Università degli Studi di Modena e Reggio Emilia

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica

Prova Finale

Titolo

Candidato:
Luca Lumetti
Matricola 118447

Relatore:
Prof. Maurizio Casoni

Correlatore:
Dott. Martin Klapez

Anno Accademico 2019-2020

Contents

Chapter 1

Introduction

Chapter 2

Cryptography Overview

Cryptography is the study of using digital coding to secure data accessed. In other words, to ensure that data can only be access by authorized entities.
To define cryptography jargon, we introduce a situation where an entity Alice wants to send a message to another entity Bob through a communication channel. Cryptography is relevant when there's an adversary that tries to access the data sent through the channel without legitimate authorization.
The *plaintext* is the message that Alice wants to send to Bob. The *ciphertext* is the data that goes through the channel and one of the resources that an adversary can access. The process of converting the plaintext to the ciphertext is called *encryption*, while the process to convert the ciphertext to the plaintext is called *decryption*. Encryption and decryption are defined by the *cryptographic scheme*, a set of algorithms that Alice and Bob decide to use before the actual communication, and one or more *keys*, that can be confidential and shared only between the authorized entities or they can also be public depending on the type of scheme used.
Regardless of the scheme's type used, which will be discussed later in detail, there are three main features that a cryptographic scheme should have to be defined secure: *confidentiality*, *integrity*, and *authentication*.

2.1 Confidentiality

Confidentiality assures that in a communication, an adversary is unable to obtain any information about the messages exchanged or the key used to encrypt them. This means that the ciphertext should appear to the adversary as completely random bits.

2.1.1 Perfectly Secret

We denote with $\mathcal{M}$, $\mathcal{K}$, $\mathcal{C}$ the message space, key space, and ciphertext space, respectively, with $\mathsf{Pr}[M = m]$ the probability that a message m is sent and with $\mathsf{Pr}[C = c]$ the probability that the ciphertext is c.

We can define a cryptographic scheme to be perfectly secret if for every $m \in \mathcal{M}$, every $k \in \mathcal{K}$:

$$\mathsf{Pr}[M = m|C = c] = \mathsf{Pr}[M = m]$$

This means that the distribution over $\mathcal{M}$ is independent of the distribution over $\mathcal{C}$. To achieve this definition, the key space $\mathcal{K}$ must be greater than the message space $\mathcal{M}$. This can be impractical and inconvenient because perfect secrecy is defined against an adversary with unbounded computational power. We can relax this latter constraint to be secure against polynomial-time algorithms.

2.1.2 Computationally Secret

Computational security is the aim of most modern cryptographic schemes. Modern encryption schemes can be broken given enough time and computation, nevertheless, the time required even for the most powerful supercomputer today built is in the order of hundreds of years.

From the previous definition of perfect secrecy, we add two relaxations:

- Security is only preserved against efficient adversaries.
- Adversaries can potentially succeed with a negligible probability.

With the term efficient, we refer to an algorithm that can be carried out in *probabilistic polynomial time* (PPT). An algorithm A is said to run in polynomial time, if there exists a polynomial $p(\cdot)$ such that for every $x \in \{0, 1\}^*$, $A(x)$ terminates within at most $p(|x|)$. A probabilistic algorithm is one that has access to some randomness so its results depend on changes.

With negligible probability, we refer to a probability asymptotically smaller than the inverse of every polynomial $p(\cdot)$. So, a function $f(\cdot)$ is $\mathsf{negligible}$ (typically denoted with negl) if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

2.1.3 Types of Attacks

Based on the capableness of the adversary, we can define different types of attack that can be carried out against a scheme, which are:

- **Ciphertext-only attack:** is the case when the attacker can only access the ciphertext and try to determine the plaintext that was encrypted. In this case, the attacker is also called "eavesdropper".
- **Known-plaintext attack:** in this attack, the adversary learns one or more pairs of plaintexts/ciphertexts encrypted under the same key. The objective of the attacker is to determine the corresponding plaintext of a ciphertext that has not been known yet.
- **Chosen-plaintext attack (CPA):** the adversary can obtain the encryption of any plaintext of his choice. Again, the adversary aims to decrypt a ciphertext to get the relative plaintext.
- **Chosen-ciphertext attack (CCA):** the final and stronger type of attack. Here the adversary can to encrypt any plaintext and decrypt any ciphertext of its choice. Once again the aim is the same as the previous attacks, but with the constraint that the ciphertext that it wants to crack can't be directly decrypted.

2.2 Integrity

Integrity assures to the receiver that a message is not corrupted or that an adversary has not modified and relayed it (for example in a man-in-the-middle attack).
The decryption of the message is not always needed to modify it, but can be enough to have the ciphertext. An example is shown in section 3.5.
Hash functions are used to assure the integrity of a message.

2.2.1 Hash Functions

In general, hash functions are just functions that take arbitrary-length strings and compress them into shorter strings. A hash function is a pair $(\mathsf{Gen}, \mathsf{H})$ such that:

- Gen: is a randomized algorithm that takes as input a security parameter n and outputs a key s.

- H: is a deterministic polynomial-time algorithm that takes as input a string $x \in \{0,1\}^*$ and a key s to output a string $\mathsf{H}^s(x) \in \{0,1\}^{l(n)}$ where l is a polynomial.

In practice, hash functions are unkeyed or, rather, the key is included in the function itself.

As an example of use of hash functions, imagine that Alice wants to send a message m to Bob and he also wants to assure its integrity. After they both agree on the hash function to use, Alice sends $(m, \mathsf{H}(m))$, then upon receiving the pair, Bob itself calculates $\mathsf{H}(m)$ and verifies that it is the same it received from Alice. If they match, m can be considered intact.

The domain of H is unlimited, instead, its image is limited. For the pigeon-hole principle this means that there are infinite pairs of differents string x and x' such that $\mathsf{H}(x) = \mathsf{H}(x')$, this is also known as a *collision*.

2.2.2 Collision-resistant hash functions

Hash functions used in cryptography are also called collision-resistant hash function, to emphasize the importance to have the property that no polynomial-time adversary can reverse them in a reasonable time. There are 3 levels of security:

1. **Collision resistance:** is the most secure level and implies that, given the key s, is infeasible to find two different values x and x' such that $\mathsf{H}^s(x) = \mathsf{H}^s(x')$.

2. **Second preimage resistance:** implies that, given s and a string x, is infeasible for a polynomial-time algorithm to find a string x' such that it collides with x

3. **Preimage resistance:** implies that given the key s and an hash y, is infeasible for a polynomial-time algorithm to find a value x such that $\mathsf{H}^s(x) = y$.

Notice that every hash function that is collision resistant is second preimage resistant, also a second preimage resistant function is a preimage resistant function.

2.2.3 The Merkle-Damgård Transform

Even if we have defined hash functions as functions with an infinite domain, in practice they are first constructed to be *fixed-length*, that means their domain is finite, then they are extended to cover the full domain $\{0,1\}^*$.

This extension is made easy by the Merkle-Damgård transform which, also preserves the collision-resistant propriety.

We will denote the given fixed-length collision-resistant hash function (or *compression function*) by (Gen_h, h) and use it to construct a general collision-resistant hash function (Gen, H) that maps inputs of any length to output of length $l(n)$.

Let (Gen_h, h) be a fixed-length hash function with input length $2l(n)$ and output $l(n)$. Construct a variable-length hash function (Gen, H) as follow:

- $\mathsf{Gen}(n)$: upon input n, run the key-generation algorithm: $s \leftarrow \mathsf{Gen}_h$.
- $H^s(x)$: upon input key s and message $x \in 0, 1^*$, compute as follows:
    1. Pad x with zeroes until it's length is a multiple of $l(n)$. Let $L = |x|$ (length of the string) and let $B = \left\lceil \frac{L}{l(n)} \right\rceil$ (number of blocks of length $l(n)$).
    2. Define $z_0 := 0^{l(n)}$ and then $\forall i = 1, ..., B$, compute:

       $$z_i := h^s(z_{i-1}||x_i)$$

       where h^s is the given fixed-length hash function.
    3. Output $z = H^s(z_B||L)$.

We remark that the value z_0, also known as *IV* or *initialization vector* can be replaced with any constant of length $l(n)$ bits.

2.2.4 Authentication

In a cryptographic scheme, authentication is needed to authenticate the entities involved in the communication. In other words, authentication assures that messages received by Bob are for sure from Alice.

We talk about authentication in the integrity section because, the difference between these two concepts is blurry, moreover, authentication implies integrity (but not vice versa).

In symmetric schemes, we will see *messages authentication codes* (MAC) and in asymmetric schemes, we will see *digital signatures algorithms* (DSA).

Chapter 3

Private Key Cryptography

With *private-key cryptography* we refer to schemes that use a single key to encrypt and decrypt a message, for this reason, we will refer to them as *symmetric* schemes. A **private-key scheme** is a tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- $\mathsf{Gen}(\cdot)$**:** is a randomized polynomial algorithm that generates the key. It takes as input a security parameter n and outputs a key k that satisfies $|k| \geq n$.
  We will write this as $k \leftarrow \mathsf{Gen}(1^n)$.

- $\mathsf{Enc}(\cdot)$**:** is a probabilistic polynomial-time algorithm that encrypts the message (or other forms of information) to send. It takes as input a key k and a message m to output a ciphertext c. We will refer to the unencrypted message also as plaintext.
  We will write this as $c \leftarrow \mathsf{Enc}_k(m)$.

- $\mathsf{Dec}(\cdot)$**:** is a deterministic polynomial-time algorithm that takes as input a ciphertext c and a key k, and outputs a plaintext m.
  We will write this as $m := \mathsf{Dec}_k(c)$

It's also required that for every n, every k and every m it holds that $m = \mathsf{Dec}_k(\mathsf{Enc}_k(m))$. Now we want to look which tools are used in the construction of secure private-key scheme.

3.1 Pseudorandom Permutations

Pseudorandom functions are functions that map n-bits strings to n-bit strings and that cannot be distinguished from a random permutation chosen, uniformly, from every function that map n-bit string to n-bit string. The first set of function, for a key of length s bit, have a cardinality of 2^s, while the second set has a cardinality of $2^{n \cdot 2^n}$.

Pseudorandom permutations are pseudorandom functions with some extra proprieties: Let $F : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^n$ be an efficient, length-preserving, keyed function and $F_k(m) := F(m,k)$. F is a pseudorandom permutation (PRP) if:

- $\forall k \in \{0,1\}^s$, F is a bijection from $\{0,1\}^n$ to $\{0,1\}^n$.
- $\forall k \in \{0,1\}^s$ exists an efficient algorith F_k^{-1}.
- For all probabilistic polynomial-time distinguishers D:

  $$|\mathsf{Pr}[D^{F_k}(n) = 1] - \mathsf{Pr}[D^{f_n}(n) = 1]| < \mathsf{negl}(n)$$

  where k is chosen uniformly at random from $\{0,1\}^s$ and f_n is chosen uniformly at random from the set of every permutations on n-bit strings.

3.2 Block Ciphers

Block ciphers are PRPs families that operate one a block of a fixed length. To ensure security against CPA, there are various mode of operations for block ciphers, like *Electronic Code Block* (ECB), *Cipher Block Chaining* (CBC), and *Counter Mode* (CTR).

3.2.1 Electronic Code Block

Given a plaintext $m = m_1, ..., m_l$, the encryption is obtained by encrypting each block separately: $c = \langle F_k(m_1), ..., F_k(m_l) \rangle$. The decryption is carried out by applying to every block F_k^{-1}. Since the encryption process is deterministic, repeated blocks will be repeated also in the ciphertext. This means that this mode is not CPA-secure neither has indistinguishable encryption in the presence of an eavesdropper.

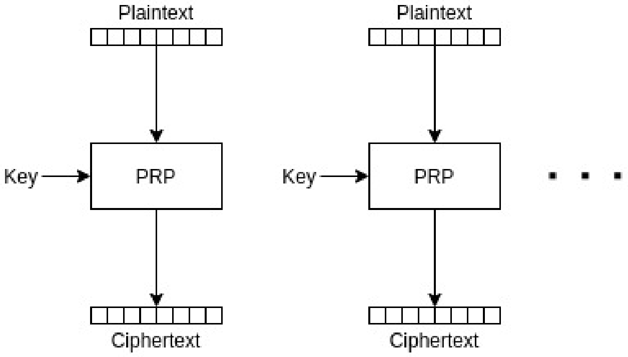

Figure 3.1: ECB encryption.

3.2.2 Cipher Block Chaining

First is an initial vector IV of length n is chosen. So is set $c_0 = IV$ and for every $i > 0$, $c_i := F_k(c_{i-1} \oplus m_i)$. The final ciphertext is $< IV, c_1, ..., c_l >$. The IV is not kept secret to allow decryption. The encryption of single blocks must be carried out sequentially

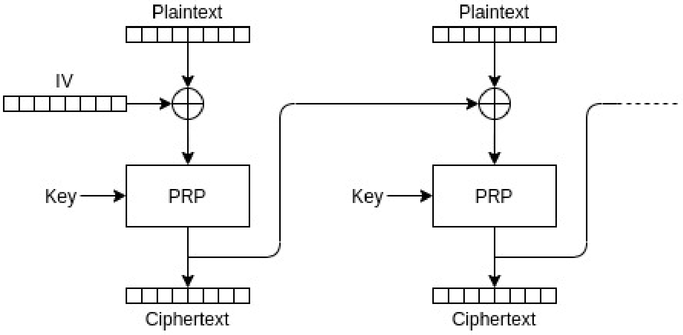

Figure 3.2: CBC encryption.

3.2.3 Randomized Counter Mode

As in CBC, an IV of length n is chosen. Then is computed $r_i := F_k((IV + i) \bmod 2^n)$. Then each block of the plaintext is computed as $c_i := r_i \oplus m_i$. Unlike in CBC, with CTR it's possible to encrypt and decrypt in parallel.

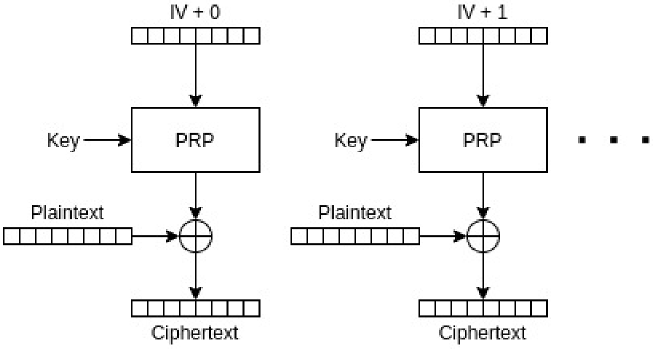

Figure 3.3: CTR encryption.

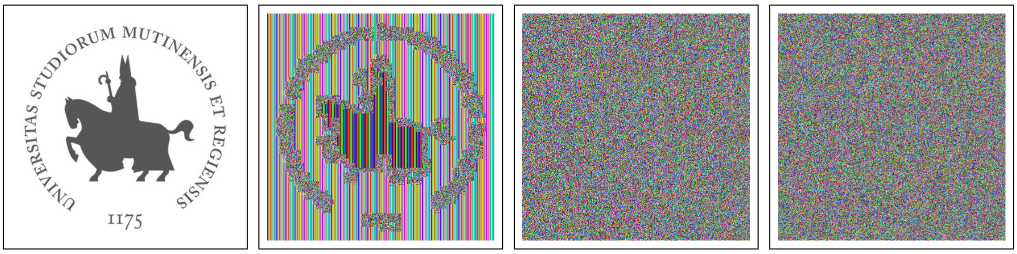

Figure 3.4: From left to right: original image, image encrypted with ECB, image encrypted with CTR, image encrypted with CBC. It's easy to notice the problem with ECB.

3.3 Substitution-Permutation Networks

3.3.1 Confusion-diffusion

The confusion-diffusion paradigm has been introduced by Shannon for concise construction pseudorandom functions. The idea is to break the input up into small parts, execute on them different random functions, mixed the outputs together and, repeat the process for a finite amount of time. One cycle of this process is called *round*, while the full construction is called *network*.

Shannon's original definitions are that confusion refers to making the relationship between the ciphertext and the key as complex as possible, diffusion refers to hide the relationship between the ciphertext and the plaintext. Confusion achieves the fact that each bit of the ciphertext depends on several parts of the key. This means that, even if a single bit of the key is changed, most of the bits in the ciphertext will be affected. Diffusion implies that changing a single bit in the plaintext results in a change of (statistically) half of the bits in the ciphertext. Also, if one bit of the ciphertext is changed, half of the bits of the plaintexts change.

3.3.2 Substitution-Permutation Networks

Substitution-permutation networks are a practical implementation of the confusion-diffusion paradigm. The substitution part is achieved by small random functions called *S-boxes* and the permutation part is achieved by mixing up the outputs of those functions. In the intermediate results, a key is XORed with the output of the round. Different keys are used each round and each key is derived from the previous one (that is called the *master key*).

3.4 Advanced Encryption Standard

AES is a common symmetric key block cipher used worldwide to protect data. It is the successor of *DES* (Data Encryption Standard), another block cipher that now has been classified insecure because of its length key, and after some attacks became too efficient. It has been chosen after a public competition held to meet the need for a new encryption standard.

3.4.1 Galois Field

A Galois Field (or finite field) is a field with a finite number of elements. The most common fields used are given by the integers $\mathsf{mod}\, p$ where p is a prime number, also written as $\mathsf{GF}[p]$. For $n > 1$, with $\mathsf{GF}[p^n]$ we refer to all polynomial of degree $n-1$ with coefficients coming from $\mathsf{GF}[p]$. As an example, $\mathsf{GF}[2^3]$ is a field with 8 elements (the integers from 0 to 7) and they can all be represented as a polynomial of degree 2 (6 can be represented as 110_2 or x^2+x).

Even if the sum between polynomials is trivial, the multiplication works differently: an irreducible polynomial $g(x)$ of degree n is chosen, then the multiplication in $\mathsf{GF}[p^n]$ is the ordinary product, except you have to take the remainder of the division by $g(x)$. A different $g(x)$ defines a different field.

3.4.2 Encryption Method

The security of AES comes from confusion and diffusion achieved by a substitution-permutation network with a block size of 128 bit and it supports three different key lengths: 128, 198, and 256 bits. Every key length uses a different number of round, respectively it uses 10, 12, and 14 rounds.

Each block of length 128 bits are rearranged in a 4x4 bytes array, called *state*, and for every round the following steps are executed:

1. **AddRoundKey:** A 16 byte round key is derived from the master key and it's interpreted as a 4x4 array. Then, the key is XORed with the state array. This step consist of computing $a_{i,j} = a_{i,j} \oplus k_{i,j}$ for every $i, j \in 1, ..., 4$ where $a_{i,j}$ is the i^{th} row and j^{th} column of the state array and $k_{i,j}$ is the i^{th} row and j^{th} column of the key array.

2. **SubBytes:** Each byte of the state array is substituted by another byte, according to a single fixed lookup table S. So, it consists in computing $a_{i,j} = S(a_{i,j})$.

3. **ShiftRows:** Each row of the state array are cyclically shifted to the left as follows: the first row of the array is untouched, the second row is shifted one place to the left, the third row is shifted two places to the left, and the fourth row is shifted three places to the left.

4. **MixColumns:** In this step, each column is mixed via an invertible linear transformation. Specifically, each column is interpreted as a polynomial over $\mathsf{GF}[2^8]$ with $g(x) = x^4 + 1$, and is multiplied with a fixed polynomial $c(x) = 3x^3 + x^2 + x + 2$.

In the final round, the MixColumns stage is replaced with an additional AddRoundKey step.

3.5 Message Authentication Codes

Everything we have seen until now assures only confidentiality. As introduced in chapter 2, message authentication codes (MAC), also called *tags*, are used to introduce both integrity and authentication.

Even if an encryption scheme assures only confidentiality, integrity and authentication are important to assure that the message is not tampered by an external entity. This is a property of encryption schemes called *malleability*.

For example, in CBC an attacker, because the IV and ciphertext blocks are directly XORed with the next block output, a bit changed in the IV or the ciphertext block corresponds with a bit changed in the plaintext. If the attacker can guess the format of the unencrypted message, the attacker could be able to change sensitive information on the plaintext.

A message authentication code (MAC) is a tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ such that:

- $\mathsf{Gen}(\cdot)$: it takes as input n and outputs a uniformly distributed key of length n.
  We will write this as $k \leftarrow \mathsf{Gen}(1^n)$.
- $\mathsf{Mac_k}(\cdot)$: It receive as input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^*$ to output a *tag* $t \in \{0,1\}^*$.
  We will write this as $t \leftarrow \mathsf{Mac}_k(m)$.
- $\mathsf{Vrfy_k}(\cdot,\cdot)$ on input $k \in \{0,1\}^n$, $m \in \{0,1\}^*$ and $t \in \{0,1\}^*$ it outputs a bit $b \in 0, 1$.
  We will write this as $b \leftarrow \mathsf{Vrfy}_k(m, t)$.

It also required that for every n, every k, and every m it holds that:

$$\mathsf{Vrfy}_k(m, \mathsf{Mac}_k(m)) = 1$$

The security of MAC is that an adversary can't forge a valid tag for a new message in a reasonable time.
The construction of MAC can be based on block ciphers, like *CBC-MAC*, or collision-resistant hash functions built with the Merkle-Damgård transform (see subsection 2.2.2 and subsection 2.2.3), like *NMAC* or *HMAC*.

3.5.1 HMAC construction

With $H^s_{\mathsf{IV}}(x)$ the hash function constructed with Merkle-Darmgård transform with z_0 set to an arbitrary value IV and we also define a keyed version of the compression function $h^s(x)$ used in H by $h^s_k(x) = h^s(k \mid\mid x)$. Two constants are also defined: opad and ipad of length n (the length of a single block of the input to H).
The string opad is formed by repeating the byte $\mathsf{0x5C}$ as many times needed and the string ipad is formed in the same way using the byte $\mathsf{0x36}$.
The HMAC construction is the same defined above in this section, with two additions:

- The Gen algorithm also run the key generation for the hash function obtaining the value s.
- The Mac_k algorithm is computed by:

$$\mathsf{HMAC}^s_k(x) = \mathsf{H}^s_{\mathsf{IV}}(k \oplus \mathsf{opad} \mid\mid \mathsf{H}_{\mathsf{IV}}(k \oplus \mathsf{ipad} \mid\mid x))$$

The hash function H can be any cryptographic hash function like SHA-1, SHA-256, etc..., and the relative HMACs are named HMAC-SHA1, HMAC-SHA256. The cryptographic strength of HMAC depends on the properties of the underlying hash function, so using hash functions like MD5 or SHA-1 is not recomended.

3.5.2 Chosen-Ciphertext Secure Encryption

By using Messages Authentication Codes with Block Ciphers, we are now able to build an encryption scheme that is secure against chosen-ciphertext attacks. To achieve this, the encryption scheme will have the property that the adversary is unable to forge ciphertext that was not generated by the legitimate users, so the decryption oracle, that the adversary can use, become useless.
The following definition is the join of a CPA-secure encryption scheme $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ and a secure message authentication code $(\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$:

- $\mathsf{Gen}'(\cdot)$: upon input n, choose $k_1 \leftarrow \mathsf{Gen}_E(n)$ and $k_2 \leftarrow \mathsf{Gen}_M(n)$.
- $\mathsf{Enc}'_k(\cdot)$: upon input key (k_1, k_2) and a message m, output the pair $(\mathsf{Enc}_{k_1}(m), \mathsf{Mac}_{k_2}(c))$.
- $\mathsf{Dec}'_k(\cdot, \cdot)$: upon input key (k_1, k_2) and the pair (c, t), where c is the ciphertext and t is the MAC tag, if $\mathsf{Vrfy}_{k_2}(c, t) = 1$, then output $\mathsf{Dec}_{k_1}(c)$, else output null.

Chapter 4

Public Key Cryptography

Until now we have seen how to achieve a secure communication over an insecure channel, but not discussed yet how keys are shared and managed. In fact these are one of the main problems of symmetric schemes, specially in open system like on the Internet. For a group of n entity where everyone wants to communicate with each others, the total number of secret keys that need to be generated is $\binom{n}{2} \approx \mathsf{O}(n^2)$ and these need to be distributed over a secure channel that is not always present. Even if partial solutions where built to overcome these problem, they were not sufficient. The first step to fully solve these problems was made in 1976 by Whitfield Diffie and Martin Hellman by publishing a paper called "New Directions in Cryptography". With this paper they laid the foundation for asymmetrics schemes. These schemes use 2 different keys called *public* and *private* key, the first one is used to encrypt the message while the second one to decrypt the ciphertext. So public key scheme is a tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- $\mathsf{Gen}(\cdot)$: takes as input the security parameter n and outputs a pair of key (pk, sk), respectively the public and the secret (private) key.

- $\mathsf{Enc}(\cdot)$: takes as input a public key pk and a message m to output the ciphertext $c \leftarrow \mathsf{Enc}_{pk}(m)$.

- $\mathsf{Dec}(\cdot)$: takes as input a secret key sk and a ciphertext c to output the message $m := \mathsf{Dec}_{sk}(c)$.

To establish a communication between 2 entities Alice and Bob, first the key pairs (A_{pk}, A_{sk}) and (B_{pk}, B_{sk}) are generated, then both public keys are shared among the entities and used to used to encrypt messages. Says tha Alice wants to send a message m to Bob, Alice craft the ciphertext using Bob's public key $c \leftarrow \mathsf{Enc}_{B_{pk}}(m)$ then send it to Bob that will decode it with his secret key $m = \mathsf{Dec}_{B_{sk}}(c)$.

In the scenario where n entities participate in the communication, the number of keys involved is $2n$ and every public key can be freely distributed throught insecure channels.

4.1 Group Theory and Modular Arithmetic

Public key scheme are deeply based on modular arithmetich and groups theory. This section define the notation used and some property and theorem used in this field.

4.1.1 Notation

We denote with N a positive integer, with p and q a prime and with $\mathbb{Z}_N$ the set of integers from 0 to $N-1$. This set is a group under addition modulo N but not under multiplication because not every element of $\mathbb{Z}_N$ has an inverse. If $x \in \mathbb{Z}_N$ we denote with $frac1x$ or x^{-1} the inverse of x, such that $x^{-1} \in \mathbb{Z}_N$ and $xx^{-1} = 1\ (mod\, N)$. As already said, in $\mathbb{Z}_N$ not every element as an inverse, for a x is possible to find an inverse if and only if $\mathsf{gcd}(x, N) = 1$.
With $\mathbb{Z}_N^*$ is denoted a subset of $\mathbb{Z}_N$ that only contains the elements that has an inverse and thus is a group even under multiplication modulo N. The cardinality of the group $\mathbb{Z}_N^*$ is denoted with the Euler function $\phi(N) = |\mathbb{Z}_N^*|$

4.1.2 Extended Euleclidean Algorithm

The Extended Euler's Algoritm is used to efficiencly find the inverse of an element $x \in \mathbb{Z}_N$. Given two integers a, b is possible to find two integers x, y such that satify $ax+by = \mathsf{gcd}(a, b)$, which is known as Bézout's identity. In the group $\mathbb{Z}_N$, is possible find the inverse of $a \in \mathbb{Z}_N$ that is co-prime with N by using the extended euclidean algorithm to solve the Bézout's identity:

$$ax + Ny = gcd(a, N) = 1 \implies ax = 1(modN)$$

Then x is the inverse of a, or $x = a^{-1}$.

4.1.3 Euler's Theorem

This theorem is a generalization of the Fermat Theorem and is used to simplify exponential operations over the group $\mathbb{Z}_N^*$. For an integer N define the Euler's ϕ function as $\phi(N) = |\mathbb{Z}_N^*|$, then for every N:

$$\forall x \in \mathbb{Z}_N^* : \quad x^{\phi(N)} = 1(mod\mathbb{Z}_N)$$

4.2 RSA

4.2.1 The RSA problem

This problem was first introduced by Rivest, Shamir, and Adleman to lay the basis for the implementation of a public key scheme. Informally, given N, and integer $e > 0$ co-prime with N and an element $y \in \mathbb{Z}_N^*$ find $x \in \mathbb{Z}_N^*$ such that $x^e = y(modN)$. This problem can be easly solved if $\phi(N)$ is known using the Euler's Theorem and the Extended Euclidian Algorithm. If N is semiprime, or the product of two prime $N = pq$ then $\phi(N)$ can be easly calculated if the factors of N are known: $\phi(N) = (p-1)(q-1)$. So the problem of computing $\phi(N)$ as hard as and factoring N, for which we don't have yet a polynomial algorithm this latter problem.
The asymmetry that, given two primes p and q, is easy to compute $N = pq$ but, given N is hard to find his factors, is exploited to built public keys encryption.

4.2.2 RSA Encryption

The RSA encryption scheme is defined by the Public-Key Cryptography Standards (PKCS) #1, today at version 2.2.

- $\mathsf{Gen}(n)$: two n-bits primes p and q are selected then $N = pq$ is computed. Then a value e (the encryption exponent) is selected such that e is co-primee to $\phi(N)$, and a value d (the decryption exponent) is calculated as $e^{-1}(mod\phi(N))$. The pair (N, e) will be the public key while (N, d) will be the private key.

- $\mathsf{Enc}_{N,e}(m)$: given the public key (N, e) and the message m, the ciphertext is computed as $c = m^e(modN)$.

- $\mathsf{Dec}_{N,d}(m)$: given the private key (N, d) and the ciphertext c, the message is computed as $m = m^d(modN)$.

In the encryption function, the message m is the original message padded using OAEP (Optimal Asymmetric Encryption Padding)

4.2.3 Optimal Asymmetric Encryption Padding

A padding in the encryption function is needed to construct a probabilistic encryption function thus be secure against chosen plaintext attacks. Usually the combination of OAEP with Textbook RSA is called RSA-OAEP.

Given a message m, the message $\hat{m}$ that will be encrypted is obtained by picking a random fixed length string r of length $2|m|$ and two hash functions G and H are chosen. Then, after computing the string $m' := \mathsf{G}(r) \oplus (m||0^{|m|})$, the final padded message is:

$$\hat{m} := m'||(r \oplus \mathsf{H}(m'))$$

After the decryption phase, to recover the original unpadded mesage, the decoded string $\hat{m}$ is splitted in half $\hat{m} = \hat{m_1}||\hat{m_2}$ with $|\hat{m_1}| = |\hat{m_2}|$. Then compute:

$$m' := \hat{m_1} \oplus \mathsf{G}(\mathsf{H}(\hat{m_1}) \oplus \hat{m_2})$$

If the last half of m' is $0^{|\frac{m'}{2}|}$ then the original message is the first half of m'.

4.3 Digital Signatures

Digital Signatures are the public-key counterpart of Message Authentication Codes in private-key because they both ensure integrity and authentication. Digital signatures has the advantage of simplify the key management and make a message publicly verifiable. A digital signature allow an entity A to sign a message such that everyone who know the public key of A has been not modified. Also, the digital signature certificate the ownership of a public key.

Digital signatures also have the property of *non-repudiation*, that is once A publicizes his public key and sign a message with latter, he can't deny having done so. This is impractical in MACs because the key used to forge the MAC must be kept secret and if it is publicized, then everyone can forge a valid MAC.

A digital signature scheme is a tuple $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that:

- Gen: is the same for public-key scheme, it takes as input the security parameter n to output a pair of keys (pk, sk) respectively the public and the private key.
- Sign: takes as input a private key sk and a message m to output a signature $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.
- $\mathsf{Vrfy}(\cdot)$: takes as input a signature σ, a public key pk and a message m. The output is a bit $b := \mathsf{Vrfy}_{pk}(m, \sigma)$. If the bit is 1 then the signature is valid otherwise is invalid.

4.3.1 Certificate Authority

Certificates authorities are thrusted third party entity that issue digital cetificates, used by an entity to check the ownership of a public key

Chapter 5

Conclusions

Bibliography

[1] John D. Cook. Finite fields (galois fields).

[2] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography.* CRC press, 2014.

[3] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Rfc2104: Hmac: Keyed-hashing for message authentication, 1997.

[4] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.