, manage, distribute, use, store, and revoke digital certificates across large user populations. The goal of a PKI is to establish the identity of people, devices, and services to control access to resources, protect data, and provide accountability. PKI can be considered as digital certificate management at scale.
- One of the principal foundations of PKI is that of trust. Three basic PKI trust models use a CA. The hierarchical trust model assigns a single hierarchy with one master CA called the root, who signs all digital certificate authorities with a single key. The bridge trust model is similar to the distributed trust model. No single CA signs digital certificates, and yet the CA acts as a facilitator to interconnect all other CAs. The distributed trust model has multiple CAs that sign digital certificates.
- An organization that uses multiple digital certificates on a regular basis needs to properly manage those digital certificates. Such management includes establishing policies and practices and determining the life cycle of a digital certificate. Because keys form the very foundation of PKI systems, they must be carefully stored and handled. Handling keys includes key escrow, setting expirations, generating renewals, revoking keys, recovering keys, suspending keys, and finally key destruction.
- Cryptographic algorithms are used to protect data in transit, called transport/communication encryption. There are different secure communication and transport protocols based on cryptographic algorithms for protecting data in transit. These protocols typically rely on encapsulating the data to be transmitted through tunneling. Secure Sockets Layer (SSL) was an early cryptographic transport protocol but is replaced with the more secure Transport Layer Security (TLS). Internet Protocol Security (IPSec) is a group or suite of protocols for securing IP communications. IPSec encrypts and authenticates each IP packet of a session between devices. IPSec is considered a more robust protocol than TLS. This is because it provides security to IP, which is the basis for all other TCP/IP protocols. There are different methods for deploying IPSec.
- There are other secure communication and transport protocols. Hypertext Transport Protocol Secure (HTTPS), a secure version for web communications, is HTTP sent over TLS. Secure Shell (SSH) is a Linux/UNIX-based command interface and protocol for securely accessing a remote computer communicating over the Internet. Secure/Multipurpose Internet Mail Extensions (S/MIME) is a protocol for securing email messages. The Secure Real-time Transport Protocol (SRTP) provides protection for Voice over IP (VoIP) communications.
- Cryptography that is improperly applied can lead to vulnerabilities that will be exploited; thus, it is necessary to understand the different options that relate to cryptography so that it can be implemented correctly. A key must be strong to resist attacks. A strong key must be random with no predictable pattern. Keys should also be long and the length of time for which a key is authorized for use should be limited. Any attempt to keep an algorithm secret will not result in strong security. A block cipher mode of operation specifies how block ciphers should handle blocks of plaintext.

Projects

Project 4-1: SSL Server and Browser Tests

Objective: Test servers and web browsers for security.

Description: In this project, you will use online tests to determine the security of web servers and your local web browser.

1. Go to **www.ssllabs.com**. (It is not unusual for websites to change the location of where files are stored. If this URL no longer functions, open a search engine and search for “Qualys SSL Server Test.”)
2. Click **Test your server**.
3. Click the first website listed under **Recent Best**.
4. Note the grade given for this site. Under **Summary**, note the **Overall Rating** along with the scores for **Certificate**, **Protocol Support**, **Key Exchange**, and **Cipher Strength**, which make up the cipher suite.
5. If this site did not receive an Overall Rating of A or better under **Summary**, you will see the reasons listed. Read through these. Would you agree? Why?
6. Scroll down through the document and read through the **Certificate #1** information. Note the information supplied regarding the digital certificates. Under **Certification Paths**, click **Click here to expand** if necessary to view the certificate chaining. What can you tell about it?
7. Scroll down to **Configuration**. Note the list of protocols supported and not supported. If this site were to increase its security, which protocols should it no longer support? Why?
8. Under **Cipher Suites**, interpret the suites listed. Notice that they are given in server-preferred order. To increase its security, which cipher suite should be listed first? Why?
9. Under **Handshake Simulation**, select the web browser and operating system that you are using or is similar to what you are using. Read through the capabilities of this client interacting with this web server. Note particularly the order of preference of the cipher suites. Click the browser’s Back button when finished
10. Scroll to the top of the page. This time select one of the **Recent Worst** sites. As with the previous excellent example, now review the **Summary**, **Authentication**, **Configuration**, **Cipher Suites**, and **Handshake Simulation**. Would you agree with this site’s score?
11. If necessary, return to the **SSL Report** page and enter the name of your school or work URL and generate a report. What score did it receive?
12. Review the **Summary**, **Authentication**, **Configuration**, **Cipher Suites**, and **Handshake Simulation**. Would you agree with this site’s score?
13. Make a list of the top five vulnerabilities that you believe should be addressed in order of priority. If possible, share this with any IT personnel who may be able to take action.
14. Click **Home** on the navigation menu.
15. Now test the capabilities of your web browser. Click **Test your browser**. Review the capabilities of your web browser. Print or take a screen capture of this page.
16. Close this web browser.
17. Now open a different web browser on this computer or on another computer.
18. Return to the **www.ssllabs.com** home page and then click **Test your browser** to compare the two scores. From a security perspective, which browser is better? Why?
19. Close all windows.

Project 4-2: Viewing Digital Certificates

Objective: View details of digital certificates.

Description: In this project, you will view digital certificate information using the Google Chrome web browser.

1. Use the Google Chrome web browser to go to **www.google.com**.
2. Note the padlock in the address bar. Although you did not enter **https://**, nevertheless Google created a secure HTTPS connection. Why would it do that?
3. Click the three vertical buttons at the far edge of the address bar.
4. Click **More tools**.
5. Click **Developer tools**.
6. Click the **Security** tab, if necessary. (If the Security tab does not appear, click the **More tabs (>>)** button to display more tabs.)
7. Read the information under **Security overview**.
8. Click **View certificate**.
9. Note the general information displayed on the **General** tab.
10. Now click the **Details** tab.
11. Note the root of trust under **Certificate Hierarchy**.
12. Under **Certificate Fields**, click **Serial Number** to view the unique number associated with this digital certificate in the **Field Value** box.
13. If necessary, scroll down to **Validity** and then click **Not After**. What is the expiration date of this certificate?
14. Locate **Subject Public Key Info** and click **Subject Public Key Algorithm**. What type of encryption is used?
15. Locate **Certificate Signature Algorithm** and click it. What hash and encryption are used for this certificate?
16. Click **Subject's Public Key** to view the public key associated with this digital certificate. Why is this site not concerned with distributing this key? How does embedding the public key in a digital certificate protect it from impersonators?
17. Close all windows.

Project 4-3: Viewing Digital Certificate Revocation Lists (CRLs) and Untrusted Certificates

Objective: Examine certificates.

Description: Revoked digital certificates are listed in a certificate revocation list (CRL), which can be accessed to check the certificate status of other users. In this project, you will view the CRL and any untrusted certificates on your Microsoft Windows computer.

1. Press the **Windows+R** keys.
2. Type **mmc.exe** and then press **Enter** to launch the Microsoft Management Console.
3. Click **File**.
4. Click **Add/Remove Snap-in**.
5. Click **Certificates** and then click **Add**.
6. Select **Computer account**, click **Next**, and then click **Finish**.
7. Click **OK**.
8. In the left pane, expand **Certificates – Current User**.

9. Expand **Trusted Root Certification Authorities**. These are the CAs approved for this computer. Scroll through this list. How many of these have you heard of before?
10. In the left pane, expand **Intermediate Certification Authorities**.
11. Click **Certificates** to view the intermediate CAs. Scroll through this list.
12. In the left pane, click **Certificate Revocation List**.
13. All revoked certificates will be displayed. Select a revoked certificate and double-click it.
14. Read the information about it and click fields for more detail, if necessary. Why do you think this certificate has been revoked? Close the Certificate Revocation List by clicking the **OK** button.
15. In the left pane, expand **Untrusted Certificates**.
16. Click **Certificate Trust List**. The certificates that are no longer trusted are listed.
17. Double-click one of the untrusted certificates. Read the information about it and click fields for more detail, if necessary. Why do you think this certificate is no longer trusted?
18. Click **OK** to close the Certificate dialog box.
19. Close all windows.

Project 4-4: Downloading and Installing Trusted Root Certificates

Objective: Install certificates.

Description: By default Windows 11 updates its root certificate over the Internet through Windows Update at least once per week through a Trusted Root Certificate List (CTL). However, if a device is not connected to the Internet, then certificates will expire over time. This could prevent scripts and applications from functioning properly. The latest root certificates can be downloaded directly from Microsoft and installed. These are stored in a Serialized Certificate Store (SST) format. In this project, you will use Windows PowerShell to download and install root certificates.

1. Press the **Windows+R** keys.
2. Type **powershell** and then press **CTRL+SHIFT+ENTER** to launch PowerShell in administrator mode.
3. PowerShell can be used to view the details on all root certificates. Type **Get-Childitem cert:\LocalMachine\root |format-list** and press **Enter**. Scroll up to view the details on the root certificates.
4. To view the expired certificates, type **Get-Childitem cert:\LocalMachine\root | Where {\$_.NotAfter -lt (Get-Date).AddDays(40)}** and press **Enter**.
5. In PowerShell, navigate to a drive or location from which you can easily retrieve a file; for example, enter **F:** and then press **Enter**.
6. Download the latest certificates in an SST file by typing **certutil.exe -generateSSTFromWU roots.sst** and press **Enter**.
7. Type **dir** to confirm that the **roots.sst** file was downloaded.
8. Run the following command while replacing **CertPath** with the complete path to the downloaded SST file, for example, **\$sstStore 5 (Get-Childitem -Path F:\roots.sst)**, and press **Enter**.
9. Now import all the certificates by typing **\$sstStore | Import-Certificate -CertStoreLocation Cert:\LocalMachine\Root** and press **Enter**.
10. All the certificates have now been imported. Type **Get-Childitem cert:\LocalMachine\root |format-list** and press **Enter**.
11. Close all windows.

Project 4-5: Downloading and Installing a Digital Certificate

Objective: Install a certificate.

Description: In this project, you will download and install a digital certificate within the Adobe Acrobat Reader DC.

1. Check to determine if you already have Adobe Acrobat Reader DC or Adobe Acrobat Professional installed on your computer. If so, you may skip these download and installation steps and go directly to Step 5.
2. Go to **get.adobe.com/reader/**. (It is not unusual for websites to change the location of where files are stored. If this URL no longer functions, open a search engine and search for “Adobe Acrobat Reader DC Download.”)
3. Click **Download Acrobat Reader**.
4. Follow the instructions to install Acrobat Reader.
5. Launch Acrobat Reader.
6. Click **Edit**.
7. Click **Preferences**.
8. Click **Signatures**.
9. Under **Identities & Trusted Certificates**, click **More**.
10. In the left pane, if necessary, click **Digital IDs** to display the menu choices.
11. On the menu at the top of the main pane, click the **Add ID** icon (it is the first icon and has a plus sign).
12. Click **A new digital ID I want to create now**. Click **Next**.
13. If necessary, click **New PKCS#12 digital ID file**. Click **Next**.
14. Enter the requested information. Under **Key Algorithm**, click the down arrow to see the two options. The default is **2048-bit RSA**, which provides more security, while 1024-bit RSA provides less security but is more universally compatible. Accept the 2048-bit RSA.
15. Under **Use digital ID for**, click the down arrow to see the three options. Select the default **Digital Signatures and Data Encryption**. Click **Next**.
16. Create and enter a strong password and then confirm that password. Click **Finish**.
17. Your file is now created. Click **Export**.
18. If necessary, click **Save the data to a file** and click **Next**.
19. Save the file to your computer.
20. Close the windows associated with configuring your certificate.
21. You can use this certificate by sending it to anyone who needs to validate your identity. Close all Windows.