**Caesar cipher**

Caesar developed a cryptographic system now known as the *Caesar cipher.* The system is extremely simple. To encrypt a message, you simply **shift each letter of the alphabet three places to the right**. For example, *A* would become *D* , and *B* would become *E* . If you reach the end of the alphabet during this process, you simply wrap around to the beginning so that *X* becomes *A,* *Y* becomes *B,* and Z becomes C. For this reason, the Caesar cipher also became known as the ROT3 (or Rotate 3) cipher. **The Caesar cipher is a substitution cipher that is monoalphabetic; it’s also known as a C3 cipher. the ROT12 cipher would turn an *A* into an *M,* a *B* into an *N* , and so on. It’s vulnerable to a type of attack known as frequency analysis.**

**Note** You should also know that data in transit is also commonly called data “on the wire,” referring to the network cables that carry data communications

**Cryptography basics**

four fundamental goals: confidentiality, integrity, authentication, and nonrepudiation.

Confidentiality is for data at rest and data in motion/on wire.

Integrity is for no alteration and it is enforced through digital signatures.

Authentication uses challenge response authentication protocol

Non -repudiation is offered only by asymmetric/public key cryptosystems.

A**ll cryptographic algorithms rely on keys to maintain their security. For the most part, a key is nothing more than a number. It’s usually a very large binary numbe**r, but a number nonetheless. **Every algorithm has a specific *key space***. **The key space is the range of values that are valid for use as a key for a specific algorithm**. **A key space is defined by its *bit size****.* Bit size is nothing more than the number of binary bits (0s and 1s) in the key. The key space is the range between the key that has all 0s and the key that has all 1s. Or to state it another way, the key space is the range of numbers from 0 to 2 *n,* where n is the bit size of the key. So, a 128-bit key can have a value from 0 to 2 128

**The kerchoff principle**

All cryptography relies on algorithms. An algorithm is a set of rules, usually mathematical, that dictates how enciphering and deciphering processes are to take place. Most cryptographers follow the Kerchoff principle, a concept that makes algorithms known and public, allowing anyone to examine and test them. Specifically, the Kerchoff principle (also known as Kerchoff’s assumption) is that a cryptographic system should be secure even if everything about the system, **except the key,** is public knowledge. **The principle can be summed up as “The enemy knows the system.”**

**Note: Cryptographic keys are sometimes referred to as *cryptovariables* .**

**The art of creating and implementing secret codes and ciphers is known as *cryptography.***

**This practice is paralleled by the art of cryptanalysis —the study of methods to defeat codes and ciphers. Together, cryptography and cryptanalysis are commonly referred to as cryptology .**

**Specific implementations of a code or cipher in hardware and software are known as *cryptosystems* . Federal Information Processing Standard (FIPS) 140–2, “Security Requirements for Cryptographic Modules,” defines the hardware and software requirements for cryptographic modules that the federal government uses**

**AND (Multiplication)**

The AND operation (represented by the ∧ symbol) checks to see whether two values are both true. truth table that only one combination of inputs (where both inputs are true) produces an output value of true:

**Ex : 1 1=1 ; 0 0=0 ; 1 0 =0 ;0 1= 0**

**OR (Addition)**

The OR operation (represented by the ∨ symbol) checks to see whether at least one of the input values is true. the only time the OR function returns a false value is when both input values are false:

**EX:1 1=1 ; 1 0=1 ;0 1=1 ; 0 0=0**

**NOT**

The NOT operation (represented by the ∼ or ! symbol) simply reverses the value of an input variable. This function operates on only one variable at a time.

**Ex :1 0 ; 0 1**

**Exclusive OR (Subtraction )**

The final logical function you’ll examine in this chapter is perhaps the most important and most commonly used in cryptographic applications—the exclusive OR (XOR) function. It’s referred to in mathematical literature as the XOR function and is commonly represented by the ⊕ symbol. The XOR function returns a true value when only one of the input values is true. If both values are false or both values are true, the output of the XOR function is false.

**Ex : 1 1 = 0; 0 0= 0 ; 1 0=1; 0 1=1**

**Modulo Function**

The *modulo* function is extremely important in the field of cryptography. The modulo function is usually represented in equations by the abbreviation *mod* ,although it’s also sometimes represented by the % operator

**One-Way Functions**

**A *one-way function***is a mathematical operation that easily produces output values for each possible combination of inputs but makes it impossible to retrieve the input values. Public key cryptosystems are all based on some sort of one-way function. In practice, however, it’s never been proven that any specific known function is truly one way. Cryptographers rely on functions that they suspect may be one way, but it’s theoretically possible that they might be broken by future cryptanalysts. **might be attacked using a computer and a brute force algorithm**, but there’s no easy way to figure it out in your head, that’s for sure.

**Nonce**

Cryptography often gains strength **by adding randomness to the encryption process. One method by which this is accomplished is through the use of a nonce. A *nonce* is a random number that acts as a placeholder variable in mathematical functions. Whe**n the function is executed, the nonce is replaced with a random number generated at the moment of processing for one-time use. The nonce must be a unique number each time it is used. One of the more recognizable examples of a nonce is an initialization vector (IV), a random bit string that is the same length as the block size and is XORed with the message. IVs are used to create unique ciphertext every time the same message is encrypted using the same key.

**Zero-Knowledge Proof**

One of the benefits of cryptography is found in the mechanism to prove your knowledge of a fact to a third party without revealing the fact itself to that third party. This is often done with passwords and other secret authenticators. The classic example of a *zero-knowledge proof* involves two individuals: P *f* eggy and Victor. Peggy knows the password to a secret door located inside a circular cave. For example, Victor would like to buy the password from Peggy, but he wants Peggy to prove that she knows the password before paying her for it. Peggy doesn’t want to tell Victor the password for fear that he won’t pay later. The zero-knowledge proof can solve their dilemma. Victor can stand at the entrance to the cave and watch Peggy depart down the path. Peggy then reaches the door and opens it using the password. She then passes through the door and returns via path 2. Victor saw her leave down path 1 and return via path 2, proving that she must know the correct password to open the door.

**Split Knowledge**

When the information or privilege required to perform an operation is divided among multiple users, no single person has sufficient privileges to compromise the security of an environment. **This separation of duties and two-person control contained in a single solution is called *split knowledge* .**

**Work Function**

You can measure the strength of a cryptography system by measuring the effort in terms of cost and/or time using a work function or work factor. Usually the time and effort required to perform a complete brute-force attack against an encryption system is what the work function represents.

An easy way to keep the difference between codes and ciphers straight is to remember that **codes work on words and phrases whereas ciphers work on individual characters and bits.**

**Transposition Ciphers**

Transposition ciphers use an encryption algorithm to rearrange the letters of a plaintext message, forming the ciphertext message. The decryption algorithm simply reverses the encryption transformation to retrieve the original message.

**Substitution Ciphers**

Substitution ciphers use the encryption algorithm to replace each character or bit of the plaintext message with a different character.

You can express the ROT3 cipher in mathematical terms by converting each letter to its decimal equivalent (where *A* is 0 and *Z* is 25). You can then add three to each plaintext letter to determine the ciphertext. You account for the wrap-around by using the modulo function

C = (P + 3) mod 26

Polyalphabetic substitution ciphers use multiple alphabets in the same message to hinder decryption efforts. One of the most notable examples of a polyalphabetic substitution cipher system is **the Vigenère cipher**

Although polyalphabetic substitution **protects against direct frequency analysis**, it **is vulnerable to a second-order form of frequency analysis called *period analysis***, which is an examination of frequency based on the repeated use of the key

**One-Time Pads**

A *o ne-time pad* is an extremely powerful type of substitution cipher. One-time pads use a different substitution alphabet for each letter of the plaintext message. They can be represented by the following encryption function, where *K* is the encryption key used to encrypt the plaintext letter *P* into the ciphertext letter *C* :

C = (P + K) mod 26

Usually, one-time pads are written as a very long series of numbers to be plugged into the function.

**One-time pads are also known as *Vernam ciphers* ,**

The great advantage of one-time pads is that, when used properly, **they are an unbreakable encryption scheme**. There is no repeating pattern of alphabetic substitution, rendering cryptanalytic efforts useless. However, several requirements must be met to ensure the integrity of the algorithm:

■ The one-time pad must be randomly generated. Using a phrase or a passage from a book would introduce the possibility that cryptanalysts could break the code.

■ The one-time pad must be physically protected against disclosure. If the enemy has a copy of the pad, they can easily decrypt the enciphered messages.

Each one-time pad must be used only once. If pads are reused, cryptanalysts can compare similarities in multiple messages encrypted with the same pad and possibly determine the key values used.

■ The key must be at least as long as the message to be encrypted. This is because each character of the key is used to encode only one character of the message.

Caesar cipher, Vigenère cipher, and one-time pad sound very similar. They are! The only difference is the key length. The Caesar shift cipher uses a key of length one, **the Vigenère cipher uses a longer key (usually a word or sentence), and the onetime pad uses a key that is as long as the message itself**.

One-time pads have been used throughout history to protect extremely sensitive communications. The major obstacle to their widespread use is the difficulty of generating, distributing, and safeguarding the lengthy keys required. One-time pads can realistically be used only for short messages, because of key lengths.

**Running Key Ciphers**

Many cryptographic vulnerabilities surround the limited length of the cryptographic key. As you learned in the previous section, one-time pads avoid these vulnerabilities by using a key that is at least as long as the message. However, one-time pads are awkward to implement because they require the physical exchange of pads. One common solution to this dilemma is the use of a *running key cipher* (also known as a *book cipher* ). In this cipher, the encryption key is as long as the message itself and is often chosen from a common book. For example, the sender and recipient might agree in advance to use the text of a chapter from *Moby Dick* , beginning with the third paragraph, as the key. They would both simply use as many consecutive characters as necessary to perform the encryption and decryption operations.

**Block Ciphers**

*Block ciphers* operate on “chunks,” or blocks, of a message and apply the encryption algorithm to an entire message block at the same time. **The transposition ciphers are examples of block ciphers**. The simple algorithm used in the challenge-response algorithm takes an entire word and reverses its letters. The more complicated columnar transposition cipher works on an entire message (or a piece of a message) and encrypts it using the transposition algorithm and a secret keyword. Most modern encryption algorithms implement some type of block cipher.

**Stream Ciphers**

*Stream ciphers* operate on one character or bit of a message (or data stream) at a time. The Caesar cipher is an example of a stream cipher. **The one-time pad is also a stream cipher because the algorithm operates on each letter of the plaintext message independently**. Stream ciphers can also function as a type of block cipher. In such operations there is a buffer that fills up to real-time data that is then encrypted as a block and transmitted to the recipient.

**Confusion and Diffusion**

Cryptographic algorithms rely on two basic operations to obscure plaintext **messages confusion and diffusion. *Confusion* occurs when the relationship between the plain text and the key is so complicated that an attacker can’t merely continue altering the plain text and analyzing the resulting ciphertext to determine the key**. ***Diffusion* occurs when a change in the plain text results in multiple changes spread throughout the ciphertext.** Consider, for example, a cryptographic algorithm that first performs a complex substitution and then uses transposition to rearrange the characters of the substituted ciphertext. In this example, **the substitution introduces confusion and the transposition introduces diffusion**.

In key exchange an offline key distribution method must often be used is called , out-of-band exchange).

**Hashing Algorithms**

It’s extremely diffi cult, if not impossible, to derive a message from an ideal hash function, and it’s very unlikely that two messages will produce the same hash value.The following are some of the more common hashing algorithms in use today:

■ Message Digest 2 (MD2)

■ Message Digest 5 (MD5)

■ Secure Hash Algorithm (SHA-0, SHA-1, and SHA-2)

■ Hashed Message Authentication Code (HMAC)

ECB -In everyday use, ECB is used only for **exchanging small amounts of data**, such as keys and parameters used to initiate other DES modes as well as the cells in a database

CBC-CBC implements **an IV and XORs it** with the first block of the message, producing **a unique output every time the operatio**n is performed. One important consideration when using CBC mode is **that errors propagate**—if one block is corrupted during transmission, it becomes impossible to decrypt that block and the next block as well.

CFB-Cipher Feedback (CFB) mode is the streaming cipher version of CBC. In other words, CFB operates against data produced in real time. However, instead of breaking a message into blocks, **it uses memory buffers of the same block size**

OFB-In Output Feedback (OFB) mode, DES operates in almost the same fashion as it does in CFB mode. However, instead of XORing an encrypted version of the previous block of ciphertext, **DES XORs the plain text** with a seed value.

The major advantages of OFB mode are that **there is no chaining function and transmission errors do not propagate to affect the decryption of future blocks**

CTR-it uses a simple **counter that increments** for each operation. As with OFB mode, errors **do not propagate in CTR mode.**

3DES

There are four versions of 3DES. The fi rst simply encrypts the plain text three times, using three different keys: K1 , K 2 , and K3 . It is known as DES-EEE3 mode (the *E* s indicate that there are three encryption operations, whereas the numeral 3 indicates that three different keys are used). DES-EEE3 can be expressed using the following notation, where

E(K,P) represents the encryption of plaintext *P* with key *K* :

E(K1,E(K2,E(K3,P)))

DES-EEE3 has an effective key length of 168 bits.

**IDEA operates** on **64-bit blocks of plain** text/ciphertext. However, it begins its operation with a **128-bit key**. This key is broken up in a series of operations into **52 16-bit subkeys**. The subkeys then act on the input text using a combination of XOR and modulus operations to produce the encrypted/decrypted version of the input message. IDEA is capable of operating in the same five modes used by DES: ECB, CBC, CFB, OFB, and CTR.

The IDEA algorithm is **patented by its Swiss developers**. However, they have granted an unlimited license to anyone who wants to use IDEA for noncommercial purposes. One popular implementation of IDEA is found in Phil Zimmerman’s popular Pretty Good Privacy (PGP) secure email package

**Blowfish**

Bruce Schneier’s Blowfish block cipher is another alternative to DES and IDEA. Like its predecessors, Blowfish operates on **64-bit blocks of** text. However, it extends IDEA’s key strength even further by allowing the use of variable-length keys ranging from a relatively insecure **32 bits to an extremely strong 448 bits**. Obviously, the longer keys will result in a corresponding increase in encryption/decryption time. However, time trials have established Blowfish as **a much faster algorithm** than both IDEA and DES. Also, Mr. Schneier released Blowfish for public use with **no license required**. Blowfish encryption is built into a number of commercial software products and operating systems. A number of Blowfish libraries are also available for software developers.

**Skipjack**

The Skipjack algorithm was approved for use by the US government in Federal Information Processing Standard (FIPS) 185, the Escrowed Encryption Standard (EES). Like many block ciphers, Skipjack operates on 64-bit blocks of text. It uses an 80-bit key and supports the same four modes of operation supported by DES. Skipjack was quickly embraced by the US government

**Rivest Cipher 5, or RC5**, is a symmetric algorithm patented by Rivest, Shamir, and Adleman (RSA) Data Security, the people who developed the RSA asymmetric algorithm. RC5 is a block cipher of variable block sizes (32, 64, or 128 bits) that uses key sizes between 0 (zero) length and 2,040 bits.

**The Two fish algorithm** developed by Bruce Schneier (also the creator of Blowfish) was another one of the AES finalists. Like Rijndael, Twofish is a block cipher. It operates on 128-bit blocks of data and is capable of using cryptographic keys **up to 256 bits in length**.

Twofish uses two techniques not found in other algorithms:

**Prewhitening** involves XORing the plain text with a separate g subkey before the first round of encryption.

**Postwhitening** uses a similar operation after the 16th round of encryption

**AES**

The AES cipher allows the use of three key strengths: 128 bits, 192 bits, and 256 bits. AES only allows the processing of 128-bit blocks, but Rijndael exceeded this specification, allowing cryptographers to use a block size equal to the key length.

**Diffie-Hellman** in some cases, neither public key encryption nor offline distribution is sufficient. Two parties might need to communicate with each other, but they have no physical means to exchange key material, and there is no public key infrastructure in place to facilitate the exchange of secret keys. In situations like this, key exchange algorithms like the Diffi e-Hellman algorithm prove to be extremely useful mechanisms.

1.The communicating parties (we’ll call them Richard and Sue) agree on two large numbers: *p* (which is a prime number) and *g* (which is an integer) such that 1 < g < p.

**2.** Richard chooses a random large integer *r* and performs the following calculation:

R = gr mod p

**3.** Sue chooses a random large integer *s* and performs the following calculation:

S = gs mod p

**4.** Richard sends *R* to Sue and Sue sends *S* to Richard.

**5.** Richard then performs the following calculation:

K = Sr mod p

**6.** Sue then performs the following calculation:

K = Rs mod p

At this point, Richard and Sue both have the same value, *K* , and can use this for secret

key communication between the two parties.

**Key escrow**

There are two major approaches to key escrow that have been proposed over the past decade:

**Fair Cryptosystems** In this escrow approach, the secret keys used in a communication are divided into two or more pieces, each of which is given to an independent third party. Each of these pieces is useless on its own but may be recombined to obtain the secret key. When the government obtains legal authority to access a particular key, it provides evidence of the court order to each of the third parties and then reassembles the secret key.

**Escrowed Encryption Standard** This escrow approach provides the government with a technological means to decrypt ciphertext. This standard is the basis behind the Skipjack algorithm discussed earlier in this chapter..