**What is cryptography?**

Cryptography involves creating written or generated codes that allow information to be kept secret. Cryptography converts data into a format that is unreadable for an unauthorized user, allowing it to be transmitted without unauthorized entities decoding it back into a readable format, thus compromising the data.

Information security uses cryptography on several levels. The information cannot be read without a key to decrypt it. The information maintains its integrity during transit and while being stored. Cryptography also aids in nonrepudiation. This means that the sender and the delivery of a message can be verified.

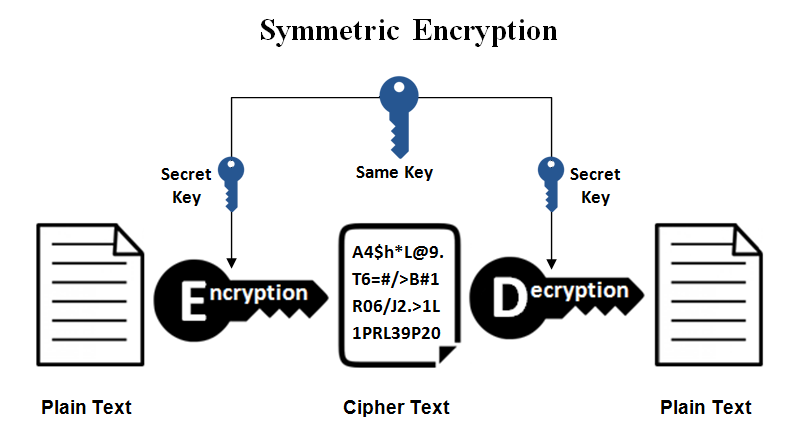
We can divide the cryptographic processing of messages in two major ways to manage the key that is used to decrypt the messages: symmetric and public key encryption.

**Symmetric key encryption**

Before we start, we must think as encryption and decryption in a symmetric encryption scheme as two algorithms that receive the same key to perform its routines. That is, we define the encryption scheme as two algorithms for message M, the symmetric key K and a ciphertext C as follows:

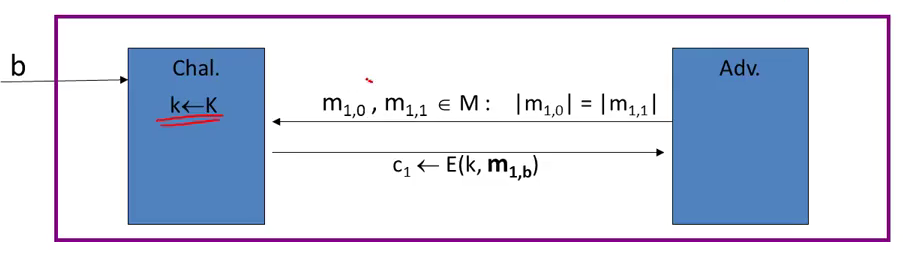
* **Encoder:** A function E (K, M) that outputs C.
* **Decoder:** A function D (K, C) that outputs M.

In a nutshell, the main characteristic in this scheme is that we use the same key to perform both tasks. This can be reduced in the following image, considering the “keys” as the symmetric key encryption functions.



**Semantic security (Chosen Plaintext Attacks)**

Using the same key to encrypt several messages is crucial to preserve the practicality in several environments, for example, assigning a specific key for a certain person to prevent that other people know the message that is intended to transmit. In the real world, the attackers have certain abilities that may help them to break the security of the encryption, one of them is the ability to request several ciphertexts for a set of messages that the attacker knows. So, the mission of the attacker is to guess which pair of cyphertext-message is valid for the key that is being used. This can be explained in the following image:



If the difference in probability in identifying the ciphertext to its corresponding message is negligible, then we say that the **encoder is semantically secure.**

A widely used encoder is AES, which counts with several optimizations in hardware such as Intel processors.

**Ciphertext integrity (Message Authentication Code)**

Although we can prevent the attacker of collecting the information of our messages, the attacker still could tamper the ciphertext to modify the message that is intended to be given. This is extremely dangerous because there exist several environments in which the correct use of information is crucial, for example, if the quantity in a bank deposit is altered, then it could result catastrophic for the economical status of the account. Hence, the MACs were developed as a signing algorithm to prevent whatever message an attacker would wish to be decrypted be processed. So, we define a MAC as a pair of algorithms for a message M, an identifier T and a secret key K such that:

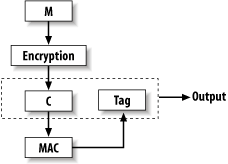
* **Signing Algorithm:** A function S (K, M) that outputs T.
* **Verification Algorithm:** A function V (K, M, T) that outputs ‘yes’ if the identifier matches with the evaluation of the signing algorithm.

We define that a MAC provides Ciphertext Integrity (CI) if and only if the attacker can produce a valid tag for a new message. That is, **if the attacker can only produce a tag for the message that is intended to be given for all messages possible, then the MAC provides CI.**

A widely used MAC is CBC-MAC, which is used in several standards of encryption.

**Authenticated encryption**

This mode of encryption allows us to produce a ciphertext that provides both **Semantic Security under Chosen Plaintext Attack and Ciphertext Integrity.** Hence, if we use the previously exposed knowledge to create an algorithm of this characteristics to produce a new algorithm that meets tis requirements, then we must mix the two types of algorithm, we just have to be sure to mix the algorithms in the correct order. The right way to perform this task is using the **Encrypt-then-MAC paradigm** to produce this kind of ciphertext. This is reduced in the following diagram:



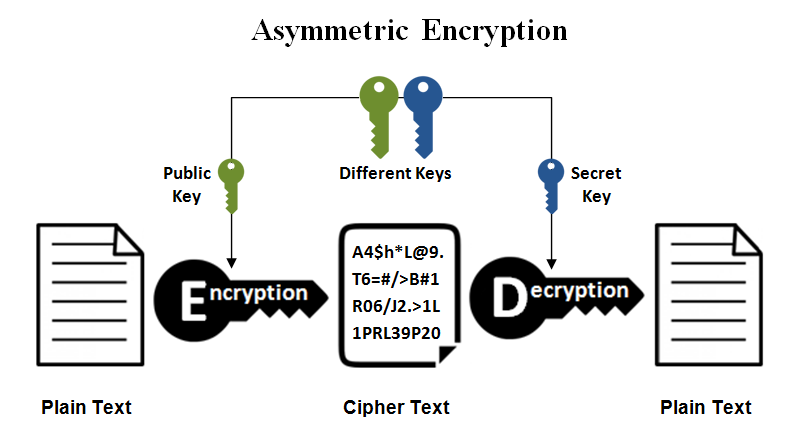
**Public key encryption**

**The big problem with symmetric encryption**

When two people want to stablish a conversation (i.e. exchange encrypted messages), the key that is used to decrypt needs to be sent to make both sides of the exchange capable of decrypting the messages, so the scheme is vulnerable from an attacker “listening” to the conversation as **the attacker can intercept both ciphertext and the key easily**. So, we need to establish a way the both parts of the exchange can generate the secret key without making the attacker capable of generating it by himself.

**The public key encryption scheme**

The public key encryption scheme is described as the following diagram:



Taking in count the previous image, we define the public key encryption scheme as a triple of algorithms for a public key PK, a secret key SK, a message M and a ciphertext C such that:

* **Generator:** a function G () that outputs the key pair (PK, SK)
* **Encoder:** a function E (PK, M) that outputs the ciphertext C.
* **Decoder:** a function D (SK, C) that outputs the message M.

**Important:** The secret key is always used to decrypt the cyphertext and it cannot be used to encrypt the messages.

**This scheme is widely deployed in several protocols and is used to implement the RSA encryption scheme.**

**Modular arithmetic**

In modular arithmetic, we perform all the operations inside of a finite set whose limit is an integer called the modulo. Specifically, the set is defined as the group of integers from 0 to the greatest integer less than the modulo or the limit minus one.

Generally, we define the modulo as:

Any multiple of the modulo is considered as 0 in its own set. Intuitively, we may think this aspect as counting in a cyclic way until we suffice the magnitude of the number. If we extrapolate this idea, for all integers possible and for all positive integers we will encounter this formula.

To perform arithmetic operations, we only have to perform the operation as normal and then traduce it to the modulo desired.

**The set of invertible numbers**

Firstly, we will look to Euclid’s theorem that makes a reference to another way to represent two numbers that have a greatest common divisor of one, in other words, are **relative primes**.

If:

Then exists a pair of integers that satisfies this equation:

If we define the inverse of a number as another integer whose multiplication with the number is equal to one, with the theorem previously shown, we are able to proof that an invertible number in the modulo set is invertible, if and only is a **relative prime of the modulo** by just traducing the Euclid’s formula to a number that is from the set. That is:

Hence, we have proofed that there exists an whose multiplication with the number equals to one, formally, it is an inverse of the number.

In modular notation, we define as a “star set” the group of invertible numbers in a certain modulo, in other words, the set of numbers whose greatest common divisor between its modulo and the integer is one. For example:

Or if we look at a star set with a prime modulo:

**Theorems of modular exponentiation**

**Fermat’s little theorem**

If we take a number of a star set that has a prime modulo, then that number to the power of the modulo minus one is equal to one.

**Euler’s theorem**

We must look first at Euler’s phi function which is defined as the number of elements of a star set with a specific modulo. For example:

Euler’s theorem says that a number to the power of the phi function equals to one in the modulo of the star set used to compute the function. That is:

**RSA implementation**

The triple of algorithms of this asymmetric encryption scheme is defined as follows:

**The key pair generator g**

The first step is to select the modulo which consists in the product of two primes, its size is of approximately 1024 bits.

Then, a pair of integers is generated whose product equals to one in the modulo of phi of N.

Finally, we stablish the pair of keys (public and secret) such that:

**The encryption algorithm**

This algorithm is as simple as computing the e’th power of the bits of our desire, that is:

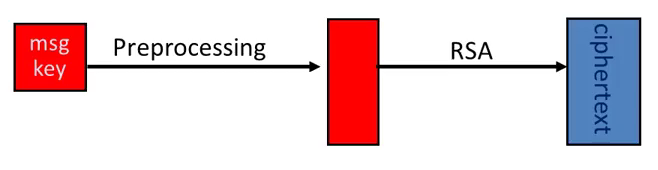
**The decryption algorithm**

The only action we perform in this step is calculating the encrypted bits to the power of d.

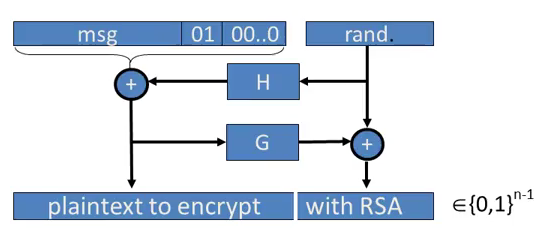
By Euler’s theorem:

**OAEP: a correct way to encrypt with RSA**

Anyone should never encrypt the plaintext away with RSA, the bits must be preprocessed first following this pad:



The following preprocessing scheme is widely deployed on various net protocols, it’s a good example how should we process the plaintext:



In practice, G and H are hash functions that implement SHA-256, which is known to be a secure algorithm for the task. However, this scheme is vulnerable to timing attacks, so one should never implement **RSA-OAEP by himself, instead we must use the OpenSSL library.**