Chapter 3

Encryption and Authentication

In This Chapter

- Secret key encryption
- Hashes and one-way functions
- Public key encryption
- Password authentication
- Challenge/response authentication
- Sessions
- Public key authentication
- Digital signatures
- Certificates
- Biometric authentication

Nearly all modern security mechanisms are based on keeping secrets private to certain individuals. Security systems use encryption to keep secrets, and they use authentication to prove the identity of individuals. These two basic security mechanisms are the foundation upon which nearly all security mechanisms are based.

Encryption

encryption
The process of encoding a plain-text message so that it cannot be understood by intermediate parties who do not know the key to decrypt it.

The primary purpose of *encryption* is to keep secrets. It has other uses, but encryption was first used to protect messages so that only the person that knew the trick to decoding a message could read it. Today, encryption allows computers to keep secrets by transforming data to an unintelligible form using a mathematical function.

Just like simple arithmetic, encryption functions combine the message and the encryption key to produce an encrypted result. Without knowing the *secret key,* the result makes no sense.

secret key
A key that must be kept secret by all parties because it can be used to both encrypt and decrypt messages.

For example, let's say I need to hide the combination to a lock. In this case, the combination (also called the *message*) is 9-19-69. To keep things simple, I'm going to add (adding is the *algorithm*) 25 (which is the key) to each of the numbers to produce the encrypted value: 34-44-94. I can post this value right on the combination lock so I won't forget it because that number won't do anyone who doesn't know how to use it any good. I just need to remember the algorithm, subtract, and the key, 25. The encrypted text is worthless without the key. I can also simply tell my friends what the key and the algorithm are, and they can combine that knowledge with the encrypted data to decode the original combination.

algorithm
A method expressed in a mathematical form (such as computer code) for performing a specific function or operation.

You may have noticed that in this example I used the opposite mathematical operation to decode the encrypted text; I added 25 to encode and subtracted 25 to decode. Simple arithmetic algorithms are called *symmetrical algorithms* because the algorithm used to encode can be reversed in order to decode the data. Since most mathematical operations can be easily reversed, symmetrical algorithms are common.

symmetrical algorithm
An algorithm that uses the same secret key for encryption and for decryption.

Although this example may seem simplistic, it is exactly what happens with modern secret-key *cryptosystems.* The only differences are in the complexity of the algorithm and the length of the key. This example, despite its simplicity, shows exactly how all symmetric encryption systems work. Here is another example, using a slightly more complex key. Notice how the key is repeated as many times as necessary to encode the entire message.

ENCRYPT

	D	E	A	R		D	I	A	R	Y	,		I	T	'	S		B	E	E
+	S	E	C	R	E	T	C	O	D	E	S	E	C	R	E	T	C	O	D	E
	W	J	D	J	E	X	M	P	V	D	S	E	L	W	E	M	C	Q	I	J

DECRYPT

	W	J	D	J	E	X	M	P	V	D	S	E	L	W	E	M	C	Q	I	J
–	S	E	C	R	E	T	C	O	D	E	S	E	C	R	E	T	C	O	D	E
	D	E	A	R		D	I	A	R	Y	,		I	T	'	S		B	E	E

The most common use for encryption with computers is to protect communications between users and communications devices. This use of encryption is an extension of the role codes and *ciphers* have played throughout history. The only difference is that, instead of a human being laboriously converting messages to and from an encoded form, the computer does all the hard work.

cryptosystem
A computing system that implements one or more specific encryption algorithms.

Encryption isn't just for communication. It can also be used to protect data in storage, such as data on a hard drive. Most modern operating systems like Unix or Windows are configured to allow only authorized users to access files while the operating system is running, but when you turn your computer off, all those security features go away and your data is left defenseless. An intruder could load another operating system on the computer or even remove the hard drive and place it in another computer that does not respect the security settings of the original computer, and your data would be accessible. Encryption solves this problem by ensuring that the data is unintelligible if the correct key isn't provided, irrespective of whether the computer is still running in order to protect the data.

cipher
An algorithm specifically used for encryption.

Secret Key Encryption

Our example in the last section was an example of *secret key encryption*. In secret key encryption, the same key is used to both encode and decode the message, so it is said to be symmetrical—because both keys are the same. Secret key encryption requires that both parties know the algorithm and the key in order to decode the message. Until the development of *public key encryption* by cryptographers in the 1970s, secret key encryption was the only type of encryption available.

secret key encryption
Encryption by means of a secret key.

Secret key encryption works well for keeping secrets, but both parties have to know the same secret key in order to decode the message. There's no secure way to transfer the key from one party to the other without going to extraordinary lengths, like having both parties meet in the same secluded area to exchange keys. There's certainly no way to exchange keys over an electronic medium without the possibility of a wiretap intercepting the key.

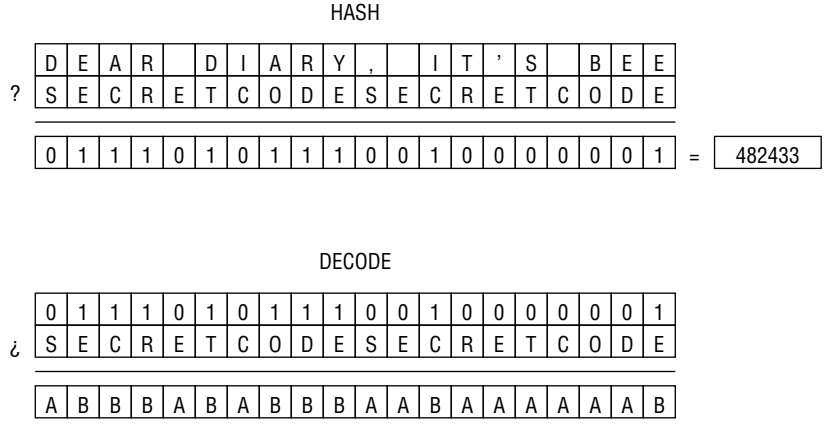
public key encryption
Encryption by means of a public key; an encryption methodology that allows the distribution of an encryption key that does not compromise the secrecy of the decrypting private key due to the utilization of a related pair of one-way functions.

One-Way Functions (Hashes)

Hashes are used to verify the correctness of information and are based on mathematical algorithms called one-way functions. Some mathematical functions cannot be reversed to retrieve the original number. For example, let's say that we're going to divide 46,835,345 by 26,585. This results in 1,761 with a remainder of 19,160. So let's say that we have an algorithm that simply returns the remainder (19,160) and discards the quotient (1,761). Now, if we have just the remainder (called a modulus) and one of the original numbers, there's no way to reconstruct the other operand because the quotient has been discarded. The remainder alone does not retain enough information to reconstruct the original number.

plaintext
An unencrypted text, either before it has been encrypted or after it has been decrypted. The encrypted version is referred to as a ciphertext.

The following illustration shows an extremely simple hash. In this algorithm, two numbers are added, and if the result is even, a binary 1 is the result. If the result is odd, a 0 is the result. The combination of binary digits forms the hash. Because the actual numbers are lost, the hash cannot be reconstructed, but it can be used to determine with reasonable certainty whether or not two plaintexts match. Simple hashes are often called *checksums*. This hashing algorithm is not appropriate for security because there are many other plaintexts that will produce the same result, so the probability of an accidental match is too risky.



one-way function
An algorithm that has no reciprocal function and cannot therefore be reversed in order to discover the data originally encoded.

Of course, since we can't reconstruct the original number, we can't use *one-way functions* to encode and decode information. But we can use them to be certain that two people know the same number without revealing what that number is. If two people were aware of the original dividend (46,835,345) in the previous scenario and you told them to divide the number they knew by 26,585, discard the whole product, and tell you the remainder, they would both report 19,160—thus proving that they knew the same original number or that they were amazingly lucky, because there is only a 1 in 26,585 chance that they could have guessed the correct remainder. By simply making the number large enough that the odds of guessing the correct remainder are impossibly high, you can use this one-way nonreversible function to prove that two parties know a number without ever revealing what that number actually is to anyone who might overhear you.

For example, let's say a system requires logon authentication in the form of a user providing their name and password. They could simply enter their name and password and the computer could check that against a list of stored passwords. If the passwords matched, the user would be allowed in. But let's say that hackers gained access to the computer. They could steal the list and then have everyone's password. Once other people know a user's password, you can no longer hold

that user accountable for their actions because you can't guarantee that they actually performed those actions on a system.

We can't use secret key encryption to encrypt the password file because the secret key would need to be stored on the system in order for the system to decode the password file. Because we've already stipulated that hackers have access to the system, they would also have access to the secret key and could use that to decrypt the password file.

This is a situation in which one-way functions work well. Rather than storing the user's *password,* we can store a *hash,* or the result of a one-way function, instead. Then, when they enter their password, we perform the one-way function on the data they entered and compare it to the hash that's stored on disk. Because only the hash is stored and the hashing algorithm can't be reversed to find the original password, hackers can't compromise the authentication system by stealing this list of password hashes.

password
A secret key that is remembered by humans.

Hashes allow a user to prove that they know the password without the system having to know what the password actually is. Protecting passwords is the most common use for using hashing algorithms in computer security.

hash
The result of applying a one-way function to a value.

Public Key Encryption

Whereas symmetric ciphers like secret key algorithms use the same key to encrypt and decrypt messages, public key encryption uses one key to encrypt a message and a different key to decrypt it, so they are called *asymmetric algorithms.* In a public key encryption system, the encryption key is called the public key because it can be made public. The decryption key is called the *private key* because it must be kept private in order to remain secure.

asymmetrical algorithm
A mathematical function that has no reciprocal function.

The problem with secret key encryption is this: Both the sender and the recipient must have the same key in order to exchange encrypted messages over an nonsecure medium. If two parties decide to exchange private messages, or if two computers' network devices or programs must establish a secure channel, the two parties must decide on a common key. Either party may simply decide on a key, but that party will have no way to send it to the other without the risk of it being intercepted on its way. It's a chicken-and-egg problem: Without a secure channel, there is no way to establish a secure channel.

private key
A secretly held key for an asymmetrical encryption algorithm that can be used only to decode messages or encode digital signatures.

With public key encryption, the receiver can send a public encryption key to the sender. There is no need to keep the *public key* a secret because it can be used only to encode messages, not decode them. You can publish your public key to the world for anyone to use for encoding message they send to you.

public key
A publicly distributed key for an asymmetrical encryption algorithm; a public key can be used only to encode messages or decode digital signatures.

When the receiver gets a message that has been encoded with their public key, they can use their private key to decode the message. Revealing their public key to the world for encoding does not allow anyone else to decode their private messages.

cryptography
The study of codes, ciphers, and encryption.

Public key cryptography is a relatively new development in *cryptography*, one that solves many long-standing problems with cryptographic systems—especially the chicken-and-egg conundrum of how to exchange secret keys. In 1976, Witfield Diffie and Martin Hellman figured out a way out of the secure channel dilemma. They found that some one-way functions could be undone by using a different key for decryption than was used for encryption. Their solution (called public key cryptography) takes advantage of a characteristic of prime and almost prime numbers, specifically, how hard it is to find the two factors of a large number that has only two factors, both of which are prime. Since Diffie and Hellman developed their system, some other public key ciphers have been introduced using even more difficult one-way functions.

One problem that plagues secure public key ciphers is that they are slow—much slower than symmetric ciphers. You can expect a good public key cipher to take 1,000 times as long to encrypt the same amount of data as a good symmetric cipher would take. This can be quite a drag on your computer's performance if you have a lot of data to transmit or receive.

Hybrid Cryptosystems

Although it is much slower than symmetric systems, the public key/private key system neatly solves the problem that bedevils symmetric cryptosystems—exchanging secret keys.

hybrid cryptosystem
A cryptosystem that exchanges secret keys using public key encryption to secure the key exchange and then using the higher speed allowed by secret key encryption to transmit subsequent data.

But there's no need to give up the speed of secret key cryptosystems just because secret keys can't be exchanged securely. *Hybrid cryptosystems* use public key encryption to exchange secret keys and then use the secret keys to establish a communication channel. Nearly all modern cryptosystems work this way.

When two people (or devices) need to establish a secure channel for communication, one of them can generate a random secret key and then encrypt that secret key using the receiver's public key. The encrypted key is then sent to the receiver. Even if the key is intercepted, only the intended recipient, by using their private key, can decrypt the message containing the secret key.

Once both parties have the secret key, they can begin using a much faster secret key cryptosystem to exchange secret messages.

Authentication

authentication
The process of determining a user's identity in order to allow access.

Authentication is used to verify the identity of users to control access to resources, to prevent unauthorized users from gaining access to the system, and to record the activities of users in order to hold them accountable for their activities.

Authentication is used to verify the identity of users logging on to computers, it's used to ensure that software you download from the Internet comes from a reputable source, and it's used to ensure that the person who sends a message is really who they say they are.

Password Authentication

Passwords are the oldest and simplest form of authentication—they've been used since time immemorial to prove that an individual should be given access to some resource. Technically, passwords are secret keys.

Password authentication is simple: By knowing a secret key, you could prove that the individual who invented the secret trusted you with that secret key and that the trust should be bestowed upon you. This sort of password authentication proves only that access to a resource should be allowed—it does not prove identity.

To prove identity, a password must be unique to a specific person. In secure computer systems, this is accomplished by creating user accounts, which are assigned to individuals. The account contains information about who the owner is and includes a unique account name and password.

When a user logs on to the system, they simply type in their account name to assert their identity, and then they provide the password associated with that user account to prove that they are allowed to use that account. If the entered password matches the stored password, the user is allowed access to the system.

Password authentication can fail in a number of ways:

- There's no way to control password distribution. If the account holder loses control of the password, it can be distributed to anyone. Once a password has been compromised, password authentication can no longer be used to prove identity.
- Passwords are often simple and easy to guess, and many systems limit passwords to lengths that lend themselves to *brute-force* guessing—that is, simply trying all possible combinations. This can lead to password compromise and is one of the most common ways that hackers gain access to systems.
- Naïve implementations may not protect the password in transit or may be compromised through simple techniques like *replay*.

brute-force attack
An attack in which every possible combination of values is tried in sequence against a password system. Given an unlimited amount of time, these attacks will always succeed, but they are impractical against long passwords, which could require more time than the age of the universe to crack.

replay attack
An attack in which a secret value like a hash is captured and then reused at a later time to gain access to a system without ever decrypting or decoding the hash. Replay attacks only work against systems that don't uniquely encrypt hashes for each session.

Despite the common failure of password-based systems to actually prove identity and restrict access, they are by a wide margin the most common way to secure computers.

Password Hashing

To prevent hackers from capturing your password from your computer's hard disk or while it transits the network, passwords can be encrypted using a one-way function or hashing algorithm to keep them from being revealed.

In most modern operating systems, the operating system does not compare your password to a stored password. Instead, it encrypts your password using a one-way cryptographic function and then compares the result to the original result that was stored when you created your password. Because the hashing function is one way, it cannot be reversed to decrypt your password.

However, password hashing is susceptible to brute-force indexed decryption. Using this technique, hackers create a "dictionary" of all possible passwords by encrypting every possible password (i.e., AAAAAA through ZZZZZZ in a system limited to six letters) using the same hashing function as the password system and storing the results in a database along with the original text. Then, by capturing your hash, they can look up the matching value in their database of hashed values and find the original password. Although compiling this sort of dictionary can take a long time for a single computer, it is the sort of problem that could be easily distributed over a network of computers (like the Internet) to speed up completion. Once finished, the hashing algorithm would be all but worthless for encrypting passwords and a new algorithm would be required to maintain password security.

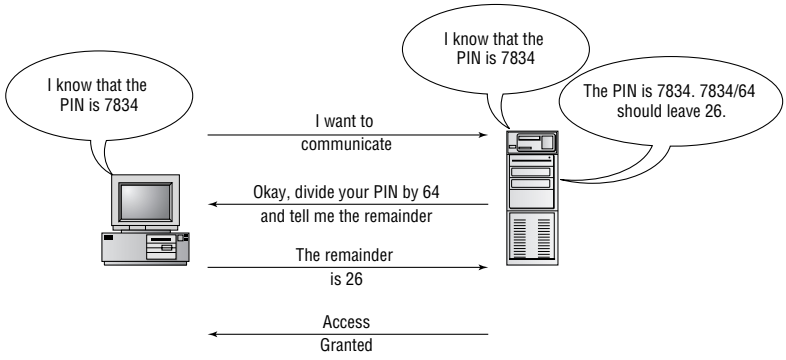
Challenge/Response Authentication

One-way hashes are great for preventing password lists from being exploited, but they can't be used securely over a network. The problem is that a hacker might be wiretapping or "sniffing" the network connection between the two computers and could intercept the hashed password and "replay" it later to gain access. This is actually a very common problem that can occur whether or not the password has been hashed. A large number of older TCP/IP protocols like FTP and Telnet are susceptible to sniffing and replay attacks.

challenge/response
A method used to prove that a party knows a password without transmitting the password in any recoverable form over a network.

Challenge/response authentication defeats replay by encrypting the hashed password using secret key encryption. A challenge and response authentication session works like this:

1. The client requests a connection.
2. The server sends a random secret key to the client.
3. The client encrypts the random secret key using a hashed password and transmits the result to the server.
4. The server decrypts the secret using the stored hashed password and compares it to the original secret key to decide whether to accept the logon.



This system works because the password is never transmitted over the network, even in hashed form; only a random number and an encrypted random number are sent, and no amount of cryptographic work can be used to decrypt the encrypted random number because all possible results are equally likely.

Challenge and response authentication can also be used to defeat a brute-force indexed hash decryption as well—as long as the hacker's computer isn't the one you're trying to gain access to. If it is, the hacker can decrypt your hash because they sent you the secret key that you encrypted. By decrypting the secret key, they have your hash and can then compare it to their database of encrypted hashes. For this reason, it's imperative that you never log on to a hacker's system using your system's encrypted hash.

WARNING

Internet Explorer will automatically hand over an encrypted hash of your Windows logon account name and password to any website that asks for it—behind the scenes and without asking for your permission. This means that by attracting you to a website, hackers could steal your account name and password for your local system. This is one of the many esoteric security flaws that Microsoft has committed in the name of ease of use.

Session Authentication

Sessions are used to ensure that, once authentication has occurred, further communications between the parties can be trusted—in other words, that others cannot hijack the connection by forging packets. The authentication that occurs at the beginning of the session can be reliably carried forward throughout the remainder of the packet stream until both parties agree to end the session.

session
An authenticated stream of related packets.

As you know, all modern networking systems are packet based. Packets transmitted from one computer to the next are reconstructed into a communication stream by being put back in their original order using a special number called a sequence number embedded in each packet.

Sequence numbers are exactly that—a number that indicates which packet in the stream a specific packet represents. Sequence numbers could be simple serial numbers, like 1, 2, 3, 4, and so on. Unfortunately, that would be too easy to predict, and a hacker could insert the fifth packet in a stream by simply inserting it into the communication with a sequence number of 5.

pseudorandom number
A member of a set of numbers that has all the same properties as a similarly sized set of truly random numbers, such as even distribution in a set, no predictable reoccurrences, and incompressibility, but that occur in a predictable order from a given starting point (seed).

To prevent sequence number prediction, sequence numbers are not sequentially generated; they're generated using a special function called a *pseudorandom number generator (PRNG)*, which can reliably create the same sequence of random numbers given a known *seed* number.

pseudorandom number generator (PRNG)
An algorithm that generates pseudo random numbers.

NOTE

Computers are very good at performing the same calculations reliably every time. But the property of reliability makes them terrible at generating random numbers. To generate truly random numbers, computers require some real-world external input. But computers can generate numbers that seem to be random—*pseudorandom numbers*.

seed
The starting point for a specific set of pseudorandom numbers for a specific PRNG.

Pseudorandom numbers have all of the same properties as random numbers, such as even distribution in a set, no predictable recurrences, and so forth. They're not, however, truly random because they use the same PRNG algorithm to generate numbers from the same starting number (or seed), hence the exact same series of pseudorandom numbers will be generated every time.

For example, consider the number pi. As we all know, it starts with 3.1415926 and goes on ad infinitum. There is no known sequence to the numbers in pi, and the numbers in the series seem to be completely random one to the next. So, if an algorithm can calculate a specific digit in pi—let's call the function pi, so that pi(473) would give you the 473rd number in pi—the number pi(*n*+1) is completely random compared to pi(*n*). We can generate seemingly random numbers by simply starting at pi(1) and sequencing through the algorithm. This is exactly how pseudorandom number generators work. In fact, a number of them are based on calculating pi or the square root of 2 or other such irrational numbers. In this example, *n*, or the location in the series, is the seed.

NOTE

PRNGs are terrible for games because games would play the exact same way each time given the same user input and the same starting number, but they're great for the specific purpose of generating session sequence numbers.

If a hacker does not know the seed number for a PRNG, the pseudorandom numbers in a sequence are, for all intents and purposes, completely random and unpredictable. But if a remote computer does know the seed, then the series of pseudorandom numbers is as predictable a sequence as 1, 2, 3. This is how sessions pass authentication from one packet to the next. When the sequence starts with a known encrypted seed number, every packet following in the session with the correct sequence number is known to have a chain of authentication back to the securely transmitted and authenticated seed number.

Public Key Authentication

key
A secret value used to encrypt information.

To use public *key* cryptography for authentication, the system administrator can install a user's public key on the system. For a user to gain access, the system sends a random number to the user, who then encrypts the number using their private key. This number is then sent to the remote system. If the system can decrypt the number using the stored public key and the result is the same random number that was originally provided, then the user has proven that they have the private key.

This proves that the administrator installed the corresponding public key and that the user should therefore be granted access to the system.

Public key authentication is often used when authentication should be performed automatically without user intervention. The systems involved can trade public keys and authentication information without the user interacting with the system. For this reason, public key–based authentication and its derivatives like *certificate*-based authentication are frequently used for machine authentication and for establishing anonymous encrypted sessions such as Secure Sockets Layer (SSL).

public key authentication
Authentication by means of a digital signature.

Certificate-Based Authentication

Certificates are simply *digital signatures* that have themselves been "signed" using the digital signature of some trusted authority, thus creating a "chain" of authentication.

certificate
A digital signature that has been digitally signed by one or more trusted authorities.

Digital signatures are used to prove that a specific piece of information came from a certain individual or other legal entity. Digital signatures do this by performing public key encryption in reverse—that is, the document is encrypted using the private key (which cannot be encrypted by any other key) and decrypted by anyone using the sender's public key.

Because everyone can have the sender's public key, the encrypted information cannot be considered secure or secret, but it can be trusted as authentic because only the person holding the original private key could possibly have created the digital signature.

digital signature
Any identity information encrypted with a private key and therefore decryptable—with a public key. Digital signatures are used to prove the validity of publicly available documents by proving that they were encrypted with a specific secretly held private key.

Typically, a digital signature contains identity information that is easily recognizable, such as the signer's name, physical address, and other easily verifiable information, along with a hash of the entire document that can be used to prove that the signed document has not been modified. If a document has been modified, the hash inside the encrypted signature (which cannot be modified) would not match a new hashing of the document during the authentication process.

The signature is usually appended to the end of the document and appears as a series of hexadecimal or ASCII text. When a document has been hashed and has a digital signature attached to it, it is said to have been "digitally signed."

Certificates are useful when you want to allow wide access to a system by means of distributed authenticators. For example, the government could set up a system whereby public notaries are given digital certificates created by the government. Then, when a member of the public at large wants a certificate that could be used to prove their identity, they could go to a notary public, prove their identity, and have the notary generate a certificate for them. Because that certificate contains a digital signature that can be proven to have come from a notary, which in turn contains a digital signature that can be proven to have come from a government body, anyone who trusts the government body to verify identity can trust that the person using the certificate is who they say they are.

Root Certifying Authority
An organization that exists simply to be trusted by participants in order to provide transitive trust. Root CAs certify the identities of all members so that members who trust the Root CA can trust anyone that they've certified. A Root CA is analogous to a notary public.

In a certificate hierarchy, the ultimate authority is called the *Root Certifying Authority (Root CA)*. Users of the system simply have to trust that the Root CA is a legitimate body. Currently, the vast majority of "rooted" certificates come from a single company, VeriSign, or its subsidiary, Thawte. Entrust is another large Root CA.

Certificates can be used for authentication in the same way that digital signatures can, but authentication systems can be configured to allow access based on higher-level certificates. For example, if your company received a single certificate from a Root CA, the system administrator could use it to generate unique certificates for each department. The department administrators could then generate unique certificates for each user. Then, when you needed to access a resource, the resource server could use the certificate to verify your identity (or the identity of the department that signed your certificate or the company that you work for). If the resource server only cared that you worked for the company, the company signature in your certificate would be the only data that it checked.

Biometric Authentication

biometric authentication
Authentication by means of invariant and unique biological characteristics such as fingerprints or DNA.

Biometric authentication uses physical sensors to detect patterns that uniquely identify a person, such as facial features, fingerprints, handprints, vocal characteristics, blood vessels in the eye, or DNA. These patterns are applied through a hashing algorithm to come up with a hash value that is invariant (in other words, a secret key) and can be matched to a stored value on a server.

Biometric scanning devices can range from simple to complex:

- Microphone (vocal characteristics)
- Optical scanner (fingerprint, handprint)
- Electrostatic grid (fingerprint, handprint)
- Digital video camera (facial features, retinal pattern)
- Deoxyribonucleic Acid Sequencer (DNA)

Currently, low-cost (<$150) fingerprint scanners are popular choices for biometric authentication, but I think they will probably be supplanted by voiceprint recognition because most computers already contain the hardware necessary to perform voiceprint authentication. Voiceprint sensors can also be sensitive enough to fail when stress is detected in the user's voice (such as when they're being coerced into providing access to a system).

Usually, biometric authentication devices use a challenge/response mechanism to ensure that the hashed value never leaves the sensor because it could be captured in transit and "replayed" to foil the authentication system. Because biometric "scans" are never exactly the same twice and must be hashed to generate the key, the system can store a copy of each authentication (or a separate type of hash that is unique) to record a "history" of logins, which can be compared to new attempts to ensure that they are unique and not replays (for example, a recording of a user's voice).

An example of a replay attack against a biometric algorithm would be the recording and playback of a person's *pass phrase*. Without replay detection, there would be no way for the sensing algorithm to determine that a recording (and not the authorized user's actual voice) was being used to gain access to the system. Biometric sensors usually must include additional hardware to ensure that they are not being faked by a replay attack. This usually includes sensors to verify that other requirements of the system are actually in place. For example, a fingerprint scanner doesn't allow access for a person, it allows access for a fingerprint. They can be fooled by something as simple as a color photograph of a valid fingerprint.

pass phrase
A very long password consisting of multiple words.

What the system designers really want to do is prove that the person with the fingerprint is the one accessing the system, so they must include "live finger detection" in addition to fingerprint detection. Therefore, the system could include other simple biometric sensors such as temperature, pulse, and even blood oxygen sensors that would be extraordinarily difficult to fake.

Terms to Know

algorithm
asymmetric algorithms
Authentication
biometric authentication
brute-force
certificate
challenge/response
ciphers
cryptography
cryptosystems
digital signatures
encryption
hash
hybrid cryptosystems
key
one-way functions
pass phrase
password
private key
pseudorandom number generator (PRNG)
pseudorandom numbers
public key
public key authentication
public key encryption
replay attack
Root Certifying Authority (Root CA)
secret key
secret key encryption
seed
sessions
symmetrical algorithms

Review Questions

1. What is the primary purpose of encryption?
2. Secret key encryption is said to be symmetrical. Why?
3. What is a hash?
4. What is the most common use for hashing algorithms?
5. What is the difference between public key encryption and secret key encryption?
6. What long-standing security problem does public key encryption solve?
7. What is the major problem with public key encryption when compared to secret key encryption?
8. What is a hybrid cryptosystem?
9. What is authentication used for?
10. What hacking attack is challenge/response authentication used to prevent?
11. How are sessions kept secure against hijacking?
12. What is the difference between a random number and a pseudorandom number?
13. What is a digital signature?
14. What is the difference between a certificate and a digital signature?
15. What sort of characteristics are typically used for biometric authentication?