Cryptography Basics

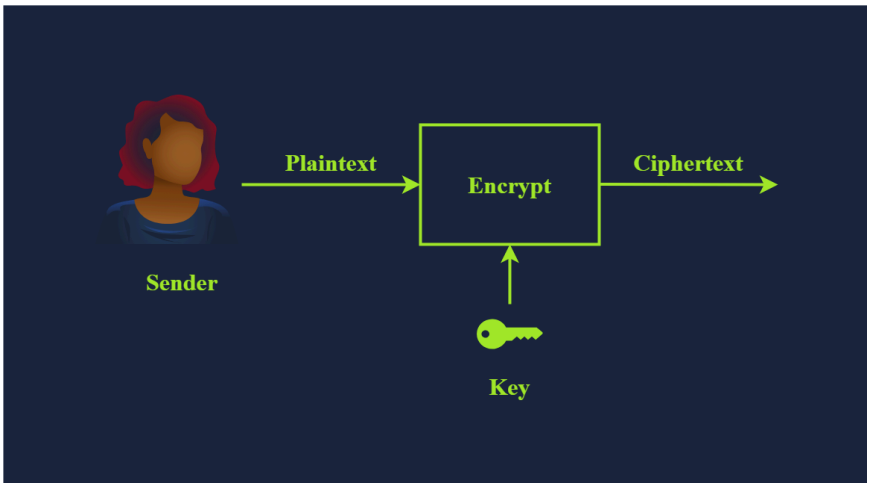
Introduction

Cryptography is the art of exchanging your data online without having the risk of a third party monitoring it.

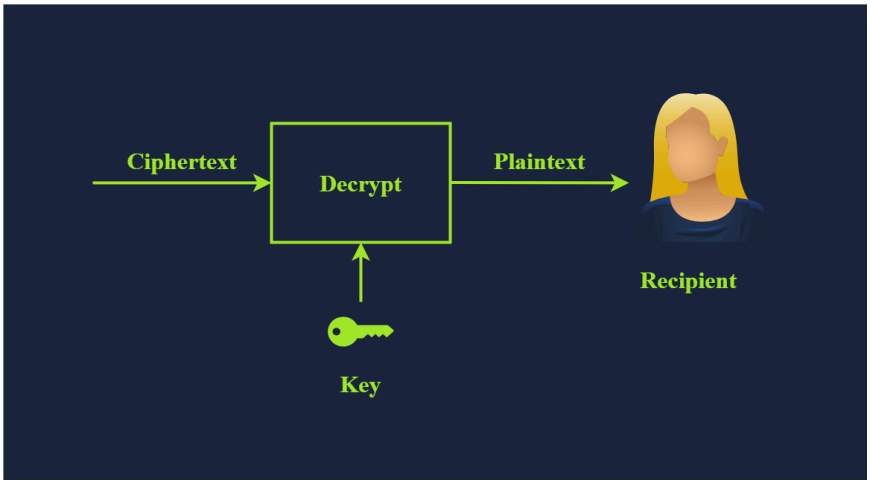
> https://listings.pcisecuritystandards.org/documents/PCI_DSS_for_Large_Organizations_v1.pdf

Plaintext to Ciphertext

During encryption process we begin with a the plaintext that we want to encrypt. The plaintext is readable data. Such as a picture, or a text, or a credit card information. The plaintext then is passed through the encryption function along with a proper key; the encryption function returns a ciphertext.



To recover the plaintext, we must pass the ciphertext along with the proper key via the decryption function, which would give us the original plaintext. This is shown in the illustration below.



We have just introduced several new terms, and we need to learn them to understand any text about cryptography. The terms are listed below:

- **Plaintext** is the original, readable message or data before it's encrypted. It can be a document, an image, a multimedia file, or any other binary data.
- **Ciphertext** is the scrambled, unreadable version of the message after encryption. Ideally, we cannot get any information about the original plaintext except its approximate size.
- **Cipher** is an algorithm or method to convert plaintext into ciphertext and back again. A cipher is usually developed by a mathematician.
- **Key** is a string of bits the cipher uses to encrypt or decrypt data. In general, the used cipher is public knowledge; however, the key must remain secret unless it is the public key in asymmetric encryption. We will visit asymmetric encryption in a later task.
- **Encryption** is the process of converting plaintext into ciphertext using a cipher and a key. Unlike the key, the choice of the cipher is disclosed.
- **Decryption** is the reverse process of encryption, converting ciphertext back into plaintext using a cipher and a key. Although the cipher would be public knowledge, recovering the plaintext without knowledge of the key should be impossible (infeasible).

Historical Ciphers

Most famous historical ciphers that I might have to keep in mind are:

- Caesar Cipher from the first century BCE

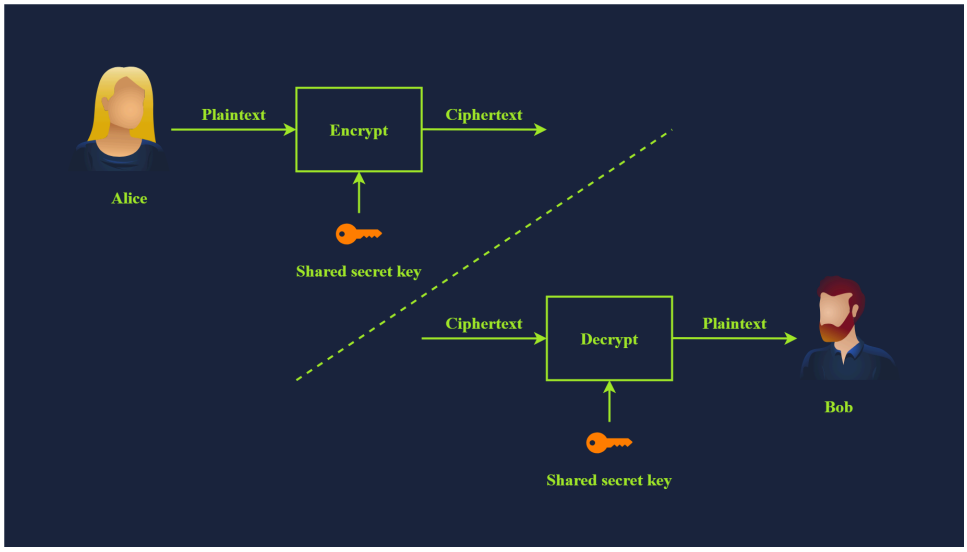
- The Vigenère cipher from the 16th century
- The Enigma machine from World War II
- The one-time pad from the Cold War

Types of Encryption

The two main categories of encryption are **symmetric** and **asymmetric**.

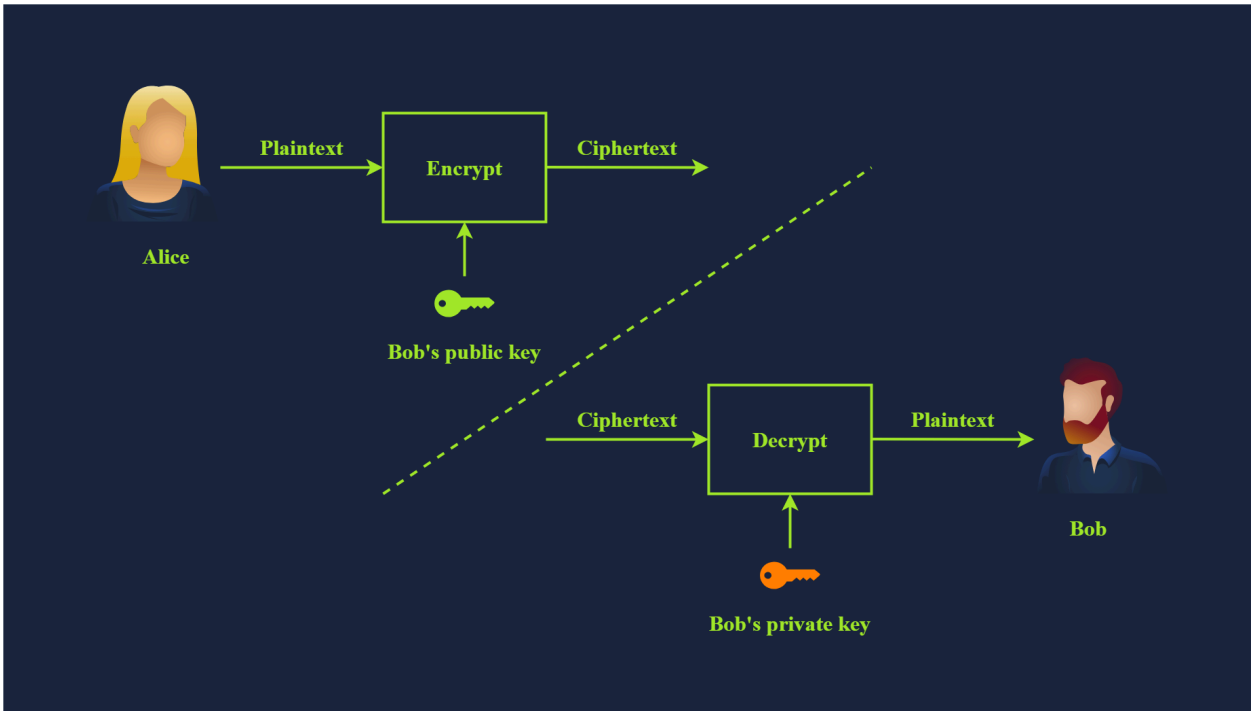
Symmetric Encryption

Symmetric encryption, also known as symmetric cryptography, uses the same key to encrypt an decrypt the data. As shown in the figure below. This method implies some risks and challenges as it requires a secure communication channel.



Asymmetric Encryption

Unlike symmetric encryption, which uses the same key for encryption and decryption, asymmetric encryption uses a pair of keys, one to encrypt and the other to decrypt, as shown in the illustration below. To protect confidentiality, asymmetric encryption or asymmetric cryptography encrypts the data using the public key; hence, it is also called public key cryptography.



Asymmetric encryption tends to be slower, and many asymmetric encryption ciphers use larger keys than symmetric encryption. For example, RSA uses 2048-bit, 3072-bit, and 4096-bit keys; 2048-bit is the recommended minimum key size. Diffie-Hellman also has a recommended minimum key size of 2048 bits but uses 3072-bit and 4096-bit keys for enhanced security. On the other hand, ECC can achieve equivalent security with shorter keys. For example, with a 256-bit key, ECC provides a level of security comparable to a 3072-bit RSA key.
the important thing to note is that asymmetric encryption provides you with a public key that you share with everyone and a private key that you keep guarded and secret.

Basic Math

The building blocks of modern cryptography lie in mathematics. To demonstrate some basic algorithms, we will cover two mathematical operations that are used in various algorithms:

XOR Operation

XOR, short for “exclusive OR”, is a logical operation in binary arithmetic that plays a crucial role in various computing and cryptographic applications. In binary, XOR compares two bits and returns 1 if the bits are different and 0 if they are the same, as shown in the truth table below. This operation is often represented by the symbol $\oplus$ or ^.

A	B	A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

Let’s consider an example where we want to apply XOR to the binary numbers 1010 and 1100. In this case, we perform the operation bit by bit: 1 $\oplus$ 1 = 0, 0 $\oplus$ 1 = 1, 1 $\oplus$ 0 = 1, and 0 $\oplus$ 0 = 0, resulting in 0110.
You may be wondering how XOR can play any role in cryptography. XOR has several interesting properties that make it useful in cryptography and error detection. One key property is that applying XOR to a value with itself results in 0, and applying XOR to any value with 0 leaves it unchanged. This means A $\oplus$ A = 0, and A $\oplus$ 0 = A for any binary value A. Additionally, XOR is commutative, i.e., A $\oplus$ B = B $\oplus$ A. And it is associative, i.e., (A $\oplus$ B) $\oplus$ C = A $\oplus$ (B $\oplus$ C).
Let’s see how we can make use of the above in cryptography. We will demonstrate how XOR can be used as a basic symmetric encryption algorithm. Consider the binary values P and K, where P is the plaintext, and K is the secret key. The ciphertext is C = P $\oplus$ K.

Now, if we know C and K, we can recover P. We start with C $\oplus$ K = (P $\oplus$ K) $\oplus$ K. But we know that (P $\oplus$ K) $\oplus$ K = P $\oplus$ (K $\oplus$ K) because XOR is associative. Furthermore, we know that K $\oplus$ K = 0; consequently, (P $\oplus$ K) $\oplus$ K = P $\oplus$ (K $\oplus$ K) = P $\oplus$ 0 = P. In other words, XOR served as a simple symmetric encryption

algorithm. In practice, it is more complicated as we need a secret key as long as the plaintext.

Modulo Operation

Another mathematical operation we often encounter in cryptography is the modulo operator, commonly written as % or as mod. The modulo operator, X%Y, is the remainder when X is divided by Y. In our daily life calculations, we focus more on the result of division than on the remainder. The remainder plays a significant role in cryptography.

Let’s consider a few examples.

- 25%5 = 0 because 25 divided by 5 is 5, with a remainder of 0, i.e., 25 = 5 × 5 + 0
- 23%6 = 5 because 23 divided by 6 is 3, with a remainder of 5, i.e., 23 = 3 × 6 + 5
- 23%7 = 2 because 23 divided by 7 is 3 with a remainder of 2, i.e., 23 = 3 × 7 + 2

An important thing to remember about modulo is that it’s not reversible. If we are given the equation $x\%5 = 4$, infinite values of x would satisfy this equation.

The modulo operation always returns a non-negative result less than the divisor. This means that for any integer a and positive integer n, the result of $a\%n$ will always be in the range 0 to $n - 1$.