Computer and network security

Slide password: }DIAGamat1c{++

Introduction

The internet first started with **no security.** Then it has been patched, so technically it's not a good project. What is **information security?** There are many definitions, but we are now interested in the most important ingredients:

- **Confidentiality** (or privacy): we consider data and information as the same term, no difference. Confidentiality means that we must provide access to data only to people **who have the right** to access them. All the other people should not be able to access the information.
- **Data integrity:** if I'm storing data or I'm obtaining data maybe downloading some information, I want to be sure that the data I'm obtaining are **the original data**. They must not change, I must be sure about this. What if an attacker makes a change and provide you another file? This is the most frequent attack, called the **man in the middle attack.** The MITM can check the content of every communication and he can also change that content.
- **Data authentication:** I'm downloading the right file that is on the web, that's ok, but that file was what the original publisher meant to upload? Data integrity and authentication often go together.
- **Authentication:** You often authenticate yourself with username and password. Well, that's the worst way to make authentication. Very weak, but we still use it.
- **Availability:** Information should be available to all people who have the right to access it.

Cryptography

What is cryptography? In some books you will read that cryptography is the **art of hiding information**, changing information into something which looks different in such a way that people can't understand the real content without the algorithm. **Modern cryptography** has invented new tools to ensure data integrity. Today we study cryptography to ensure all the properties we talked about before. Modern cryptography is going further, not only encryption.

Cryptography will give you a **mathematical model** which you will transform into a running program. We will check then the system security like firewall's work. When you download a file, you have permissions for it. Talking about security, we basically have two types of attacks. The first is **automatic tools attacks.** When you just connect to the internet, you can check that you are receiving a lot of attacks. This is because there are some automatic tools that just go over **the existent 4 billions IP** numbers and check wether there is an open port and so on. We don't see that because we have firewall in our computers. We also have **target attacks**, when someone collects information about you for months, years, and then attack you. They are very dangerous.

The original meaning of cryptography is to make **encryption**. Generally, we talk about "messages", but it means also a file on your computer, or a stream. "Message" is the information you want to hide. So, the encryption takes the message and turns it into something not understandable. The symmetric, **decryption**, takes this and returns the original message. This is made using encryption and decryption functions that are paired. Algorithms are updated, because as time passes new methods are discovered, new technologies. But the **main idea** is the same.In order to make encryption and decryption we use an encryption function and algorithm **E** and a decryption function and algorithm **D.** Then we use an encryption and decryption key, which are just an input to the algorithm. No meaning, just a string of bits. So you have:

$$E_{k1}(m) = C$$

where k1 is the key, m is called *plaintext* and C is called *ciphertext*.

This is what the encryption algorithm produces, the idea is that if the attacker **doesn't have** the key he can't understand the message. If you then use the decryption algorithm with its key you obtain the message.

$$D_{k2}(C) = m$$

In some cases $k_1 = k_2$ and this is symmetric encryption. If $k_1 \neq k_2$ we talk about asymmetric encryption.

Now, we don't like **security by obscurity**. A typical approach is "let's use this encryption algorithm but don't tell anyone". For the attacker is easy to sniff a message. Following an **old approach** is secure to keep all as a secret. That's not good. The only thing that should be secret is the key, the algorithm instead should be open. Because a very secret algorithm could be checked by a very **few people**. When instead an algorithm is open it's easier to check its security and strength because **everyone** can access it. **Modern approach** is: the attacker knows the encryption algorithm, the strength is based on the secrecy of the key. We will use the terms thread and exploit. A **threat** is just something that could be a source of danger, it's not an implementation of an attack, it's just e menace. The **exploit** is a sequence of operations that is well defined, like a program, that leads the attacker to obtain some success, violating some requirements of information security.

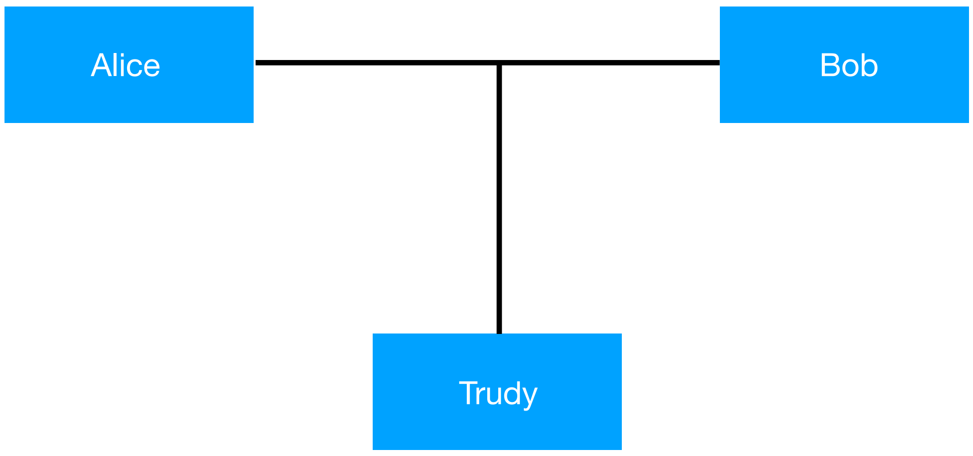
Communication model

This is our model.

For now, we just consider using encryption for privacy. To do this, the model considers:

- The usage of a **reliable communication line**
- A **shared encryption scheme:** E, D, k_1, k_2. So the goal is to send a message **m** confidentially.

Then we have a typical attack model.



This means that the attacker is able to access the communication line. That is **really** so easy, both with cable or wireless connection. We assume that every attacker can easily access the **information** we are transmitting, so we must send **ciphertext**, not the plaintext.

Adversary

There are two types of attacker. The **passive** attacker is the weakest. He can only intercept what we are transmitting. The **active** attacker can instead modify the message being sent and could also send a message claiming to be someone else.

In the **packet sniffing** (passive attack) the intruder Trudy can read all the information that B sends to A and vice versa. In the **IP spoofing** (active attack) it's based on cheating on the identity of the sender. The man in the middle attack could run at different levels. The **denial of service (DoS)** is an attack to the availability of a service, whatever it is. It's realised sending many packets to the attacked host. The SYN packets mean "I want to open a connection". The server then sends a SYN-ACK packet. That's how the protocol works.

If the client is an attacker, when receiving the SYN-ACK packet from server, it doesn't send the ACK and the server keeps waiting for it. If the request are very frequent and all the clients keep the connection half open, the server will use a data structure to keep information about these and when another client sends a request the server could not accept it because its memory for accepting requests is full. This attacks the **availability** of a service.

Security goals

We don't want the attacker to know even **partial information** about the message. For instance, if we are sending the same encrypted message every day at the same time, the attacker could see that the length is always the same. This is information. There are protocols implementing this behaviour: same plaintext, same algorithm, same key, **different ciphertext**. It could be possible that the attacker knows both plaintext and ciphertext, so he could try to **generate** all possible keys. So we want to make this hard, meaning computationally hard to try all possibilities.

If the key has N bits, it takes 2^N to try all possibilities. That's **exponential**. But if N is small enough it's easy to determinate. As time passes, we need **longer** keys because computational power of computers grows. This is very important, the attacker uses brute force attacks, what we want is that no adversary can determine any **meaningful information** about the message m. The attacker could collect small information and then prepare the attack.

Adversary model

The typical model is this: the attacker knows the algorithms E, D, the message space and at least partial information about $E_{k1}(M)$. But he doesn't know k_1 and k_2. Julio Caesar was using the **shift cipher**. In this case, the key is just an integer, the algorithm just shifts the characters, so for example is k is 2, A becomes C, B becomes D and so on…

The **substitution** cipher works like this, you have random permutation of characters. The size of the key space is 26!, it seems large but in every language there is a **statistic distribution** of letters. So it's easy to analyse a ciphertext like this.

Perfect cipher

This is the first formal definition. Given the plaintext space = $\{0,1\}^n$ with D known:

Pr[plaintext = P | ciphertext = C] = Pr[plaintext = P]

This means that the ciphertext does not reveal any information about the plaintext. Probabilities are over the key space and plaintext space. You can use **Bayes theorem** and write this condition like

Pr[C] = Pr[C|P]

Example - One Time Pad

This algorithm has a symmetric scheme. Key K is chosen **at random**.

$E_k(P) = C = P \oplus K$

$D_k(C) = C \oplus K = P \oplus K \oplus K = P$

The cipher is a **perfect cipher**. If the key is equal for everyone, then we could have

$C_1 = P_1 \oplus K$

$C_2 = P_2 \oplus K$

If the attacker discovers these 2 values, then the value $C_1 \oplus C_2 = P_1 \oplus P_2$, so it's easy for the attacker to discover P_1 and P_2. It is a **bad idea** to reuse the same key for different encryptions. So you need a new key for another encryption. The random generators from C or java are **not good** under a cryptographical point of view because they are **predictable**. One time pad is a perfect cipher if we have a **true random key** for every encryption. In this case the xor between P and the key will be really random. If the plaintext is short I need padding. If it's very long I don't need it, but to be really sure I need it anyway. So I don't know the real length of the message.

Theorem of Shannon

This theorem says that a cipher cannot be perfect if the size of its key space is less than the size of its message space. This is proved **by contradiction**.

- Let's assume that #keys (e) < #messages (n) and consider a ciphertext C_0 such that $Pr[C_0] > 0$ (so it exists).
- For some key k, consider $P = D_k(C_0)$. We could have one different P for every different key. We can have max e plaintext.
- Now we choose P_0 such that it is not of the form $D_k(C_0)$.
- In this case $Pr[C_0 | P_0] = 0$
- But in a perfect cipher we have that $Pr[C_0 | P_0] = Pr[C_0] > 0$, that is a contradiction.

So n can't be greater than l. For a perfect cipher, l ≥ n.

Attack models

- **Eavesdropping:** the attacker can get messages exchanged in a private conversation.
- **Known plaintext**: the attacker has some samples of C and P. He has some pairs but doesn't have the key, but the pairs have the same encryption algorithm. For discovering a key of N digits, we need max 2^N tries. This happens in a **brute force** attack. For example, 128 is a very high number. But if a **mathematician** says that there is some math property that can exclude some keys, he would say that the algorithm is **broken**. For us, under a practical point of view, **it's the same**.
- **Chosen plaintext:** in this case, the attacker could know some plaintext to be encrypted and obtains the ciphertext, without knowing the key. Also this could be useful.

- **Adaptive chosen plaintext**: this is the variant where the input of the algorithm is chosen basing on the **previous information**. This is more powerful, but also more difficult.
- **Chosen ciphertext**: the attacker chooses some C and obtains the relative P.

These attacks, chosen plain/ciphertext, are also called “lunch time attack”, because it’s easy to imagine the case where the analyst leaves his room for lunch and the attacker is free to use his tools. The security today is based on **computational power**. What is the threshold? It’s a limit, growing in the time. In the 70’s a keys of 56 bits was looking like a good threshold. **It wasn’t**. 80 is a great number. But google broke even an algorithm with 80 bits key. Today 128 is **secure**, but many people are already using 256 bits keys.

The approach of the **stream cipher** is based on the idea of the one time pad. Is defined a secret key, called **seed**, that is used to generate a byte stream (keystream). The keystream is a string of bits enough long to permit encryption. The function generating the next bytes uses the seed or both the seed and the previous generated bytes. In the **synchronous** case you use only the key, in the **asynchronous** is the other case. Managing a key is like managing a password: **I need another key**. Where I save this? The new key is simpler. Then I store another key, simpler. And simpler. And so on, so in the end I have only a simple key to remember. We’ll see how it works.

The **block cipher** is the logical way to organise operations. In this approach you have an encryptor and the plaintext has a fixed size. Also the ciphertext has a fixed size.

For any given length N: if the keystream is truly random then One Time Pad is a *perfect cipher.*

Spare phishing

Is any of you watching the **Hackmageddon** website? If you want to be up to date it becomes a job and you have to spend 1 hour every day to read specific websites, news, forums and so on. It’s not so easy to be up to date with attacks. What is more important than knowing a specific list of attacks is knowing the **techniques** that have been employed for running attacks. What is still important today is the human factor, never under evaluate this factor. Do you know what a **spare phishing** email is? What is a phishing email?

I'm sure that all of you have been attacked by phishing attends. You know, those emails asking you to **open some url** where you have to put you credentials, credit card number or your email credentials and password because something weird has happened or it's going to happen. Your account is blocked, your money is frozen in the bank and so on. This is **phishing**. It's just an attend to make crime and targeting you and these emails are designed by some attacker sending emails to many many people. How many? I prepare my website for tracking people, I write an email, I send it to **1 million people**, how can I get 1 million addresses? You can buy email addresses. If you go to the **dark web** you will find many black markets and you can buy whatever, you can also buy a killer, other bad services finding in the black markets. You know how to go to the black market, using a browser that is specific, tor browser, anyway the attacker can easily buy for 10$ millions of email addresses and then can send the phishing email. Have you ever written your credentials when receiving these emails? How many stupid people, on 1 thousand you can imagine, is there 1 stupid every 1 thousand people? Let's say 1. Let's say 1 million emails. How many stupids? So you understand that even an attack like that can be successful. This is a **generic phishing**. I called it generic because it's not targeting you or you, it's a generic text and I hope people will follow my request. The **spare phishing** is very dangerous because the text of email is specific for you. So the spare phishing email cannot be sent to 1 million people, not even 10, it's for only one person, after studying what that person is doing yesterday, last week, what he is expecting, getting information about his life, after collecting some information you can send a **target email.** Maybe you can add some expectation for the target and you can easily spoof the sender of email. **Spoofing** an email means cheating about the real sender. You can easily look like the boss, the friend, the wife. So, spare phishing attacks are very hard, even **expert people** can be easy targets, it's not possible for even a cyber security officer to be every time at any time all days checking every email because you should make some analysis on emails. What if you get 100 emails for day? You spend the whole day analysing that? Even experts are target by success by spare phishing emails. This means that in every case where the email you are getting is containing some important information, that is impacting on you life, on you working life, in those cases is needed that you analyse the email, analysing the header of the email. This topic is covered in web security and privacy course. In any case I invite you to be aware about spare phishing attacks and the need to be able to **check the header of emails**. Is not normally shown to people, because you will see just a small part of the header, you will see the name of the sender, the date time, the subject, you won't see the path of the email, you have to open the whole email as a text file and you have to analyse the header part. Clients of even web browsers allowing people to write and read email are not normally set up so that you can easily find the sender. You have to look for the specific command to see the whole source of email. just as an exercise find a way to find the whole content of your email.

Algebra and mathematics

I want to discuss some basic information, some basic concept about algebra and mathematics because modern cryptography is **strongly based** on that. Even if we can agree about the fact that our job will not be to design a new encryption algorithm. Maybe if you like, but it requires **strong mathematical skills**, if you are really interested you have to study the first introduction to cryptography, more mathematics, attend some courses, after that maybe you can become a strong cryptanalyst. Is a very nice field, successful people risk to be reach for the whole life. Normally people are not targeting money in life, they are targeting happiness. What is happiness? A good cipher. Anyway, in order to understand some ingredients we need the basics of mathematics for discussing some part of the encryption algorithms, even when talking about hash functions.

In order to talk about **groups**, we should consider a **set.** This set can be finite of infinite, and over this set is defined a **binary operation**, you see here addition, always remind that when you see addition and symbol of addition it's just a name, you have to define the addition operation in some way you may like, I mean this is no necessarily the traditional addition operations, in many case it will be, but you can define some different addition. In particular, if the properties you see are are satisfied, we are talking about abelian group. What are the properties?

- The **closure**, it means that whatever element of the set you pick, you compute the operation, binary operation between two elements, the result still belongs to the set.
- The operation is **associative**, is very easy.
- The operation is **commutative**, for all elements you may choose in the set. Also the commutative property is very easy. Not all binary operations are commutative.
- The **identity element** must exist. The 0 symbol is just a possible way to denote the identity element, is an element such that if you combine the element with any element of the set by operations you get the element of the set. So if we are talking about the real traditional addition the identity element is 0. If we talk about

multiplication, the identity element is number 1. So it is very important when you are visiting these concepts that what we read here are just symbols. This not means number 0, is a symbol we are using because we use the + symbol in this case. In some cases you can use the dot, reminding the product. In that case you can put the identity element as 1. This can lead to some misunderstanding so I want to point out this.

- The existence of the **negative element**. Denoted by - symbol. The meaning is when you are combining the 2 elements, whatever element of G and its inverse you get as result the identity element. Mathematician like to to all about additive inverse, multiplicative inverse. What is that? Just the same. The inverse. But if you like to denote the operation of the group by +, you have some interest in reminding the addition, in that case mathematicians will be talking about additive inverse.

Groups have so many properties. Some basic definitions.

- **Subgroup**. If G is a group, you can obtain what is called a subgroup by making some subset of G, named H, by taking the **same operations**. Generally speaking, it's not true that taking some elements of a group and the same operations, what you get is still a group. If this happens, you get a **subgroup**. For example, the subset could be not closed under an operation.
- The well known **Lagrange theorem** about group, if G is finite and H is a subgroup of G, then the size of H divides the size of G. This is a very important property, so the size of G is a multiple of the size of H. In this part of mathematical properties like being multiple or prime number, and so on, they are all important properties, very used in modern cryptography.

Congruence

Def: *Two natural numbers a and b are said to be* ***congruent modulo n*** *(n is a positive integer)*

$$a \equiv b \pmod{n}$$

if |a - b| is multiple of n, or, equivalently, the integer divisions of a and n and of b and n yield the ***same remainder****.*

So if the difference between 2 numbers is a multiple of n, they are congruent modulo n. This is a relationship between numbers. It is very important, satisfying reflexive, symmetric and transitive property.

A **quotient set** Z_n is the set of n classes of equivalence, congruent to 0, 1, … , n-1. For example, Z_5 = { [. . . 0 . . .], [. . . 1 . . .], [. . . 2 . . .], [. . . 3 . . .], [. . . 4 . . .] }. All the elements in one class, which is here represented as a [•] subset, are equivalent, accordingly to some definition. In this case we consider the **congruence modulo n** relation. So we have the class of all elements congruent to 0, congruent to 1 and so on. I stop in 4 because 5 belongs to the 0 mod 5 class. What class contains **-1**?

| m - (-1) | = h • 5 **=>** m + 1 = 5 **=>** m = 4.

Most people write this like Z_n = {0, 1, 2, … , n - 1}. This is not a **formal** notation, but it's shorter. Every element should be considered as a **class of equivalence**. You could see a notation like Z/nZ.

Let's see some properties.

- **Invariance over addition**

$a \equiv b$ (mod n) $\Leftrightarrow$ $(a + c) \equiv (b + c)$ (mod n) $\forall a, b, c \in N, \forall n \in N_0$

- **Invariance over multiplication**

$a \equiv b$ (mod n) $\Rightarrow$ $a \bullet c \equiv b \bullet c$ (mod n) $\forall a, b, c \in N, \forall n \in N_0$

- **Invariance over exponentiation**

$a \equiv b$ (mod n) $\Rightarrow$ $a^c \equiv b^c$ (mod n) $\forall a, b, c \in N, \forall n \in N_0$

Let's talk about **order of elements**. Consider a^n as (a + a + … + a) (n times), because remember, that's just a notation, you have to define what it means.

We say that a is of **order** n if $a^n = 0$, and for any m < n, $a^m \neq 0$.

The symbol Z_m* represents the set of natural numbers *mod m* that are **co-prime** to m. This means they are not having common divisor. So the greatest common divisor is just number 1. You can say relative prime.

Then Euler's **totient** function is defined as $\phi(m) = |Z_m*|$.

So, **Euler theorem** says: For all a in Z_m*, $a^{\phi(m)} = 1 \ mod \ n$.

Hence $a^{k\phi(m)+1} = a \mod m, k \geq 0$.

Now, can you calculate $2^{200} \mod 127$?

You can use Euler because this is a very nice number. Is 127 **prime**? Yes. You can prove it checking if is divisible for every number from 2 to $\sqrt{n}$. What is the size of $Z_{127}*$? 126. 2^{126} is equal to 1. You can just by reducing the exponent. 74 is just 37 times 2. Another hint is: What is $2^7 \mod 127$?

In the case you have a group G and you have an element a of that group, of order n, you can consider all such powers, $a^0, a^1, \ldots, a^{n-1}$. This is a called <a> and is a subgroup of G. Element a is called the generator of <a>. If G is generated by a, G is called **cyclic** and a is a **primitive element** of G. There is a theorem saying that for any prime p, Z_p* is cyclic.

Rings and fields

Let's talk about rings and fields. In the case of **ring** we are interested in commutative ring with identity. You have a set of properties here, you have two operations, traditionally denoted by addition and multiplication, if you check the left side you recognise you are just asking that the set F with operation + is a **commutative group**. Now for the second operation you still see closure, associative property, commutative. There is the **identity**, you see 2 different symbols for identity of first and second operation. The last property you want is the unity property giving a cross property for the two operations. It is a **distributive** property, if you want to compute the multiplication between a and the sum of b and c you can evenly distribute the product with respect to the addition. In this case we talk about **commutative rings**. We are requesting property number 9. Now you have ring with additional property. What you see here is all about commutative rings with identity.

Then you have another property, if you add to a commutative ring with identity another property that is the inverse for second operation you get a **field.** You see the definition, a times its inverse its giving the identity, there exists the inverse for whatever element of the set except the identity for the first operation. This is field, modern cryptography is based on **finite fields**.

Z_{15} with addition and multiplication is a ring but not always a field. Z_{15} is not a group to respect of multiplication. Because you loose the closure. And in the case where you consider Z_5 it happens that Z_5 is a group with respect of multiplication.

Let’s look this definition, we are asking that F is a group with respect to the addition, we’re asking that f is a group to respect of multiplication, plus the distributive property.

This is an **equivalent way** to see the definition of field. In practise you want each of operation of commutative group plus the distributive property. Some operation that are important operations while computing the encryption of information can be expressed as operations between polynomials. In order to define this we should make it clear what polynomials we are going to consider. We are considering **polynomials over fields**, you see here the traditional notation for a polynomial, this is polynomial of degree n based on one variable x defined over a field F. All the coefficients here are numbers belonging to the field. The power is the repetition of multiplicative operation just like standard polynomials with operations defined over a field.

The equation you get by computing a polynomial equal 0 has **at most n solutions** in F. We not surprised because this is fundamental, but if you are not working on fields. Considering rings with identity, this means we don’t have the multiplicative inverse, equation 6 times x = 0 in Z_{24} is having **6 solutions**. Even if the degree of this polynomial is 1. You can easily meet strange cases. You can surprise how possible first degree 6 solutions. Because it is **not a field**. It’s not a group with respect to the multiplication. Number 2 is not having a multiplicative inverse.

In the next slides there are some statements, some lines of code. I don’t know whether how to use symbolic you know, there are several tools for computing informative symbolic computation, like Mathlab. I’m not going to discuss these lines, they are just examples for showing the properties.

First definition, if we have 2 polynomials f(x) and g(x) over a field F, the first you see is degree n and the second is m, with m <= n. It is easy to prove that exist **only one polynomial r(x)** whose degree is less than m such that you can express f(x) the polynomial with higher degree like the product of 2 polynomials g(x) and another polynomial h(x) + r(x). This r(x) is unique. This is the statement of the theorem, this polynomial is called the remainder of f(x) mod g(x). Because you can consider it like the result of f(x) divided g(x). Of course the remainder is having the degree smaller of the degree of g(x).

Definition: a Field F is finite if set F is finite.

Galois Fields

$GF(p^k)$

Galois proved that for every prime power p^k, where p is a prime number and k is just a natural number, there exists a unique finite field containing exactly p^k elements. This is **unique**. This filed will be denoted by Galois filed GF of p^k. This means that there are no finite fields in other cardinalities, I may ask you, is there any finite field of cardinality 10? Can it exists a finite field of cardinality 10? Why not? Because 10 is not a power of a prime number. Of course for every prime number you have a finite field of that cardinality but this is just the result stated here. All the powers of 2 are interesting for computer scientist, do you have one Galois field of cardinality 2? Under a pure mathematical point of view when defining a GF of size for example 256 Galois proved that there is one only field. He proved that all fields having that size they are **isomorphic**. It is unique except for possible isomorphic transformation. When talking about polynomials over finite fields you can find some results that are looking different from what we get from more typical situations like the rational numbers. You see here a session on maple, here the operator is asking to compute the **factorisation** of this polynomial here, over the rational numbers.

The third line is very interesting, you can factor this polynomial over the rational and the answer is this one, meaning is irreducible, is like a prime number, you cannot compress it. The same polynomial mod 2 can be factorised. You must be careful with the hypothesis we are considering. We are interested in the power of 2. We can talk about irreducibility if you cannot factorise a polynomial over some finite field, and again you see examples here like asking to compute a factorise of this polynomial mod 5, this is the results, 3 factors, if for the same you see the results, irreducible, notice a very simple detail but we should be aware. The coefficient here you see number 1 and you see that the it means the coefficient is number 0, right, such coefficient are ok because we are computing module 5, the same coefficient are ok because we are computing module 2, I want to say that you can't write here a coefficient 2 in front of x^5. You are competing module 2, remember the coefficient must belong to the field. The polynomial is reducible in Z_5, not in Z_2. To implement an arithmetic polynomial or if you want GF of any possible size here you see, you can user such a procedure based on polynomials.

Let's make an example, let's consider the finite field of 2^5 elements. You want to **realise** this finite field. You can consider this irreducible polynomial and we can consider the set of degree 4 polynomials working on Z_2. So, since we are considering degree 4 polynomials, we are working over Z_2, just bits, this is a possible polynomial degree 4 in particular you see it is degree 3, just a particular case, you can describe this polynomial here as the set of coefficient, you see, 0, 1, 0 means degree 4 term.

A parenthesis on transmission

I want to mention a very interesting aspect, not trivial. In textbooks is not sufficiently covered in my opinion. Let's assume there is a transmission of information, you are the attacker, you get the ciphertext, assume is just one block, you don't know the key, you start trying all possible keys, we already said that there are too many keys but even if the size of the key is small think of the case of the very old **DES** key. The size was 56. You can manage generating all the possible keys. What happens? You have the ciphertext, you know the algorithm, in this case is DES but it can be another algorithm. You try all possible keys, since this is just an algorithm when you give as input a ciphertext and a wrong key you get the output, it should be the plaintext but if the key is wrong you get a wrong plaintext. What is this? **Whatever**. How it looks like? Like **garbage**. So, how can the adversary understand he found the good key? This is a very important question. Is the adversary able to understand he just found **the good key**? If after trying a key here he gets some meaningful text written in whatever language he could say ok I found it. But what if the original plaintext was not an English text? Was if it was just a **sequence of numbers**, coordinates? Not so easy because if the original plaintext was a sequence of numbers, with whatever key you get a sequence of numbers. How to understand if the sequence is correct or not? I want to mention an additional problem the adversary should face. If he doesn't have any information about the **nature**, the characteristics of the plaintext, he find very difficult to understand while trying a brute force he found the right key. You are running a brute force attack, how many keys per second you want to try? Too many, otherwise it will take too much time. So it's impossible the adversary is **inspecting** the outcome for some key. The adversary is running a program that is testing many keys and it's analysing the output. It's not so easy. In some cases only the adversary may be very lucky because instead of testing wether the output is a good plaintext he can just use this output as input for some next block. If the next block works the input was good. This is just a very particular case. In general, brute force attacking requires the adversary to make some effort to understand wether the key just used for running the algorithm was good or not good. Not trivial.

I want to make it a little bit more complicated. For some reason, when Alice is sending information to Bob, Alice may agree with Bob to send a **prefix** before the message. A prefix is a standard pattern. Why? Because that is the pattern that only Alice and Bob know. Bob can have some confirmation. Why? Because Bob will have the same problem that the attacker is having, I mean, suppose that the attacker is running the **man in the middle** attack, who can change the content of the transmission, the message that Alice is sending to Bob can be changed by the attacker. Bob will get a wrong ciphertext, he's knowing the good key, he will use it to decrypt a wrong ciphertext, he's getting garbage. If the original plaintext was original English text bob will be able to understand the information has been changed. But if the original message was just a **sequence of numbers** the fact the man in the middle is changing information how will impact on Bob? He will get a sequence of numbers, it's not so easy for Bob to understand the impact of the attack. You can understand that putting a prefix here so that Bob will decrypt the message and expects to find a prefix here, so he will trust the content of the message. This is just a very simple approach that is somewhat anticipating the meaning of **data integrity**, because we will be studying measures for guarantee data integrity. Alice an Bob are just concerned about data integrity. So there are possible measures, the pattern can be at the beginning, in the tail, in the middle, whatever pattern, so that once the pattern has been agreed the two parts will be able to use it so they can distinguish a good message from a bad one. I close the parenthesis.

Advanced Encryption Standard (AES)

We can now see again the characteristics of the **Rijndael algorithm** (AES). It has a symmetric block size. It can work with 3 possible key lengths. The larger is the size of the key, more secure is the algorithm and more resistant to brute force attacks. The smallest length of 128 it's large enough to be resistant to brute force attacks due to the computational power of today. It requires a **very high number** of attends to find the right key.

So far, AES is resistant to all possible attacks that have been tested. It is very **fast**, because Rijndael is using only operations that are available in every hardware, as a primitive, like shift, xor, change the content of a cell and so on. All quick primitive elementary operations for cpu, very fast, and also the code will be **very compact**. This is also meaning that you can use it also in modern devices, because you understand in the current era in most of these encryption algorithms should run on devices having the big issue of the **battery life**. You need algorithms that are light, not requesting computational effort, otherwise they are not good for mobile devices. The market of mobile devices is increasing, and the one of other devices is just stable. This is an important requirement.

In the 90s when **smart cards** were introduced, people thought they were the future for secure computations. Actually there are more comfortable solutions, smart cards are not the future but still can be useful. The modern view of the designer is not precisely focusing on smart card, but on **small resources**, because smart cards have a very small memory, this means poor resources, computational power and so on. If ok, then you will have a good algorithm running on whatever type of device.

We see some details about Rijndael now. We already said that the **input** of the algorithm is just a sequence of bytes, seen as a matrix like this one, a 4x4 matrix. Every cell of this metric is a byte. How many bytes? 4 times 4. Every byte is viewed as an element of the **GF(2^8)**. We can implement GF by these operations, with polynomials over the GF. This metrics containing at the beginning just the input is called the **state of the algorithm**. Operations of the algorithm are changing the state and after all operations the state will contain the ciphertext. This is just an area of memory where the input is stored, changed on place, then it becomes into the output. There are several key lengths for Rijndael algorithm. You see here the 3 numbers and whatever is the length of the key, the key is considered as organised in a metrics an 4 lines. How many columns? Depend on the size of the key. This is the first byte of the key.

AES operations

Under a **high level** point of view the Rijndael algorithm is taking on **input** the state and the key and you see here the name of the operations it does on them.

The name **key expansion** that is used to make the key bigger than the original size. We need extra bits from the key in order to manage key of small size because they are secure enough. After that the first part of the key is **combined** with the state, running **xor** operations, then there are a number of rounds. Every round is changing the state of the algorithm, according to a fixed schema, after the last round there is a final round and after the final round the state will contain the ciphertext.

So the **schema** of the algorithm it's quite simple, some initialisation, a first operation on the state, a cycle here running several times the same operation, and a final round, this is just a schema showing how you obtain the **input lock.** Then you run a new iteration on this state metrics by taking the output of the previous operation, by taking some more bits from the secret key. This means that as we already commented the secret key needs to be **longer** than the original one. For every step here of this iteration we will need 128 bits. To be combined with 128 bits because there will be xor bit by bit. For every iteration there will be an operation of xoring, the state with the bit of the key, so since the state is having 128 bits we need the same number of bits coming from the key. In the case of the key of this size Rijndael is using **10 rounds**.

People studying the algorithm trying to crack Rijndael also considered the possibility of **weakening** Rijndael by just decreasing the number of rounds. Some attacks can be successful for 1, 2, 3, 4 rounds. But after 6 rounds all known attacks have been unsuccessful. Each round is making the state a little bit **more complicated**, making some high non linear operation on bits, every operation made with the metrics should be **invertible** because otherwise you cant decrypt it. So the function that is used for changing the metrics is in any moment a 1 to 1 function. You need to be able to invert the operation, otherwise you are not able to decrypt.

So, here is now the description of **4 steps**.

- Substitution
- Shift of rows
- Mixing columns
- Xor

The **substitution** is very simple, let just consider 1 byte of the metrics and replace it. Since we are working on a GF of size 2^8 we know that the **multiplicative inverse** is existing, for every element we can consider the multiplicative inverse and just do the replacement. Very often the name of this operation is **S-box**. S-box is just the name of the tool used for implementing the operation. Since the size of the field is small, is not efficient to compute the multiplicative inverse for every case. There are only small numbers multiplicative inverse, you can compute it once, you build a table, look up table, and instead of computing the multiplicative inverse you just pick from the table the good one and replace the original byte with its multiplicative index. 0 is an **exception**, if one byte is 0 there is not replacement. Ok, so the S-box is just the name of the lookup table. S is substitution and box stands for the table. By Managing such a table with size 256 bytes you can implement a very quick replacement for the substitution step.

Let’s see now the other steps, the **round** is the **shift** of rows. Very simple because the first line doesn’t shift, the second line shifts of 1 position, the third 2 positions and the fourth of 3 positions etc. This step is very easily invertible.

Now the step is **mixing columns**, is run by considering every column on the state as a polynomial defined over the GF of size 2^8. It means that the numbers that are in every column are just the coefficient of a polynomial, a polynomial multiplied by a **special polynomial** invertible that is this one. This means you take the first column, consider it like a vector describing the coefficient of a polynomial, this polynomial is multiplied by a standard polynomial here defined so the column is replaced by the bits coming from the representation of the result. This operation is invertible, such that polynomial is admitting an inverse.

This is just the **sequence** of operations, substitution, shift rows, mix columns and the final xoring between the bit coming and the bits obtained by mixing columns. The last step is just a xor. The xor is invertible. About the process of making key longer, you understand if the key is 128 bit you have to run 10 rounds. Every round is requesting to compute xor between the key and the state, so you need 128 bit per round. If rounds are 10 then you will need 1280 bits.

You understand you need to make a **key expansion**. I don’t want to go into details because the definition depends on the length of the key. There is a well known **book** from the two designers of Rijndael, containing all the math needed to implement all details. What is nice, the 2 guys wrote the book after designing the algorithm, it is nice to think they just some reverse engineering of the algorithm just to make confusion on numbers. I don’t know if this is really true.

Is Rijndael **strong**? Very very strong. The **strongest**. Among the ones we use today for symmetric block encryption. Nobody is able to break 5 or 6 rounds. It is considered impossible to break by brute Rijndael, even for the key size 128. Also because don’t forget the detail I told you when you try 1 key you run something to understand if the key is successful or not.

I want to say something more important under a **cultural** point of view. When you will be checking for standards, meany standards are coming from the internet engineering task force, you can read the documents. They are numbered, you can check these documents and some of them are internet standards, other are just proposal, in many cases you will need some standards telling that during some communication you should encrypt information. By symmetric encryption. But the standard is not naming Rijndael. Why not? A very simple answer. When deciding standards, you use concepts and the strength of the standard is in the **concepts**. Then you write another document saying for now this is the list of the welcome algorithms for making an encryption. So that you will change this list from time to time, the standards requiring to use symmetric encryption will be generic and you will need to check for the list of current good one. If you are changing this list, you make just one change in one document, it is obvious that the particular algorithm will change over time because computational power is increasing, and because some cryptanalyst will be able to find some bugs in some encryption algorithms.

Encrypting many blocks

Now we will look at the problem of encrypting something longer than just one block. It's **not so easy**. Symmetric block encryption is encrypting blocks having a fixed size. **16 bytes** in case of Rijndael. This opens a new topic that is the **modes of operation**. This is the term we are using for denoting the possible different implementations of the encryptions of many blocks. Why this is an important problem? Let's just start from the easiest solution. What is it?

I have to encrypt a long file, I just subdivide my file into blocks having a good size and then I separately encrypt every block using the same key. You will get the first block of ciphertext, the second and so on. The final ciphertext will be the **concatenation** of these blocks. If this is the **easiest** approach? Yes, the first thing you can think. What if the original plaintext is having a size **not a multiple** of a size of a block? The last blocks may be padded with some extra bits. Pad by 0, by 1, by something. This is a good solution? No. Why?

The worst problem is that this solution is not **hiding** possible **patterns** that are in the original plaintext. Look at this picture. If you use the simple approach you are not able to really hide patterns. If you think of the requirement we defined when describing the goals of our security, the attacker should not be able to get any **meaningful information**. If you go back to this schema, what happens if block P1 is equal to block P2? It happens that the ciphertext block C1 will be equal to ciphertext C2. The adversary will be able to understand there is a **repetition** on the plaintext. We hate this. This is why this simple approach is never used. And this is just presented and discussed for **teaching purposes**, as a starting point.

Modes of operation

I want to introduce you to the methodology to analyse the performance of a solution, a mode of operation. This is just one mode of operation. It's name is ECB, **Electronic Code Block**. Let's say that ECB is **not good.** But we will be analysing it because we will use the same technology for analysing the next mode of operation. And the ability of the adversary to recognise the possible repetition of patterns is just one of the check we need to do to analyse the performance of a module operation. And we already told that we don't like the fact that ECB is not hiding patterns. What other points can be interesting in analysing a mode of operation? Several points. Looking very interesting.

So, these are two **important points:**

- Patterns?
- Do it in parallel?

Can you do it in parallel? It means, you have many **encryptors**. Can you just make the encryption of many blocks in **parallel**? You can as long you have the encryptors. In most important organisations the encryption is made not by a computer program, by an application, small devices, is **hardware**. More performance with respect to software. If the organisation decides to buy 10 encryptors they can do in parallel some encryption, up to 10. This is interesting because this is increases performances, doing in parallel, but there are 2 sub cases. Can you do in parallel the **encryption** and the **decryption**? Answer in this case is yes. Both cases. Be aware that the encryptor is a device different from the decryptor. The case of Rijndael, one thing is making encryption, another is making decryption. In some cases you have hardware devices able to run only encryption or decryption. You can do in parallel encryption and decryption.

The ability to operate in parallel is a **good ability**. Another point is **preprocessing.**

Can it be useful? What I mean by preprocessing? Let's you are the officer for making encryption and decryption you have at your availability encryptors and decryptors. You also **know the key**. Of course you can't know the plaintext will be sent to you **tomorrow** to make encryption, is there any possible job I can do in advance for making the encryption or decryption **faster**? I don't see any possible preprocessing to make the process of encryption faster. Just we use the encryptor, the key and the plaintext is unknown so we cannot preprocess the plaintext.

Another point that is interesting is **error propagation**. What is an error? The simplest error is 1 bit wrong. So, let's imagine we make some encryption, we get a ciphertext and we have 1 bit wrong. When we are decrypting the ciphertext what is the damage you can image coming from this error? The error here is just impacting on the block where the error **occurred**. What is the impact when making the decryption? Of course it depends on what particular **algorithm** you are using, this is just a general schema, we are not telling what algorithm we are using. The idea is good for every symmetric block of encryption algorithm.

In general we can expect that one bit wrong here will make rubbish on the **corresponding** plaintext. The error is not propagating. Is this good or not? It depends. Why do we have an error? We may have an error for 2 reasons. A **transmission error,** just because this happens. And in this case we like the idea we have a framework that is somehow able to recover from error. Just because ok, you lose some part of information, but then you keep the other information. This is good. What if the error is related to an **attacker**. The attacker changes a bit. What is the purpose of the attacker? Making some sabotage, because the attacker is not able to change bits in a way that the decryption will give another plaintext. The attacker may change 1 or more bit. In this case what happens? In practise is just the same, the effect of the attack is damaging **one block**, not the previous one, not the following. In this case you may consider this property like a **bad property** because it allows the attacker to make a very precise attack, suppose the attacker is having information on the plaintext for other reasons, in this case may choose a particular block because he understands this is the **critical block** containing the information and changes just one bit in the block so it's very hard for Alice and Bob that the cipher has been changed. The effect for the attacker is very good because he was able to damage one precise block.

The analysis of this part here, error propagation depend on your purposes, from other settings of your **environment**, in many cases you can just say ok my information is transmitted over TCP, what happens in case of transmission error for TCP? Retransmission. Why? Why TCP is able to retransmit in case of error? There is a **checksum**, it is able to recognise that checksum is not good. If the checksum is good can we really completely sure that **no error** occurred? No. Because the checksum is short. You may have a collision of checksums. What is the probability of this occurring? The longer is the file, longer is probability, **not so small**. We will be evaluating probability over collisions while studying hashing functions.

Checksum is just a particular **hashing function**. There are other possible attacks. What if the attacker **swaps** 2 blocks? The attacker can remove one block. This strong relationship 1-to-1 between the plaintext, ciphertext can be exploited by the adversary in order to make some changes.

Cipher Block Chaining (CBC)

Let's see a more realistic mode of operation. **Cipher Block Chaining**, CBC. One of the most used. So, consider the drawing here. We have the original message subdivided in equal size of block, 3 blocks. You see that when making the encryption of this block here, the input to the block is not just the block of plaintext, **it is changed** by the previous block ciphertext. Before encrypting block M2 the approach is computing the xor between M2 and C1. The result of previous encryption. Computing this **xor** you have the input for the encryptor and you can get the ciphertext. Every block for ciphertext will be used for the next phase of encryption, participating in the xor between next block and itself. And so on. So, you understand why the name of the approach is block chain, you build a **chain**. Of course we want that all the vertical component of this drawing to be the same because we want to reuse the same approach for every column. This means there is a problem for **first column**, it should be managed like the second you want here the first block of plaintext to be xored with the result of previous encryption, but this is the first one. How to manage this?

Just by introducing what is called **seed**. What is a seed? It is a sequence of bits, the size of 1 block of plaintext, ciphertext. You compute the xor between the first block and the seed. Then you continue. This seed is also known as **initialisation vector**. This is introducing another point when making analysis. Alice is sending the encrypted material to Bob, Bob does need to know the seed? **Yes**. Who decides the seed? Alice. Because they already share a secret key, we don't want they to share **another secret**. Alice is choosing the seed by some approach, maybe a random seed. It is important that Alice transmits the seeds to Bob. And now the point, should the seed by transmitted as **plaintext or encrypted**? This is a good point. It is nice to think about that. Of course, to make decryption you should just invert operations in encryption, I notice that when people just drawing the schema for decryption introduce an error in the schema, look at the position of the xor block. This is the encryption. When you to go back from C2 to M2 you need to decrypt C2. If you decrypt C2 you get what is here. For finding M2 you need to xor again this content, because xoring twice the same content is like the identity. Ok? You take the ciphertext, you decrypt with the same key, then xor the result to the previous cipher text and you get the plaintext.

We can try the **analysis**. Patterns. Do you expect that patterns are revealed by using this schema? 2 blocks are equal, do you expect the ciphertext will be equal? **No**, there is the impact of the previous block. This is hiding schema patterns. This is the worst point of TCP, the fact that is not hiding pattern. CBC is hiding patterns, we like that. Do it in parallel? Can you do **encryption** in parallel? I see people denying. Why not? You need the previous block for the next, you can't do this in parallel. Can you **decrypt** in parallel? Yes. Because when you are computing M3 you need two blocks. C2 and C3, bust you **already have** them. Alice is having the ciphertext, then Alice is sending the whole ciphertext to Bob, he receives the whole file, he's able to use all blocks at the same time if he wants to do that. He can use a pair of blocks for making a parallel decryption. Encryption is totally sequential, decryption can be parallelised. We like that? Yes, because we expect that encryption is made **once**. If information is to be kept confidential, when we receive information we store it encrypted. We want **confidentiality**. Every time you want to open the file you need to make a decryption. This is not a theorem but we expect it will be **higher** the number of times you want to decrypt rather than encrypting.

You can redraw the schema by explicating the role of this blocks used twice. C2 is used twice, once for getting M2 and once for getting M3. You just need **2 blocks** of ciphertext to get one of plaintext. Can you make some **preprocessing**? Can the officer do some job the day before the encryption or decryption when he's not yet known the plaintext but he knows the key? We don't see possibility to get some advantages by spending some time before making encryption or decryption. if you don't know the plaintext you can't start the process.

What about **error propagation**? It is clear that if you get an error here one bit error is impacting on 2 blocks. But the way it is impacting is **different**, one bit error is generating garbage in the corresponding block, one bit error in the next block. In the forth block not affecting. Don't forget the error of one bit is just a **bit flip**. It is possible that if the adversary is having already some information about the plaintext he may decide to change one bit on the plaintext, how to do that? Just by **attacking** the previous block of ciphertext, changing one bit, this is affecting the corresponding block of plaintext but changes just one precise bit on the next block. Theoretically speaking this can be exploited by the adversary. In some cases special while using CBC, information was encoded by using some **special codes** offering special properties, like possibility to recover error in case of 1 bit error, self correcting but it's not so used now, we use another approach.

Don’t forget your goal while analysing the effect of an error. Is an error introduced by the attacker? A transmission error? Just focus on what you are interested in.

What about the **seed**? Can we decide a seed very simple like a **sequence of 0**? What happens? It the seed is just a sequence of 0, when computing the decryption Bob will decrypt C1 and he will get some information here, what you get if you xor your information to 0? You get **no changes**. Means that the first decryption will get the first block of plaintext. Is this a weakness? I would say that we don’t see how the adversary can use this characteristic. Theoretically speaking maybe the adversary can find some way to find the information but the strength of this approach is based on goodness of encryption algorithm and the fact the key is **unknown** from the adversary. If we make the seed secret we are just making things for the two parties. If the algorithm is good and the key is robust you are not getting any further **guarantee**. In many cases the seed is just sent as clear text. If you want to retransmit the same information you should be able to retransmit different ciphertext. What is the difference between using a fixed seed 0 or generating a random seed and sending the seed as **clear text** so the adversary making eavesdropping on the line he will be able to have the seed and the ciphertext? Even if Alice is sending the same file to Bob, with different seed, the adversary will not easy to understand the file is unchanged, is the same transmission. Changing the seed from a fixed seed to a random one, communicated as plaintext, introduces some **more robustness** in the approach. What if you want to **encrypt** the seed? What problems you will have while encrypting? What algorithm you use? The same. What key? The same. What module operation? You don’t need. The size of seed is the size of one block. Just send one block encrypted. Ok but in this case the adversary will recognise same seed. Even if it is encrypted. So we don’t see any strong erasing for encrypting the seed that can be sent as plaintext unless there is further usage of the seed. We are studying just a small ingredients. The encryption of a file, in real cases you will deal with **protocols**, will use many types of encryption, communication, and in some case keys reused for different purposes, in more complex case the seed may have some other usage, it’s up to you analyse.

Smart changes on CBC

Now we use some properties of CBC to make some smart changes. Not very frequently used but it is good to see for several reasons. One reason is a list standard, we said the official standard are important for community. Another reason is that is just an approach aiming at removing the need for padding your plaintext. Why we need to **pad**? Because we have block encryption so that the last block should be padded just because the original size of the plaintext not a multiple of the size of the block. Is possible to avoid such padding so that the final ciphertext will be have the same size of your original plaintext without padding?

This is known as **ciphertext stealing** and consists of using CBC and sending all the blocks constructed by CBC but swapping the last two blocks. Let's now try to understand the meaning of this **swapping** and what properties we have. You see here a schema describing CBC. This is the seed, these are the blocks of plaintext, this is the encryption schema. This is the way you get C1, C2 and so on. Up to now nothing new. What happens if you just think of padding the last block of plaintext by a sequence of **0s**? The last block is made by an original part belonging to the plaintext and an extra part that is just the padding. When we do computation according to CBC what happens in this xor operation? The previous block C_{n-1} is xored to the last block. But now since we have padded the last block what happens? The outcome of the part of original plaintext xored with the first part of the previous block of ciphertext. The second part of the plaintext is the padding, just a sequence of 0s, what happens if you xor a sequence of 0 to something? Something gets **unchanged**. If you consider C_{n-1} as separated in two parts, 1* and 2* and these two parts are obtained where the first part has just the same size of the original block in the plaintext and the second is the **remainder** between the size of plaintext and size of the blocks. The result of this xor will be the one described here.

You will have actually the xor between the remaining part of the plaintext and a small part of the previous ciphertext. The second part is just the same as here, because is **xored** with 0s. If you just make your padding and make your CBC encryption it will happen that when you are sending all the blocks of ciphertext the second part of this block and the second part of this block will be originated by same information. This is introducing some **redundancy** and we want to find a way to skip this repetition of information.

So, it is true that you need to pass as input a whole block, of the correct size, but what you don't need is this part here, because you can reconstruct this part here, the 2* by decrypting Cn. If you have Cn you can decrypt it by **inverting** this encryptor with the correct decryptor. You have this, containing this information that is the same information as here. This is why the propose is send all the blocks, when you arrive to the last 2 blocks send the last and then send one part of the previous. Send Cn, all blocks, skip this, send Cn and send C_{n-1}*. Why? Because by decrypting this one you will be able to reconstruct what is missing here. The size of ciphertext you are sending is **the same** as the original unpadded plaintext.

Conceptually speaking you can imagine making such a padding here but you don't need to adjust the size of the plaintext. You can reconstruct this part of information by decrypting this one. But why I should swap the 2 ones? This way of seeing things is allowing to not implement the padding schema and I would say. This is just an optimisation.

There is an official document you can check for seeing some **details** about the implementation of this schema. In this way the adversary will known the exact size of the original message. This is not really producing a traumatic impact on efficiency, but it is an official standard so it's good you are able to deal with this variant of CBC.

Propagating cipher-block chaining (PCBC)

There is a variant of CBC that was used in **Kerberos**. A system allowing people to get all the services available to the network without reinsert the credential for every service. I want to print, to go to the file sharing, open my email. Kerberos is offering a very **secure environment** for implementing a single sign in on the local network.

They invented a variant of CBC. Actually the variant is having a particular issue. This is the **propagating** CBC. You see here the two schemas for computing the encryption or decryption in this approach. Consider the encryption, you have the seed, you have the plaintext, you see several blocks of plaintext, you have padding. You can see that the encryptor here, several encryptors, they take the input from the outcome of a xor operation. The xor operation is combining a block of plaintext and something that is coming from the previous block. This is very **strange**, this xor is computed between the previous plaintext and the previous ciphertext. So, at the previous step you have a plaintext and a ciphertext, than the xor between plaintext and ciphertext is computed, the outcome is used for xoring it with the next plaintext, then computing the encryption of this outcome. You are **propagating** information because the plaintext is providing something for the next encryption.

When you need to **invert** this process you may use the ciphertext, your input, and you decrypt the ciphertext because you need to invert this process, then to obtain the corresponding plaintext you have to undo the xor here. You just do it again. While encrypting ciphertext you just xor the ciphertext with the original first block plaintext. In this case here in order to get the second block you use the ciphertext and the ciphertext is encrypted and you get the feedback from the **previous phase** where you have computed the xor between ciphertext for undoing the corresponding xor. It's just like CBC with some more connections in the drawing, making components more related between them. What is the **effect**?

Related to the **goal** of initial design of this mode of operation. The mode of operation was designed to **propagate** a single bit error, encryption or decryption. If you have one bit error, in CBC is affecting corresponding block heavily, next block just a bit, then nothing. If you want a single error affecting all sequence blocks, because in case of attack you want to make clear all information is rubbish, an error in this schema is propagating because if you have error in this ciphertext this error is affecting this decryption, this block here, but the one bit error here is combined to the damaged plaintext here, what you have here is something **completely damaged**, this means when you are decrypting the second block of ciphertext you will xor it with random information. You will get the error propagating over all blocks until the last one. This is what the designed of the algorithm was aiming at. Because this is just a schema for propagating a single bit error over all operations. But there is a funny property that makes propagating CBC **unwelcome**.

This is the reason if you take two adiacentes blocks of ciphertext and you swap them, actually the errors introduced by such a swap are not propagated because they are fixing because of similar operations used, I mean two errors because you change two blocks but they are just making **local effects**. Look at the mathematical property, you see a block as the result of two ingredients in the computation of the next block, if you look at 2 blocks far from you the 2 ingredients for computing the result are both in the formula, if you just swap them you are just using a **commutative property** of xor, you have no effect on the other blocks. This means there are possible attacks not propagating over the whole encryption or decryption, this is why this variant of CBC has been **dismissed** while passing from version 4 to 5 of Kerberos.

We'll see 3 more modes of operation. I recommend you when you are studying this part of to try the following, **invent** a mode of operation. What i mean invent? invent. Just make your drawing, plaintext, encryptors and use some feedback connecting some block with some other. What you want. And try to make the analysis point by point. This is the best way for **learning** and making the methodology yours.

Cipher feedback (CFB)

Anyway, this is another variant of CBC, the idea is making CBC a **self synchronising** stream cipher. What does it mean? It denotes the ability to resynchronise after some error. This can be done if you add an additional component in the schema, the **shift register**.

Again you have several blocks of plaintext, you have the seed, the seed is just encrypted, the result of this encryption is xored with the plaintext, you get the ciphertext, then is sent as a **feedback** in the next phase of encryption and it's the input of the next block of encryption. Then the result of encryption is xored with the plaintext. And so on. In this variation you have a different way because actually the encryptors are used for generating what looks like a key stream. Because if you again split the drawing you see plaintext **xored with something**. You get ciphertext. In this point of view this is looking like a **stream cipher**. This submission is not really important because then you have a bit by bit xor. This is an implementation detail. Again you see that the key stream is depending on the previous ciphertext.

This is synchronous or asynchronous? **Asynchronous**, the key stream depends on the previous ciphertext. This is the schema for the decryption, very easy, again you have the blocks containing the encryptors here, they work in order to reconstruct the stream, what is the important here? You see, for making the encryption you use encryptors, for making the decryption you use **encryptors**. You don't need decryptors. Why? Because you need to reconstruct the same key stream.

If you want to **revert** such this, you start from ciphertext, you want to reconstruct the plaintext, you reconstruct this information so that if you have this information here you compute this xor between this information and the ciphertext and you have the plaintext. This information is the output of an encryptor, you need to encrypt again. Again you have the encrypted seed, with same key you get same information. Again, for the next encryption you have to give as input this encryptor the ciphertext and you get the same information. In this way you can compute the xor between ciphertext and the output of encryptors and you get the **original** plaintext.

This is a nice implementation, allowing to build the possibility to choose a mode of operation where you can just use encryptors. What if you want to just use **decryptors**? It would be the same? Answer is yes, it works. The encrypt, decrypt just compute a function f, they both need a key to be run. It is very interesting because it allows the admin of the officer of the secret office to **optimise** the choice of the hardware. What is the effect of one **error** here?

This is encryption, this is decryption. Assume you have an error here in ciphertext, a one bit error, what you get? One bit error in the previous plaintext here and then this is a different input to the encryptor, if you just change one bit the output will be **completely different**. Ok? So, this plaintext here will be completely damaged. But this other plaintext here will be good? Yes, you get it from the previous ciphertext and the current ciphertext, again you are affecting two blocks.

What you need to **decrypt**? You want to get his plaintext, you need this ciphertext and the input. This is the same as CBC, you can do in parallel. Can you do in **parallel** encryption? Let's see. In order to get the block here you need what? A plaintext and the outcome of this encryption. Just the previous block of the ciphertext, you must complete the previous encryption for starting the new encryption, you **can't** do it in parallel. Just like in CBC. If you change the schema for encryption and decryption introducing in the input to the encryptor the **shift register** is having the ability to shift the content. In one direction, and this means that you can use such a content for the input the encryptor then you shift, what happens if you shift? One byte is exiting, one is entering. You may decide some numbers of bytes to shift. After the shift you can recompute. What happens in this way?

You can you have some **redundant** information because the same byte participates in several encryption. You will obtain the same byte, more bytes. What is the advantage? The obvious is that you **recognise** one error. This is the redundant approach. What is interesting is that you don't need to make padding because you have this approach based on encryption after one shift. Shift, encrypt, shift, encrypt and so on. You need to use the **shift register** for making you decryption because you need to replicate the way you are generating blocks.

Output feedback (OFB)

In my opinion this is the prettiest, because again this is changing a block cipher into stream ciphering, if you view at the scheme by the cycle you see your blocks of plaintext xored to something, you may interpret its content going into the xor as the keystream. Now there is some new property, is this stream cipher synchronous or asynchronous? it is **synchronous**. The output of this block is depending on the key and on the seed because this is the seed encrypted, input of the new encrypted and so on.

We always met the asynchronous approach, now it is synchronous. Is this **useful**? Answer is yes. Because in our analysis all the modes of operation we have analysed when we ask if we can do some **preprocessing**, some job in order to make encryption faster, answer was always no. Now answer is yes we can. Because before making this encryption you can compute the key stream, we don't need information about the plaintext, you can compute the keystream in advance the day before.

So you make preprocessing, you compute all this parts here and then when the plaintext is provided to the officer making the encryption the officer just have to compute xor, very fast. Can you do in **parallel**? Yes. If anybody asks you can you compute encryption or OFB encryption in parallel? Yes, but you need preprocessing to do in parallel. Otherwise you can't do it in parallel. Is it clear? You can compute this part of the keystream here after computing this one. You have to make sequential encryptions, you can do in advance, preprocessing, after this you can encrypt in parallel. You have a key store if you want to decrypt you need to reconstruct the same **keystream** , for making decryption you need encryptors again, same used to make encryption. Even in decryption can you have some useful preprocessing? Answer is yes because you can compute the keystream but just knowing the seed and the key.

What is the effect of one bit error? It is affecting one bit on the plaintext. is it **propagating**? No. Just the same effect you have on a pure synchronous stream cipher. OFB is a way to implement what is a good stream cipher starting from a block cipher. You can see it like an algorithm to generate a random key stream. This is offering a good property. Since using OFB we are going back to typical characteristic of stream cipher, we need what property? The property that the adversary is not able to predict the next bits. What if the adversary is using the same block of encryption? By means some like lunch time attack. It is a good practise to start from a **secret seed**, in this case. Because the generation of the keystream is a critical point. This means is a good practise to encrypt the seed if you use OFB.

Counter mode (CTR)

The last mode very similar to OFB. Instead of compute key stream by propagating the bits of encryptor to the next encryption you compute several encryptions here using a **counter**. Is having the initial value of something, +1, +2 and so on. Again you can draw a line here and see it just as a **stream cipher**, synchronous. So you can do preprocessing to compute all the bytes that are needed for computing the xor with the plaintext and again in order to make decryption you have to replicate the computation of keystream so you need again encryptors to decrypt, same you used to generate the keystream. Counter mode. Ok, we have discussed the seed, I conclude this discussion pointing out that while defining the standard of Rijndael, Rijndael is proposed to be used with counter mode and with a variant of CBC that is called **offset.**

Some details about CFB

Here are some details about the mode of operation **CFB**. It looks like a simple variant of CBC because in CBC the xoring is made **after the encryption**, not before. The mode of operation was implemented in many cases by using a **shift register**, that you see on the top of the encryptor in the schema, now what happens here is that you have an encryptor that is expecting an input having a **fixed size**. It should be composed by b bits. So you want to partition the plaintext in b bits blocks. In this case you can partition the plaintext in **smaller size blocks**. I mean, the size small than b. Actually in this picture here you can see there is such a parameter that is s. You can choose s whatever you like, even if there are some typical choices. Your plaintext is **separated** into blocks having size of s bits. So, s is less than b. Assuming you have an encryptor taking as input blocks of 64 bits you can decide you want to separate your blocks in smaller size. This type of approach is providing ciphertext blocks **over s bits**.

The encryptor is providing the output, but from the output only the first s bits are used for the xor operation with the plaintext for obtaining the ciphertext. So, this is encrypting s bits at the time. Also you can imagine that the implementation of this encryption is based on **one** shift register, one encryptor, what you see here is the state of the encryption at the beginning, after the first encryption, after the second and so on, the shift register is just shifting the content by taking the input coming from the previous picture. It is interesting to notice that when dealing **errors** what happens?

Exactly like CBC, when you have an error in one block of ciphertext when making the decryption the error will affect the **corresponding** plaintext block and the next. Two blocks affected, generally speaking 2b bits, where is the size of the block. But if we are introducing the shift register we can deal with **smaller blocks** and this means that you can **recover** from the error after some number of encryptions. Now what happens here is that if you have an error in the last bit when shifting the last bit of this register will be wrong so the input to the encryptor will be **wrong**. How long you will have this wrong bit? It depends on how many times the size s is contained inside b. Because every shift is a shift of s bits. So you just should compute the division, you divide b by s. Imagine you are using s equal to 1. It means that you will have to compute the **integer division** b / 1. So you will have b bits affected by the error. What is the conclusion? Is that you can more finally control the effect of the error. This was very interesting when communication between devices was based on interfaces. There were a lot of errors in **single bits.** This was allowing to control more finally the evolution of the errors and to know exactly when the error was fixed by this property.

The reason why this mode of operation is **not really used** today is just that we deal with errors with some **more modern tools** that we are going to introduce very soon. What is worth to be mentioned is that still true that the most relevant difference between CFB and CBC is that in order to encrypt and decrypt you will use **only encryptors**. In the case of CBC you use also decryptors.

Final part of symmetric encryption

Now we have faced the most relevant topics related to symmetric encryption, the last topic is ok, we are not able to invent new algorithms for making encryptions, can we use the already known algorithms for making **encryption stronger**? There are several approaches, we will see that they are not really different.

The first attempt was a very good idea introduced by Rivest. The name is **key whitening** the idea was originated by the need of making stronger the encryption making by algorithm **DES**. DES is an algorithm since employed since the 70s and its key is having the size of 56 bits. So **brute force attacking** DES is **easy**, especially today is really easy, but in the 80s it was easy enough for governments.

The idea was let’s try to make DES stronger. Divest proposed this variant of DES by introducing what was called **DES-X**. To encrypt a message by DES-X, 3 different keys were employed so not just one key, but **three keys**. Here the keys are noted by k, k1 and k2. You want to encrypt message m? Xor your message with k1. Then you can compute the encryption by DES and the encryption is based on **another key.** Last, take the result of such encryption and again xor such number with another key, a third key, k2. This is the definition.

You also understand you can use such a definition with **other encryptors**, not only DES. We are just the properties of the size of the keys. I don't remember if I clarified that when you are using DES the key size is like 64 bits, not 56. Actually in this key only 56 bits are **independent**, the other bits depend on the 56 previous bits. The other 2 keys here should have the size of 64. So the size of 3 keys is totally **184.** I looks like we want to brute force the encryption we have a key space of 2^{184}. Actually after some years some cryptanalysts studied the problem and discovered some properties. They can be **exploited** for having a space whose number of keys is 2^{119}. It's Looking like we have one key having this length. Anyway this is still a good length. Today it's not really strong but today this algorithm would be classified as **trending to be weak**.

Another approach is **iterate the encryption**. The best known algorithm that was employed for while iterating ciphers is DES, because of its weakness and triple DES was the classical example. How does it work? Actually you have the plaintext, the ciphertext and you see 3 encryptors, there is a **pipeline** so that just use 3 encryptors and use 3 different keys, you get the ciphertext. This looks easy, and apparently it looks like for brute forcing such a type of encryption we need to generate all possible keys and the size of the space is made much bigger because 56 times 3 is still a good result.

Actually by having a **deeper look** we see that this is not really true. Again brute forcing this schema is somewhat **easier** than you can expect. We will see hot to brute force such a schema because it s very interesting, there is a technique that can be applied in several cases, just interesting to notice that when you are using a triple ciphering actually you can use **three encryptors** like you see here in this formula, first, then encrypt, then encrypt. You can also use you see here encryption, decryption and encryption. The two schemas are known like **EEE and EDE**. Just for making a difference on naming. Also what you see here is interesting. **Two keys** only, not 3. Notice that you start using the encryptor with k1, then you decrypt with another key, k2, so it's not really a decryption. Then you use k1 for encrypting again. Why we don't use 3 keys but 2 keys only? This is a very interesting question because the real question is a another. Why we consider **triple encryption** and not double?

We could start by saying ok take this and encrypt twice so that you make DES stronger. Doubling DES or whatever symmetric block cipher is not a good idea because there is an attack whose name is **meet in the middle**. Not to be confused with man in the middle. This is meet, it is interesting showing how by using some reasonable approach, so not needing to be a mathematician, you can run a very interesting and **effective attack. S**uppose that we are using an encryption scheme based on **double DES**. Double DES encrypt and encrypt again. Suppose we are using 2 keys, k1 and k2 so it means that the apparent size of **final key** is 112. If it happens that you **know a pair** plaintext-ciphertext, you can break the encryption. What does it mean know a pair? It means you don't know the keys you know that some plaintext has been encrypted into some ciphertext. You don't need other. You can get this knowledge about this pair in several cases, one case is the chosen ciphertext attack. Another is chosen plaintext attack. Another case is simpler, just you are the attacker and you just start by recording the ciphertext that has been transmitted because **eavesdropping** is very easy, you store such information for some time and after some time the plaintext is made **available** because no longer needing to be secret. So without any attack the attacker can know a pair. Assume that the size of the key is n bits. So if you are using two keys in order to make the brute fore attack you should try 2^{2n} keys. Actually you can **break** the security of this encryption by using 2^{n+1} encryptions and let's see how to do that.

The adversary prepares what is called a **lookup** table and starts computing all possible encryptions, remember that the attacker knows a pair, so he prepares table, starts from the first key and gets the corresponding ciphertext. 2^n. He gets all corresponding ciphertext. How long? 2^n **operations**. If n is too big this is impossible. Now, after building this table the adversary starts a new phase, he runs again 2^n new attempts by computing the decryption of ciphertext C by using all possible keys. Size of this key is again n bits. Try the first key, the second, the third and so on. For every key he computes the **decryption** and checks if the decryption is here in the table. All possible encryptions are written in the table, when the adversary tries all possible keys he tries to guess what is k2. When finding k2 he will decrypt the ciphertext obtaining not the original plaintext, but he will get what is here inside.

When you try the decryption with k2 he will get as result this ciphertext here, so the **complexity** of this algorithm is O(2^n) steps. Then order of O(2^n) steps but every step is requesting a search here. To search in efficient way you need an ordered table. If you want to work on **sorted** table you need to sort it. With 2^n steps you just compute one line, to sort you need an approach to keep the table sorted, you cane use some incremental sorting, just build a table and then sort. You can use any sorting algorithm running on time 2^n log 2^n. This is equal to 2^n times n and this is somewhat requesting an **increase** of this number of steps. So with one try you attack two keys, this is why we **don't use** double DES.

Theoretically speaking this type of attack is **very easy**. Imagine you want to implement as a personal test. You will see it's not so easy to mount up the attack, you need a big storage for that. This is why we use **triple encoding**, now the meet in the middle attack can be used to attack the triple encoding, because in the schema we have described you can just insert another face. Because you prepare the lookup table, start encrypting using all possible keys, but now you have **another step**. The security of triple encoding is security of twice the size of the key. The meet in the middle is providing a tool for **shrinking** the size of the key space by a factor n. I you have the exponent 2n, it will become n and so on.

You can imagine also some variants on CBC for instance. The fact you know the structure of encryption allows the designer to introduce a **feedback** starting from intermediate points. Indeed there are two classical approaches, outside CBC, inside CBC. In this two pictures you see the cases of EDE encryption, using encryptor, decryptor, encryptor. This is **outside CBC,** is just the original one, the **inside CBC** is provided more feedback here, making things somewhat more complicated. This is not actually introducing some new strong properties in CBC, but a schema like that has been welcome in some environments. Most in military environments.

With this **last discussion** I want to close the topic about symmetric encrypting. The topic is very big and you are aware that we have made several algorithms, not many, because after this discussion you know we have Rijndael, very strong, we have DES very weak, triple DES is somewhat more decent of DES but not so strong as Rijndael. We have named a pair of more algorithms but we have not really discussed them. If you go to some table showing a list of encryption algorithms you will see a long list. I want to say that now for most cases Rijndael is strong enough and the **license** for using it is **completely free** for everybody so you can just use RIjndael that is the best, but in some cases you will see that particular **constraints** coming from external specification or from the environment you are working on can push you to choose some other algorithms. While talking about your our **subjects** we will meet some other algorithms but for now it’s not necessary to study other algorithms. Also I want to point out what we told last time, the many **modes of operation** are somewhat transforming symmetric block encryption into **stream ciphering,** because of the way you can interpreter the whole schema allowing the encryption of several blocks.

Last but not least we have learned that **separately encryption** of every block is not a good idea because we are unable to hide possible patterns in the plaintext. What should be natural now, it would be starting talking about **asymmetric encryption**, that type where the key used in encryption is **different** from the one used in decryption. But actually there is one only interesting case, known as **public key encryption**. Before that it is useful to start the study of the techniques for allowing the **data integrity** because public key encryption is very often employed captured to techniques for enforcing the data integrity, so let's just make a suppose while dealing with encryption and start now we start studying what are the main techniques that allow the data integrity.

Data integrity

Main techniques

In order to introduce this new requirement we have to remind in the first class we discussed the meaning of **information security**, do you remember? Is the origin, the source of most of our needs. We are studying cryptography and whatever because of the need of information security. We started discussion saying ok information security contains several ingredients. The first is **confidentiality**, we started studying encryption to guarantee it. The second need is **data integrity,** this is why we now start studying some techniques to assure data integrity, very often described together with authentication, but we should not make the mistake of interpret **authentication** like that process which you write username and password, that is another type of authentication. Let me introduce what is data integrity, after that we will discuss the real meaning of authentication here.

While dealing with the need of data integrity we just forget about confidentiality, because now we deal with **another need**. In many cases you will want both of them, confidentiality and data integrity, it means you have just to use 2 techniques, one for data integrity and one for confidentiality. We don't need to consider confidentiality like a requirement while studying data integrity. So don't get the **wrong message** that when we make data integrity we don't care confidentiality. Just we study a technique for introducing data integrity, then we can sum these techniques with another for confidentiality.

The first goal is to guarantee the integrity of a message, you see here that Alice is sending the usual message to Bob and there is the attacker here that may want to **change the message**. What does it mean? Change the message means adding bit, removing bits, how many? 1 bit, many bits, whatever. Also changing I mean you can change the message without changing the size of the message, just skip some bits, flipping bits you get **another message**. All of these attacks from the adversary are not welcome when we want data integrity. Information security wants in every case data integrity because **non integrity is bad**. In those cases we are not interested in confidentiality, let’s just consider the example where you are downloading an application from the web.

Do you do that? Do you download applications? You want to be sure the app you are installing on your computer **is the good one**. No adversary has changed the application, no one has added to the file some extra bits for installing on your computer some **extras tools**, trojans and whatever. Also when downloading just a file, a form, a standard, you want to be sure you are reading the original document. Data integrity is a basical and critical requirement of security.

Under a philosophical point of view, how you would consider the case where, let me go incrementally, Alice is sending a file to Bob, if the attacker changes **one bit**, he’s violating the data integrity. What if he’s changing **two bits**? The same, of course two bits changed are worst than 1 bit changed. What if he changes **all bits**? The worst case. What it the adversary invents a message? Bob thinks he’s getting a message from Alice but Alice sent no messages. The message is coming **from the adversary**, using **spoofing** techniques, cheating on the identity of the sender. Bob will think he got a message from Alice.

So if we put it in this way, happens that very often we want a **second guarantee,** that the information bob is obtaining is the original information sent by Alice. I mean, if Alice is sending **no information** Bob can get some file, a message, data integrity of this message is exactly the message sent by the adversary, you of course can say that type of data integrity is **useless**, we want to be sure the message was coming from Alice. So, this is why we also talk about **authentication**, the message should be authentic, this is well described in the case where we are considering Alice sending messages to Bob.

Of course you want data integrity in other cases where there is no real message, you **store your file** on your storage and later on you want to read your file from your storage, you want to be sure it’s just the same as you have written some time before. Still data integrity. You want that it is exactly the message you stored. This is why in many cases data integrity and authentication are considered in jointly.

Data integrity

General model

We are introducing a general model basing its structure on both. Integrity and authentication. Basically here are two algorithms, A and V, used by A and Bob. When Alice is sending a message to Bob, she is using algorithm a that is the **authentication algorithm.** This means that Alice is sending a message to Bob an also extra information for ensuring the **authentication** and **data integrity** of the message. To do that, besides using the authentication algorithm at Alice's side, a verification algorithm at Bob's side there is also an **authentication key**. What is this key? Just a secret key. What type? The same type we considered for making **encryption.** This is a shared key, secret key between the two parts. It's shared and secret, no other part is knowing that key. This means that when Alice is sending a message to Bob, Alice is sending a pair of information, you see here the message and the result of application of such authentication algorithm, this **algorithm** take as input message m and using the secret key. You understand this is a very **general schema**, we know nothing about what the algorithms does.

The output of the authentication algorithm is known with many different name, also synonymous. Here you see that the first name is **authentication tag** of message. So Alice is sending a message and authentication tag of the message. Notice that the message is sent as **plaintext** because no need to intersect this topic with confidentiality. When Bob is receiving this pair, he's using this information to run his **verification algorithm**, taking as input both m and the result of the authentication algorithm and also using the shared key. This verification algorithm will output an answer that is **accept or reject**. And you understand the semantic, accept means it is good.

For **n subjects** in a conversation we generally need a **quadratic number** of keys, for every pair. The authentication algorithm is known as message authentication code, giving the acronymous **MAC**. The MAC is the subject producing the authentication tag. Most people use the two words with the same meaning. MAC and authentication tag. One more information, Bob is running his verification algorithm by taking as input the message and the authentication tag but actually the real operation that runs at Bob side is very easy, Bob takes the message, uses the **same** authentication algorithm used by Alice and he's using the same authentication algorithm **on the same inputs**, message and key. The output of this algorithm should be the same of the authentication tag that Alice has sent. So Bob is making this job, starting from m Bob is computing the authentication tag of m and then Bob **compares** what he gets from this computation to what he received from Alice as the authentication tags. The two tags should be **the same**, if the same he will accept, otherwise he will reject. Very easy operation.

Alice is using an algorithm for constructing the authentication tag starting by m. Bob is getting m and uses the same algorithm to **reconstruct** the authentication tag but Alice has also sent the authentication tag together with the message so Bob after precomputing the authentication tag should check if there is a **matching** between what he reconstruct and what he got from Alice.

Now think at the adversary as the **man in the middle**. Alice sends the message and the authentication tag to Bob, but the attacker is intersecting the transmission, now what is the choice of the adversary? **Change** the message? He can try to change the message but he should be able also to change also the authentication tag otherwise Bob will be able to check the difference. The adversary **is not knowing** the key, even if the adversary is knowing the algorithm used by Alice for constructing the authentication tag, the adversary doesn't know the input, the message and the secret key. Is it clear? It is easy enough and you understand this is **very general,** we are not talking about the algorithm.

Of course these functions providing the authentication tag should satisfy some **properties**, the adversary should be not able to reconstruct a **legal pair**. Legal pair means that pair will have a successful result when checked by Bob using the **verification algorithm**. Also it may happen that for a long time the adversary is **spying** the transmission, making eavesdropping, so the adversary is knowing many pairs, message m1, tag t1, and after seeing many pairs based on the same secret key adversary still should be not able to construct a new legal pair.

Another point, the tag should be **short**. What does it mean short? The message is having any possible size. This means that the number of different possible messages is a **huge number**. Very huge. What we can't afford is having a number of possible tags that is the same number as the number of possible messages because it would be a very huge number of tags having any size. We should need tags of every possible size. This is **not comfortable**, not practise, so we want tags having a fixed size, maybe we prefer that the size is not so big because if the sizes is small we can **handle** this extra information in some easy way, without getting interference in other schema for sending information.

As a result of this requirements it happens that the MAC function is **not 1-to-1**. I don't know if you fully understand the meaning of this sentence. Is a statement very important. The MAC function is not 1-to-1. It can't be because we can't manage arbitrary huge MAC calls. We want a fixed size. Whatever is the size we choose the the authentication tag you immediately get the consequence there is no injective function, no 1-to-1 function from message to authentication tag, so there exist two different messages having **the same authentication tag**, just two more of course, many. This is introducing a lot of nice questions and all of you that loved algorithms and data structure should remind the term **collision**. We got collision when dealing with hashing functions. The adversary is knowing the MAC algorithm, maybe he's knowing pairs of messages, the goal of adversary is **finding a legal pair**.

Another point is the following, the goal of the adversary is ok I want to construct a legal pair, message authentication tag. Should the message be **meaningful**? Or i am satisfied if I get reconstruct a meaningless message having a **correct authentication tag**? Think of this. Because it makes a huge difference between the two possible cases. If the adversary want to choose the message and then be able to find the tag this is a very high power for the adversary. In that case he will have the highest power. What if the adversary is **not able to choose** the message but the adversary is able to generate a pair? A very strange message looking like random message and the correct authentication tag for that message. What happens if the adversary sends this message to bob?

Claiming to be Alice, Bob receives a message, the authentication tag, the sender looks like to be Alice, Bob runs the verification algorithms and it says success, accept. Now Bob knows that **only Alice** could produce that pair, because the construction of authentication tag is based on a **secret key**. So if the adversary would be able to manage to send a meaningless message with the correct authentication tag this would be an important success for the adversary, able to trick Bob to think he got a message from Alice but he can't understand the meaning. It's like a stream of bytes, what is the key for **interpretation** such data? This is why we should not under estimate the success in this case. It is still important success, like a **sabotage** coming from the adversary.

Information security doesn't like this type of achievement at the adversary side. So, the goal of the adversary is to choose a message and construct **the correct authentication tag** but it's till an important success for the adversary being able to construct a legal pair, even if the message is completely meaningless. If the adversary manages to construct a meaningless message with the correct authentication tag he's running what is called an **existential forgery**. In the case the adversary is able to choose the message and then construct the correct authentication tag he gets the ability to make what is called the **universal forgery**. Of course it's stronger than existential forgery but we don't want not even existential forgery.

So we don't want the adversary being able to find a legal pair, we can think about **brute forcing**, can the adversary brute force because he doesn't know the key? If the adversary is seeing a transmission and authentication tag, if he's knowing all information expect the key he can try to brute force the key. We don't want that the key is too small. We don't want that type of **visibility**. The key should be **not short.**

Authentication tags

Main approaches

There are mainly **two algorithms** to approach the authentication tag. The first is not used really much. The second one is the most used. We will see first a technique using **CBC** for another purpose, not for encryption, then we will see that we'll get the best performance for introduction of **hashing functions**, now they should have cryptographic quality, we need to understand what it is.

Before that just have a very quick review of CBC, I think CBC is very clear, we have the seed, the plaintext, all the encryptors and all the blocks of ciphertext that are constructed. This happens when we want **confidentiality** of course. So, how to use CBC to the purpose of having not confidentiality but **data integrity**?

Just simulate the need of confidentiality. It means you have a secret key that Alice and Bob are sharing, use this key for making an encryption of the message. You choose what is your favourite encryptor and you just use the message as the **input** and you get the ciphertext. Now, you only consider the **last block** of ciphertext and you say this is my **authentication tag**. It means that Alice is running the encryption at her side, she's using the key, she uses CBC and gets many blocks, she just discards all blocks except the last one. That is the authentication tag. Doing this job Alice will send the message as a **plaintext**, the authentication tag that is the last block of ciphertext she got using CBC. What Bob should be done? Use CBC, get the last ciphertext block and **compare** what he got to what he received from Alice.

Is this **secure**? This has been studied in the 90s. At the end the theorem was speaking in a very theoretical way, if the encryption based on key k is looking like a **pseudorandom** function, satisfying the properties of random functions, then if you use CBC MAC and if the message is having a **fixed length**, not any, then it is easy to prove that the approach is preserving forgery. But there are two requirements getting such an important result. The encryptor should produce what it looks like random bits, it is **easy** to obtain in the case of good encryptors. The fixed length means you have to decide what is the length of the message. The question is why if change the length the adversary can make a forger? This is the **point**.

Why CBC is not secure when you have **variable** length messages? Here you see an example. It looks weird but it's very easy. In order to understand let's check what an adversary could do if variable length message are **allowed.** Suppose the attacker knows two pairs, (m, t), and another pair, (m', t'), can the adversary know that pairs?

Yes, he's sufficient eavesdropping for knowing **legal pairs**. Starting from this the adversary can generate a new message, longer than the previous one and the **authentication tag** of this new message m" will be again t'. There is an existential forgery. Just look the last line, there is the definition of m". M" is obtained by concatenating the original message m with **something,** first we notice there are many concatenations, introducing new operants. What are? M2', mx'. They are just the blocks obtained by m'. It means m' you can subdivide it into blocks having **equal size**, just the size that is requested by the encryptor, he's requiring plaintext subdivided into fixed size blocks. If the blocks of m' are m1', m2', etc. the adversary can construct a **new legal pair**.

This is the message, very similar to the concatenation of m and m'. There is just one **difference**, what? Here, instead of using the first block of m', there is another, then the same is just like m'. So let's try to understand the **effect** of this operation here. The idea is to **prolong** the message here, if you want to compute the authentication tag on concatenation you need this output here, the next xor takes two inputs, Cn and the first block of the second message, now what happens if you use as a first block of second message this block here, m1' xor t? T is known, and t is **exactly** Cn. This means that when using the next phase of CBC this block here will be xored again to t, giving as result m1'. What happens is that if you look at the schema, the input to the next encryptor is m1'. What was the input to the **first** encryptor? Is exactly m1 here. It means that whatever is the content of m the next phase of encryption starts by m1' and this value is **not depending** on m, so the last block of the second phase would be the authentication tag t of m'. If you use this message looking like the concatenation of 2 messages with this variation you will be able to construct a new legal pair (m", t'). This is an **existential forgery**. As long as you have a secret key and a good encryptor this is good for fixed size messages.

What if you want **both** confidentiality and data integrity? You can take the message, composed by some number of blocks, construct CBC using some secret keys k1 and you get the authentication tag. Now, in order to get the confidentiality you construct **new ciphertext** blocks using another key, the one for confidentiality. So, after that Alice can send not the plaintext but the ciphertext obtained by using k2 and the authentication tag constructed by taking as input the original plaintext using another key k1. The questions is why using two different keys? There is some more security in that, if you're not using 2 keys what are you sending to Bob? With one key it means that the 2 processes are just the **same process.** When sending the message you will send all the blocks of ciphertext and send again the last block as authentication tag. This means that whatever adversary can write a ciphertext and just repeat the last block as authentication tag. So we want two different keys in order to have **both properties,** confidentiality and data integrity and authentication.

These are two pairs **known** to the adversary, how can he build a legal pair having a different length that is legal, means that Bob will accept this pair even if it's forgery. So the adversary generates a new message m" concatenating the original message and a small variation of m'. The variation is baed on the **authentication tag** of m. Because actually instead of using m', if m' we write it like m' = m1'|m2'… the small variation is use all blocks of m' **except the first.** So here I use m2', m3' and so on. All the blocks. Here instead of using the first block I use m1' xor t. This is t. Why? Because t is the authentication tag obtained by CBC, this means that the last block of m was inputed to the last encryptor obtaining t. This is CBC used for obtaining t. So we have the xor, the encryptor, the result t. This is producing t while encrypting m. Now, if you consider this new message and you want to build the correct tag you have to **continue** and use this ciphertext here and continue encrypting until the last block.

Now, how was t' obtained? By using CBC encrypting m' using as first block the usual schema using as seed a **string of 0**s. While constructing the tag t' the first block here, encryptor, m1' xor something, the seed was 0. If the seed is 0, m1' xor 0 gives as result m1'. So the first block here of ciphertext is just the encryption of m1'. If the adversary is using this one as first block, here we have m1' xor t xor t. The two xor t get into **null,** this is just m'. Then the process continues. If the input is just m1', this is just the same as the case here, the input was m1'. In this case, the **final** authentication tag is t', exactly the same, whatever we make here the input here is just m1'. The effect of this part is canceled by using this xor here. This process of computing next encryption will conclude in computing the **same authentication tag** we computed here. This is why the pair (m", t') is a legal pair. This is an existential forgery attack.

Hashing functions

Introduction

What is not clear from the slide is that now we will be separating for a short time the problem for **ensuring data integrity** by ensuring **authentication.** For now we stop considering the presence of the key, no secret key shared, and let's see what we can do **without** a key. A hashing function is a function used in several different setting in computer science. They are used for implementing hashing tables, you should remember that a hashing function takes a key as input and the result is used for addressing one cell of an array. This is the original usage in **algorithms and data structure.**

Now we want to reuse the same concept here, the hashing functions is used for computing some properties, we use them to take as input a whole input and get the result, the **authentication code**. Is the adversary able to attack this approach? What is the power of the adversary? The first questions is what does professor **Lenzerini** want outside here?

The first question is are normal hashing functions **good** for our needs? The general setting is, Alice send a pair message and authentication tag, can the adversary just replace the message with another message having the **same** authentication tag?

WPA2 broken

We will able to comment about WPA2 **next**, but we can quickly summarise the main points. The WPA2 approach to Wi-Fi was considered to be the **most secure** for connectivity of Wi-Fi network. Everyone is using WPA, different versions. The problem with the protocol was about the **key management**. We are not ready to talk about this, we skip some technical details. It is possible and this researchers are going to provide a prove of concept, they will present a paper in a conference that will be held in a few days, they will present the techniques for **attacking several implementations** of WPA of different brands in order to prove that is not a problem of implementation, an issue related to the design of the protocol. They can force to reset the key that are going to be used for some transmissions in a way that also the key composed by a sequence of 0 can become visible. This is providing to the attacker the ability to **break the cryptography** and this means that if you consider this protocol for connecting to a wifi network the security of the protocol means that you cannot have any longer the information security while connecting to some host by using this WPA.

What if you go reading about this fact **in the news**? You will not see the most important fact. Ok, the security provided at some level of the stack ISO/OSI is broken. We hope that some day the protocol will be patched and implementation will follow. The **patching** of the protocol is not hard, is easy. The real important question is: can I consider myself secure while using such a connection to a wifi network? Answer is **it depends** on the possible introduction of other security measures to some higher levels of the ISO/OSI. The **public infrastructure** that is something we have not still studied, has been decided for providing a framework providing a very good security, as long as we can use some basic information about digital identity of users. We will be talking **later on** of these certificates. Any of you knows about some generic **digital certificates** that are stating what is your public key. What I want to say is you can consider yourself still secure if your tool, your mobile phone or whatever is not providing issues about the **verification of certificates**, the process verifies the quality and correctness and not revocation of certificates, has been finalised, if yes there is no issue in the digital certificates, not even an attacker with the **man in the middle** attack is able to read your conversation, to change your transmission, you can consider yourself still secure as long as all digital certificates are good. You may notice your browser is saying you that it **can't verify** the digital certificate. With high probability, this is an attack.

So, for connections not using any security protocol the breaking of WPA is **killing the security** of your application. If the security at higher level of the ISO/OSI is still valid, you can consider yourself **still secure**, in particular the public key infrastructure, except the robustness about DoS attack, but there's another story. **Information security** is based on confidentiality, data integrity and other things.

This is just a basic view. Of course the possibility to have some secure services is not pointed about by many media that are just providing some **panic** to people. Is important to notice that the attacker should be **really near** your network, it should be able to connect the network and sniff in an effective way. I suggest to continue to follow this topic, after the conference there will be some news, the community operating in security will be enjoying the sharing of **more information** and will be providing a good feedback.

Data integrity and authentication

Keep on with the argument

We were talking about data integrity and authentication, if you remember we provided some reason why **we are coupling** these two requirements. Indeed in a general model where you can see here the basic definition the authentication algorithm is providing a way to construct this authentication tag, depending on the message, and it is also depending on a **secret key** shared between the two parts.

Now I ask you to think. What happens if the security of the secret key is **broken**? It means that the adversary will be able to construct an authentication tag of any message by using whatever algorithm is used to construct the tag. But is interesting to notice that even if the secret key is no longer secret if you just take one bit of the original message the authentication tag **needs to be changed**. This is so important. The adversary that wants to make an attack should change not only the original message buy even the authentication tag. The **MAC code** should be changed. If the adversary knows the key he will able to generate the good authentication tag. I mean that Bob, while running the verification algorithm, will get the answer **success**, accept this message as authentic.

But I want to focus on the **first part.** We change a bit, we need to change the authentication tag, this is not depending on the knowledge of the adversary about the key. You change a bit, you need to change the tag. I'm focusing on this topic because this helps us to start talking about the usage of hashing functions for **data integrity**, at the beginning we were talking about hashing functions without the use of any key. What you can imagine, what can you get from an hashing function without any key? You can get the service that the hashing functions can generate. If you change one bit of the original text you will need to **reconstruct** the authentication tag. This is the subject we need to talk now, we will see several details. We just assume that the key is **not used** for now, in this case the community talks about **unkeyed hashing functions**. Let's now focus on this, so we can understand the power and the limits of this approach. After that we can introduce the usage of a key while employing an **hashing function**.

Unkeyed hashing functions

So according to the original definition an hashing function is a function that is **mapping** a domain into another domain, but what typically happens is that the domain is large, the **codomain is smaller**. We need that for several reasons, many have been discussed last time, the need of having authentication tag smaller to be easily managed, otherwise the length of the tag becomes **comparable** to the length of the message and we need to change the infrastructure we are using.

We want a function mapping some large domain into a small range. You see here just an example of possible hashing functions. Given a number, you can compute a linear function of this number where a and b are constant and you can execute the mod p operation so that you will be sure the result of such hashing function will be a number **between 0 included and p excluded**. You can finally control the range in this way. The question is: is this a good choice of hashing function? How can I understand what is a good hashing function?

In addition, when we have a function mapping a large domain into a small range what happens? The function **cannot be 1-to-1**, it will be not injective. It means you **cannot invert** the function, for any possible value in the range you may have several values in the domain. Under the terminological point of view we talk about images and counter images. For any possible value in the range we say for any possible images. And the counter images is just a possible element in the domain that is mapped by the function to image. For every image you may have **many counter images**.

When you get this you have what is called a **collision**. Can you avoid collisions? **No**. It is very easy to understand that you have a big domain having N possible elements, our messages. You want to map this domain by an hashing function to a smaller range having n values. Whatever is the case when **N >> n**, you can expect several collisions. Is a collision a useful fact or annoying? **We don't like collisions**. Is this potentially dangerous? Yes.

Imagine that Alice is sending the message to Bob by 2 different transmission. First is sending the hash of the file, then will be sending the file. Assume that the attacker has not been able to intercept the sending of the hash, Bob gets the hash, now if the attacker wants to get some success what he needs? A message having the **same hashing**, same authentication tag. It means there is a **collision** between the message wanted by the attacker and the original message. A collision can be **exploited** by the adversary so he could not change the authentication tag, now he wants to change the message. Find a collision, not so easy.

Now without thinking of any particular hashing function the best you can imagine? Is that collisions here are **distributed on the range** in some uniform way, so for every element in the range you may have about N / n different elements mapped to this item here. In this case you have some **uniform distribution** of the values of the hashing functions. The **worst case** is all the elements of the domain are mapped to one only element in the range. All other elements are not used. This is very stupid. Of course I can also imagine that between these 2 extreme cases there are several intermediate cases. Even in the case of the best hashing function, whatever best means, you will have a **high number of collisions**. N / n is the average number of collisions per element. Notice that between these 2 numbers there may be 7 or 8 order of magnitude of difference. Very often we have domain ranges here like 256 bits.

Here you can have messages containing gigabytes, so you can imagine the average number of collisions is very high. Don't you understand what is the impact of that? For every element you have a huge number of elements in the domain mapped to the same element. **Very huge**. Still we would like that the adversary is not able to find one of them, otherwise he can run a successful attack by just changing the message, without changing the authentication tag. The adversary for changing both message and authentication tag must have **some more power**. This is dramatically important. Forever, unless you want to become a mountain, you will need to remember this property of hashing function, all **modern security is based** on hashing functions, you must be very confident with them.

Collision resistance

What if you want to use an hashing function for building a MAC? Since we have introduced unkeyed hashing functions we will continue talking about them. What if we **introduce the key**? There are several approaches. You can compute the hashing function of your messages for building the authentication tag by keeping into account the key in **several ways:**

- Concatenate the key with the message, and then compute **h(k||m).**
- Use message followed by the key, so **h(m||k).**
- Mixing key, message and key, so compute **h(k||m||k).**

You can combine in different ways key and message to have an hashing function of this type.

For now I want to continue on **focus on unkeyed** hashing functions. We are going to introduce some basic properties to be satisfied when using hashing functions for cryptography.

A very important property is **collision resistance**. There are two definitions.

- A hash function h: D → R is called **weakly collision resistant** if for ∀x ∈ D if it is hard to find x'≠x such that h(x')=h(x). Hard means that the computational effort cannot be tolerated, **not polynomial.**
- A hash function h: D → R is called **strongly collision resistant** if it is hard to find x, x' such that x'≠x but h(x') = h(x). This is a little bit **different.** A hashing functions is strongly collision resistant if it's hard to find a pair, x and x', such they are different and they are providing a collision.

In the first case you are given x, in the second case you are given **nothing.** You just have to find a pair. We use the term **strong and weak** just because being strong implies being weak. Also, ¬weak ⇒ ¬strong. This is equivalent.

So, assume that the hashing functions is **not weak**. It means it's not weakly collision resistant. If so, it means that there is a polynomial time algorithm A, depending on the hashing function h, such that this algorithm, given the hashing function h and some x is able to find x' such that this x' provides a **collision.** Polynomial time. This means the hashing functions is not weakly collision resistant, by definition. If this exists, we can easily construct a polynomial time algorithm B like this, it takes no input and provides as output a pair (x, x') that is a collision. How? Choose x at random, then return x, A having as input x. Choose the first at random and find the second, because it's not weak. It's very easy to see that **being strongly implies being weakly.** This is a typical slide that leads to some misunderstanding.

What about the vice versa? If an hashing function is weakly collision resistant it will be also strongly? We don't know, maybe not. That's why we like strongly. We like this resistance because in the case you have a strongly resistant function it will be **hard** for the adversary to find a collision having the same MAC. If it's easy, the attacker can very easily change the message **without changing the tag.**

Birthday paradox

Now the birthday paradox. I expect everyone here knows the paradox. It is good to remind the property because it is providing a very important information for the security of hashing functions.

The birthday paradox is called to be a **paradox** just because it's showing a property that is somewhat **unexpected**. This is the property, if you take at random 23 people the probability that there are at least 2 of them having the same birthday is **greater than 0.5**. It's easy you prove, if you assume all birthdays have the same probability. How can you do it? Just compute the probability of n people having **different birthdays**, all different. Then you can complement this probability to find this one. This is for 23 people. If you make the computation for 22 people the probability is somewhat less. 23 is a **threshold** for the birthday, you can repeat the calculation when you have a general case, a mapping from a domain D to a range R. It's easy to see that in order to have the probability of having two elements mapped to the same element greater than 0.5 it's sufficient to choose this number of elements random in the set, $1.774\,|R|^{1/2}$. It means that you have D, you choose at random how many elements of D? This number. If you choose random this number of elements the probability of having two of them having **the same image** is 1/2.

There is a very special point that is completely unexpected. You see the number of elements to be chosen. Is unexpected that such a number is **not depending** on the size of D, but just on the size of R. Whatever is the size of D this number is still providing a **bound**. This is a **critical point**. If you just start generating random messages using the same hash function the expected number of generations of messages in order to get the first collision is proportional to the square root of this number, the size of R.

If you have a range whose length is 160 bits there is a very well known hashing function having such size for the output. The functions is known as **SHA-1**. What happens if you have a hashing functions like this? The output is 160 bits. If you consider the problem for the attacker, he can start generating random messages, how many messages before finding the first collision with **probability 0.5**? Proportional to the square root of 2^{160}, that is 2^{80}. You try just a brute force attack, you will find some collision. Since collisions are **more frequent** than you may expect, if you have a hash function having this size like SHA-1 you can expect to find a collision with probability 0.5, that is very good the attacker.

After 2^{80} attends, please remind that this property is **not depending on the quality** of hashing function. If you have a very bad function it will be easier for the attacker to find a collision. If the functions is very good, in any case this is the expected number of attends for the attacker to find a collision, a colliding pair, after this number of attends. 2^{80}. This number was considered to be too big, today it's **not.** This is why the function SHA-1, which we'll study because it's still ver interesting, we learn from the structure or SHA-1 this is why Google published in its forums ok we now start to **sunset SHA-1**. No providing any further information. After some time it turns out they were able to **break the security** of SHA-1 by the birthday attack.

What is the **birthday attack**? The most basic and dangerous attack. Consists in running an algorithm generating random messages and trying to find a collision, there is some reason why the attacker wants to do that, because he known that the probability of finding a collision is proportional to the square root of the number of possible different codes. So be careful, what is the **result** of a birthday attack if successful? The adversary finds a **colliding pair**.

I want to tell you the most frequent **error** I meet when talking to people they are also claiming to be *security experts*. The most frequent error is ok, this is a very important digital document, maybe a contract. Here there is a hashing function, the result of a hashing function on the document. This is the MAC. Two parties sign a contract. Some day the attacker says ok I want to generate a different contract that is more useful to me, I want to change some numbers so I can get some **advantages**. If I find a collision for people making the check wether the contract is correct it will seem that the contract is the good one. If the contract is saying I will pay **100**$, I rewrite the contract that is the same but I want to write **1000**$ then I want to manage things so the new contract is having the **same hash**. How difficult for the adversary? Is this difficult? The most frequent error is ok the adversary can do it easily because he uses the birthday attack. **No**. He can't. Because the birthday attack is useful to find a pair of document, you can't choose documents, you can find a pair by generating random documents. If you want to find another document making collision, no birthday attack, you have to go **brute force**, checking all possible documents in the worst case. This is a very frequent error. Birthday attack is an attack providing a random pair, you can't choose document.

How can I manage this? First I want to be sure you understood in this form you **cannot exploit** the birthday attack to find a collision. Your algorithm will have to check all possible documents. If you have 160 bits the expected number of attends will be 159. One half of the whole space. The smart attacker can do that, this is the **real birthday attack.** This is the copy of the document I want to show to the other part. The good one. This is the bad copy, that is the one I'm changing, so I will get more money. I write two documents, now you can generate N copies of the first document where N is a big number, and N copies of the second document by inserting some **small perturbations** in the document.

For example I insert two extra space, the meaning of the document is not changing. I generate another copy, so if the document is having just 100 words, you can generate the spaces very easily by inserting one extra space or not between any pair and you can generate 2^{100}**different documents** saying the **same content**. If you want to insert two extra spaces or change a comma or replace what is the quote in the so calling English quote or other replacements? You can generate a huge number of variations of this document with no effort. Then, you can generate a huge number of the other document in the same way. Now, take all the documents of the first type and of the second, put together and now start **picking from this huge set**, generated at random, it is easy to find a pair providing a collision, especially in the case where the hash is short. When you find a pair providing a collision, what you find? Actually if you find a colliding pair, you may find document of type 1-1, 1-2, 2-1, 2-2. **One half** of the collision will be **good**. The attacker can use the birthday attack to find a colliding pair with probability greater with 1/2 by running the **brute force attack** picking at random from the whole container. Only one half of the colliding documents will be good for the attacker. You have to prepare the attack in advance generating several variations. This is the **practical birthday attack.**

crackstation.net. In this website there is a huge database of the result of many hash functions that are computed over a very big number of different inputs.

Cryptographic hashing functions

We like hashing functions having an output with a size that is **at least** 256 bits, because in this case the birthday attack will need 2^{128} that is still **too high** for modern computational power.

What we define to be **cryptographic hashing function**? This was just a preliminary discussion. We accept cryptographic hashing functions having some properties, in particular we want hashing functions that are **strongly collision resistant**. Another good property is that the hashing function must be **fast** to be computed. Many times it is used so many times and you can imagine why I want a very efficient very fast function. Imagine you are connecting to a server and your protocol is **https,** secure. Security it's including the **check on integrity**. For every bunch of bits there is an hashing function to be computed. You need to do that **many times.** This is why we want the output to be not too long. It is good that you can prove theorem like the following. This hashing functions is strongly collision resistant **unless P is equal to NP,** in that case is **impossible.** So you understand the impact if someone proves or disproves this property.

That are several words used today as synonymous. **Hash, digest, fingerprint**. They are all the same. Researches started providing hashing function working with typical approaches based on **blocks**. We have in this case a hashing function, taking an input having a **constant size** and an output having constant size. Be careful. When you are looking at a hashing function as a user, there are so many available to you. What you will need is not the original hashing function working on a single block, you need the result of some **construction** useful to make the function working for a document whatever is its length. Anyway, at the beginning researches started to design hashing functions working on some ranges whose size is fixed, like that, n bit input, m bit output.

Under a theoretical point of view there was a nice result obtained by 2 different researches, **Merkle and Damgård.** They were working not together but on the same topic. They proved the same theorem. Look at this picture. Every block is one hashing functions, the **same** hashing function. In the bottom part you see the original document for which you have to compute the digest and the document is subdivided into blocks. You see that all block are **combined** so that every hash block is taking as input a small block of document and the output of the previous hash. You can easily understand you can do that many times. If you have some **small** hash function, with small blocks providing small outputs you can combine this way, so every block is taking 2 inputs, you can do some easy computation, if you say that the range is having m bits and the domain is having n bits, this means that you must provide as input here, how many bits? **N bits.** Of these n bits, m are coming from the previous block. N - m are coming from message block. If you know n and m and you know the size of the message you compute **how many blocks** you need. Every block here will have the size of n - m. Once you know the message size, just divide the size by the size of the blocks and you get how many blocks you need. For the first block you will need some **seed**. The final result will be the hash of the message.

It is well known, the theorem they could prove is the following, if function used for every block, H, is **strongly collision resistant,** the whole construction will be strongly collision resistant. If the original function is weak, the whole construction will be weak. This is the theorem proved by the 2 researchers. This provide a useful tool to implement a hash function able to compute the digest of whatever document, for any possible size of the document.

Here you see the statement that the size of the digest should be at least 160 in order to **prevent birthday attacks**. But 3 years ago Google was able to break the security of a function with a size of 160. So now we now consider secure a hashing function whose size is 256 bits. Before taking the break I want to make some comments with you. Of course you download **contents from the web**. It's very typical. How can I be sure I'm downloading the original document? In many cases you will be find the link to download the file and you will find **checksum**, checksum, C H E C K S U M and they provide it by using some hashing function in some cases they use **md5**. Message digest 5. *Broken*. Too few bits. You can brute force md5, actually there are some bugs that can be exploited. The attacker can have a very good time in breaking md5. If you see a website providing the checksum by md5 you have to say ok you are just playing a game here. Mb5 is more secure, many people still provide checksum on SHA-2, SHA-3 and other hashing functions and **very hard to find a collision**.

Be aware, if you download a file by an **https** based connection, the s will guarantee the data integrity but what it means? It means that data are **not changed during**

the transmission. It means that data you get is the same data that was transmitted by the other network adapter. This is the security you get from https. But if the attacker compromises the server is the matter of the content that can be changed. Do you follow me? It is important to **check the checksum**, but many people forget a small detail, if the attacker **compromises** the server and changes the file he's also able to change the hash. Same difficulty, unless the website has been designed with special techniques.

So it's useless to check the checksum? It's not useless, because you know, when an attack is successful the server is **compromised**, it takes just a few hours to realise that. People understand the server is compromised. If the server is compromised it's possible to **shut down it,** restore a consistent state and then boot again the server. The consequence is that if a server is compromised if you just copy the code with your mouse and make a **google search** for the code, in case of compromised website you will find very few hits because it's a new hash code, no time for **web propagation** of such information. On the contrary, if the file is the good one and the digest is good in time many other websites will mirror the server and that the content will be mirrored by several mirror servers and you will have many copies in the search engine of that hash code. Just have this **first check,** copy the digest, search on google, if you find many hits it means that digest is well known since many days. You can be sure the server has **not been compromised**. If the server is compromised the attacker will change the digest but there will need more time to propagate it through the web, in many cases such migration can be prevented. A very careful user wants to download the installer of libre office, for example, I copy the digest, make a google search, I find many hits, it's well known, I download the copy, I run the same hashing function on this file and I have to find the same digest. If ok, I am **verified** then I can run the installer. Do you know how to run an hashing function? Let's have a break.

Keyed hashing

Now I want to go back to the **keyed hashing**. What happens in this case? We can think about the keyed hashing having in our mind the M-D construction. Why adding a key? Reason for adding a key is easily understood because if the 2 parts Alice and Bob have a secret key then the usage of the secret key allows each one to ensure that was the other part to generate the message. They only know the secret key. How **to keep into account** the secret key? We see here some different schemes. People just reading the slide think this is the description of possible approaches. This is a discussion that makes an **hypothesis** and concluding that this one is not good. Try to understand why the first attempt is not good.

Hash function has been designed to take as input the message, they don't show the possibility to get 2 inputs, if you want also a key you have to **mix the message and the key**, you can do that in so many approaches. I'm going to discuss **2 examples** here.

Let's assume we want to compute the authentication tag in a message by using a secret key and in order to do that we use the M-D construction and we compute the digest of a string obtained by **concatenating key and message**. Let's go back to the schema. It happens that the first block depends on the length on the key. Next blocks will just depend on these message blocks. So, suppose just for easy of discussion that **the first block is just the key**, you get the digest of the message and this is depending on the key. If the adversary wants to add one extra block, if the adversary wants to add an extra block, without knowing the key, can the adversary compute the authentication tag of the same message plus **one extra block**? Just add here another h block and then the 2 inputs, one input is the result of this block, the other is next block. Get the output, he will get the **authentication tag**. The correct one, using the correct key but without knowing the key. You understand the **weakness** of this idea. It's weak because if we use the key as a prefix of the content it is easy to prolong the message using whatever the adversary wants, just add extra blocks here, adding whatever message block and using this output. **Don't need the key** for doing that. Do you agree? So this is a bad idea.

We **discard** this approach, not good. I don't want to present all possible ways to mix up a key with a message, I want to enable you to make **criticism** to these possible approaches. And so on. The list of approach is big. You need to be smart enough. What happens if we use **the key at the end**? This is much better than the previous one, still not completely good. We are aiming at *perfection*. No big problems but there are some small issues.

The problem here comes if it happens that the key at the end this time appears to be the last block, it happens. Keys are not so big in this case but may happen the size of the key is **just the size of the last block**. The last block is taking the key as input. What happens in this case? It happens that if the adversary is able to find a collision by using the hashing functions between two messages he can replace the original message with a colliding message having the same size and then the key is using, let me go back to the drawing. Assume that this is the key. If the adversary manages to find a colliding message so that using the first blocks he gets the **same outcome**, he can replace the message here and will get the same outcome. He doesn’t need to know the key, he can take an original pair, can replace the message, he found the collision by using the **birthday attack**. The authentication tag will be good as well. This is a possible attack, not so easy.

It is considered a **good approach** prepending and appending the key in this way to the message. The first method is useless, the adversary cannot just add extra blocks, he needs to know the key because the last block is a key. In the second case is not sufficient for the adversary to find a collision message and he’s not knowing the key. If the adversary is not knowing the key can try generating messages but **can’t recognise** when he generates a good collision. This is a good approach because the previous attacks can’t be used here. There are other variants you can consider, like using some bits of this, some of this by extracting in the middle, beginning, in some particular positions, you can **invent difficult schemas**, but normally in cryptography the simpler is the better. Lot of issues come from implementation error. If you keep it simple implementation errors will be a little number and **easy to detect**. Now I want to think together with you about birthday attack, again. In the case of **keyed hashing**. So, let’s discard the first and second solution. The third is good enough, imagine we use keyed hashing in this way.

Now I ask, is the birthday attack **still possible**? When the message is being sent to the other part, the sending is composed by the message and by the authentication tag, based on the key of the message. Remember that and that Bob will run the same hashing function using the same key and he expects to find the same result. If the same, success, otherwise reject. Can the adversary use **again** birthday attack in the case of **keyed hashing?**

It still works, but it’s useless. After the adversary finds a collision, x’ and x” let’s say, he can manage by using the birthday, so not too many attempts. Now, what can the adversary say about x’ and x”? Are they consisting of same prefix and same suffix here? But different content. Maybe it’s a good collision, but the birthday attack is no giving any guarantee about the fact that the prefix is the same here and also the suffix. You can’t expect that colliding pairs are satisfying **this format.** If the 2 strings are showing the same pattern, same prefix and suffix, is this prefix equal to the real key? The adversary doesn’t know. He can manage to find 2 colliding strings, showing the same prefix and suffix, still doesn’t know if they are the good one. He can’t **check for success.** This is the real problem.

You understand that the introduction of the key in the hashing for data integrity not only guarantees the **authentication**, but it’s a tool for preventing the usage of birthday attacks.

Some families of hashing functions

Here we have just some names of hashing functions.

The MD family is **completely broken.** It means that it's easy to find collisions. Don't expect any useful service from MD family. Still today experts use as checksum the md5. Also, interesting to say that the SHA family is a family of hashing functions that has beed studied for many years. SHA-0 was just an **experiment**, never really used. After that, SHA-1 became a good **standard.** In many cases the output of SHA-1 is having 160 as size. This is a standard approved by the MIST.

Then other proposals were available, the RIPE, SHA-2 family, until arriving to the SHA-3 family. Today the most used hashing functions are the **SHA-2** family. It's a family because actually people use the SHA 256, or 384, or 512. They are all **samples** of SHA-2 family. That's just the size of the output. Today the smaller size, 2560 is still considered **very secure.** I mean, collision resistant. The idea for computing this SHA functions is very similar to other ideas. The SHA-3 is available since few years, at the beginning there was a bit of **confusion** on how to implement the standard.

SHA-1 basics

What are the basic steps of SHA-1? Despite the fact it has been made obsoleted because of the small size of its output if you use today SHA-1 is still very hard to run a successful attack against it. 2^{80} is still a **really good number**. Normal computational power is useless. In most practical cases SHA-1 is still a good hashing function.

SHA-1 is working well for all length of messages, producing an output of this size and the size of the block is half. The original message is **padded**. There is a padding composed by 1 bit equal to 1 and many bits equal to 0. Then you have the original length in bits, the whole should be a multiple of 512. So you may be able to compute the number of 0 in it. The **digest** consists of 160 bits. These bits are considered like 5 32 bit words, named A, B, C, D, E. At the beginning some constant are used as **constant values**. Then this number changes according to the content of the message. The idea is performing many rounds you see here, **80 rounds**, each round modifies the sequence of the five variables, the stream into the digest. There is a lot of **entropy** in this single round, the next value of ABCD is obtained by computing these strange values here. If you carefully look you will see a first value until here, there is a second value, third value, fourth and fifth. This means the value of D is assigned to E, the value of C is assigned to D. The value of B, shifted on left, is assigned to C. Here there is a very complicated way to determine the next value of A. T is just a counter. F is a function defined in next slide. This is the way SHA-1 is working. The initial values are constant. You **xor** the original input and the result as the last operation and you get the output.

SHA-1 is providing a good uniformity in output. Remember, this is not trivial. Good properties of the hashing functions without saying any words about hashing functions. We said collision resistant, it must be computed in a **fast way**, we can combine several hashing functions, and so on. The real computations done from a single hashing function, we have not discussed it. This is just an example, the functions is just doing strange things.

Last topic, very quick. This researchers, proposed the HMAC. This is a **standard**. It is very well known, the definition of HMAC is here. This is the rule to be used to compute the authentication tag. It looks weird but it's **easy.** This is the message, you want to compute the message digest and luckily the message is contributing to the digest. You have to choose your favourite hashing function. Then, make this computation: Hash this value, obtained using a key xored with a constant. Concatenate with the message, the result of this hash is used **again**, key xored to another constant, concatenate to the output of the previous hash. It's a schema for introducing a key and a hashing function, more complicated than the schema shown in the previous slides. The HMAC can be forged **if and only if** the hash function used is broken.

Public key cryptography

I want to introduce the public key cryptography. It's a complex topic for several reasons. This is **asymmetric cryptography**, a different type of encryption. We will talk about:

- Public key
- Private key

The first is meant to be known to everybody, it's **public**, while the other has to be **secret.** The problem of managing private key in a secure way is introducing several needs that make the whole subject more complex. Indeed the proper way to call all the public key cryptography is making reference to what is called **public key infrastructure**, PKI. It's a set of standards you need to have because you need to perform some frequent operations. Not only just encryption and decryption but a lot of other **requirements.** In order to understand why the topic is so complex we start by introducing the idea.

In the 70s these two researchers, Diffie and Hellman, made the following proposal: Ok, consider the classical old approach to cryptography, you have a **secret key**. What if you split the secret key into **2 parts**? Splitting is not just dividing in subsets, but it means deriving 2 informations from the original key. The names of the 2 keys, K_E and K_D and the idea is: we are talking about the key of Bob, if we split it into 2 keys, K_E and K_D we want to use K_E for **encrypting** messages for Bob. Bob will use the K_D for **decrypting** the message.

The idea is that you should make the K_E **public.** So that everybody will be able to send an encrypted message to Bob, just because the K_E is public. Of course public means everybody can easily know it, maybe Bob is just publishing his public key in the personal webpage, maybe he is attaching the personal public key in every email he's sending. There are several ways to **make known** a public key. This is a type of **asymmetric** cryptography. The key for encryption is public, the key for decryption is another key, here denoted by K_D and this is of course a private key, only Bob will know this key.

This is introducing **a new view** on the problem, if you can implement such a system, everybody can very easily send encrypted messages to Bob, he just has to know its public key. It's so important because if you make a comparison to the old approach, think of the case you want to send **to 10 people** in some host country, you send 10 spies. They need to talk in a **secret way.** They are 10 people, in your opinion establishing one secret key shared between 10 people is a good idea? Only 10 people know the key, they belong to the same team. All spies, working same side. Is it a good idea to make one secret key **symmetric** shared between among all these people? No, of course. You may very easily imagine the scenario where the enemy will capture one of them and will convince with some tools the person to make the **key known.** So all communications between other parties will be no longer confidential. The original approach is, for any pair of spies you choose a secret key. Every person in the team, if 10 people, every person should manage 9 keys in order to talk with other people. You need **a key for every pair.** How many keys over all? It is a **quadratic number**. You understand this is pushing all the participants to the team to have some tools for managing so many teams and every time one more person is made available, participates to the team, *n* new keys will be required and shared between the parties.

In the case we can setup a system like the DH one, the number of different keys is **linear**, just one key for person and in order to send an encrypted message you just need to know the public key. You change the server, the key is always the same. The public key is made known to everybody, the private key is safe and it's stored in a **safe way**, just the same way you store a symmetric key. This is the idea. Why they started imagining such a setting? One reason is already understood, you have a linear number of keys. There is another reason very strong.

Practical usage of public key

Let's call K_P the **public key** of Bob and K_S it's **secret key**. Only Bob knows it. The idea was: take a message M, encrypt the message using the public key. In symbols we write: $K_P\{M\} = C$. This is providing a **ciphertext**. Many others use the notation $K_P(M)$, it's the same. In DH idea, Bob can take the private key to decrypt the ciphertext and get the **plaintext**. So, $K_S\{C\} = M$. Everybody can use the public key, only Bob can use the private key.

Let me now propose a variant of this. What if you decide to **encrypt** a message using the private key? I mean, using the same algorithm, $K_S\{M\} = C$. You get a ciphertext, in this new setting who can do that? **Only Bob.** The owner of the private key. Now you can try to decrypt, because of the idea of DH, they wanted the 2 keys to be exchangeable. If you encrypt by the public key, you decrypt by the private key. But you can do also the contrary. This is a **strong property**. But think of the case you can just a way to implement such a proposal. This means to find the algorithm giving this type of encryption. What happens in this case? You can decrypt the ciphertext using the public key. Do we have any confidentiality? **No privacy.** Is this useful? Yes. Because we can be sure that the sender **is really Bob**.

This is introducing the concept of **non repudiation**. To make it really work we need to fix it, this is just a rough idea. We need to elaborate more. The intuition is correct, in this way all people can easily check that, they can decrypt. If they trust the public key as correct they will be sure that only Bob could made such an encryption. This is the real power of public key cryptography, you can use for **privacy** and non repudiation. Such property will lead to a **stronger** data integrity and authentication.

You can combine such a powerful tool, like a hashing function and public cryptography to make it even stronger. Why **stronger**? Think this simple case. Let's go back to the setting for data integrity. Alice and Bob are sharing a secret key, and they use a very good hashing function. Keyed hashing function. When Alice sends a message to Bob, he will be able to understand the sender is Alice. What if the message is **prepared**? The authentication tag is prepared and a third party is receiving the message? Even if such a party knows the key, there is a problem. He can't understand wether the message was originated by Alice or Bob. We have a problem in understanding who is the originator of the authentication tag. Who was the **originator**? You still have data integrity but in case of conflict you can't make any distinction. If you add to such a hashing function the power of public key cryptography used for **non repudiation** you can make data integrity stronger, you can get the result of Alice and Bob can't repudiate the action of preparing a document. This is a set of tools leading to a digital signature.

Public key cryptography will lead to several new results, including **digital signatures**. They are allowing to guarantee data integrity, authentication and non repudiation. They are stronger than just hashing functions. While trying to find the algorithm having such a strong property of the keys, they found **other results**. They couldn't manage to find the solution for the problem.

Implementation of this system

The solution came just **2 years later**. The solution is named **RSA**, still known, still used. We will be studying RSA. Let me go step by step. ACM is the association for **computing machinery**. Articles are providing very important studied and breakthrough in the current computer science. 2 year later the 3 researchers was able to find the algorithm having such a strong property, and a way to construct the keys having **this property**. They used immediately the idea for providing security in the email, because since the email system is **deeply insecure**, very deeply and people among you will be studying the course of web security and privacy will see why email is deeply insecure, they proposed to use **encryption for privacy** and another property for data integrity and non repudiation, aiming at reaching a very strong result, you know, internet was designed in the 70s, emails were designed in he 70s, web in the 90s. Email is much older than web.

The problem of email they were well known since long time, very easy to cheat about the sender of the email, this was leading to the so called email spoofing. The idea for problem was a solution leading to a public key encryption system, needing **several standards**, while defining the algorithm, the tools and the standards you define what is called the **public key infrastructure**.

One way function

A theoretical notion that can be considered is a **one way function**. A function very easily computed when you provide the argument to the function, you provide the argument and you get the outcome. But if you want to invert the function, even if the function is a one-to-one function satisfying the injective property, it becomes **very hard** under a computational point of view to **get the pre image** starting from he image. This is a generic qualitative notion.

For instance, at the time researchers were considering that problem. Given 2 numbers it's very easy to compute the product, p•q. Given the result, is it easy to **get p and q**? This is a very easy question. I'm asking, is it easy to factor an integer number? Not easy. **No polynomial time algorithm** is known for doing that. This means that we **suspect** that the integer multiplication is a one way function. You may can consider it unexpected, how possible it's not easy to compute the factor of a number? It's hard even in the case where p and q are **prime numbers**. If they are prime number, you can easily compute the multiplication, if you are given integer number and you know such an integer number is a product of 2 prime numbers there's no polynomial time algorithm known. What is the point? We want to use one way functions to **make encryption**. So we need to go back to the original information.

A very typical theoretical question also I may ask you in the exam: can we consider **SHA-2** as a one way function? Answer is **Yes**. Since now we are introducing the concept that one way functions they seem good to compute encryptions, next question is ok if SHA-2 is a good candidate for one way function, can we use it to make encryption? No, because it's not a one-to-one, not injective. If you don't know the key the ciphertext is hard to be inverted to find the plaintext, theoretically speaking you can **brute force**, very hard.

Discrete log

Next step is to consider what is called **Discrete log functions**. What is this function? Consider this case, you have a cyclic group **G** and the generator of this group, **g.** You can generate all the elements with g. Now let y be any element in G. Since g is a generator you can understand that you can write y as g^x, for some proper x. Because g is a generator. By definition you can get **every element** of G computing a proper exponentiation of g. You may have also many different integer numbers x satisfying this quality. So, if you call g^x as y, being x the minimal integer satisfying the equation, such minimal x is called **discrete log** of y to the base g, by definition.

You can easily understand why it's using the word **logarithm** because with standard algebra you have this equation to be solved computing the logarithm for the base g. This means x is equal to log in base g. But we are now in a discrete environment and it's not exactly the same as the local. Conceptually speaking you can find the base of a power, it's **not so easy** to compute such a logarithm. example, in the case you have the multiplicative group of Z_p you can consider, you should remember the **multiplicative inverse** is a group and you can consider the existence of a generator so in this Z_p* you can consider every element you can write such an element in this way. The discrete log problem is, given y, given g, given p you want **to find x**.

Is this hard? Yes, it is. Because we don't know any polynomial time algorithm been able to find the solution of this question. This is somewhat funny because it seems that also the function g^x is a one way function, it's easy to compute g^x. **Not easy to invert it** and compute the discrete log. It seems that also g^x is a good candidate to be a one way function. I want to make it clear one particular point, there is **no one way function proved** to be that. We don't know any one way function, we know many good candidates, but **no formal proof** of the existence of a one way function. If you prove that existence of a one way function **you get the proof that P != NP**.

For a one way function we want **easy** the computation in **one direction**, hard in the other. Here you can see a very small algorithm proposing an implementation of a traditional idea for computing the exponentiation. This algorithm is able to exponentiate in a quick way, independently on the usage of mod p. You can add mod p to this operation. If not, still you can get very quickly a **power of a number** by using this algorithm. I expect you know the algorithm for **fast exponentiation.** It's based on the idea that if you want to compute 2^{100} it's not good to compute 2 by 2 by 2… 100 times. **No good.** You compute 2, 2 times 2 and you get 2^2. Then you compute such last result **squared again** and then you square again. In this way you compute **any powers** having a step that is increasing in an exponential way. So if you just use the stupid algorithm you get an exponential algorithm, if you use such a square algorithm you can quickly compute the power in **linear time** over the size of the exponent. The algorithm can be described in a recursive way, I did that when studying algorithm and data structures.

So, what we are understanding? If you consider a function like this one, given x computing g^x mod p it's easy, notice that here we see **mod p**, if we go back to the algorithm we can use mod b when we are computing the result. If this number gets to be greater than b you can compute just the mod. Given the result g^x mod p, computing x is believed to be hard, this is a good candidate as one way function. Now it should be clear why I'm always using the word candidate, **no formal proof** of one way function are known. Actually if you want to formally define a one way function you need to give a **stronger definition**, the idea is just the idea we have just considered, I think you correctly got the idea, but when you are going to formally define a one way function you need to include the concept that not only there is no polynomial time algorithm computing the inverse of the function, in the worst case, you make worst case analysis, consider the case where you have a function easy to be computed and the inverse function, in the worst case, you can compute in exponential time in the worst case. We want that the computation is **hard** even in the **average case**. That's why the definition of one way function is hard to be formally proved.

The definition, a function transforming **bit streams into bit streams** is on is **one way** if it can be computed by a polynomial time, I remind you that polynomial time algorithm means that in the worst case you get the outcome in a number of steps that is a polynomial function having the size of the input. For every **randomised** polynomial time algorithm A, for every polynomial p(n) and for a large n, the probability that the function f computed on the result of A computed to the result of the function is **equal to the function.** It looks weird but you just read that A is inverting x. The probability that the algorithm is inverting f is **less than 1/p(n)**. For every polynomial. This is a **very strong property**. Very theoretical. Assuming that x is chosen from a distribution.

By using this definition you get the result that the function is hard to be inverted even in the **average case**. You see that the goal of defining a function this way makes it more complex to give a formal definition, just in terms of upper bound. I told you this is an open problem and if you can prove there is an open function then P != NP. Not known the vice versa. If we want to make encryption using one way function we are **not completely satisfied** with a one way function. Why?

How to decrypt? It is hard for the attacker to decrypt because it’s a one way function. But the legitimate receiver of the message **should decrypt** in a quick way, if there is an algorithm for inverting the function the function is not one way. I don’t know if you get the theoretical problem. Indeed, a variant of one way function concept is introduced to **fix this**. While elaborating this theory, Diffie and Hellman got an important result. Such result, is providing an **unexpected property,** look at the setting. There are 2 parties, Alice and Bob. They are **not sharing** any secret information, what they share is the same information known to be public. The idea is, if we let Alice and Bob talk in front of the public and they exchange messages, **plaintext** messages, the public of course is listening, can Alice and Bob determine a secret key but the public that is listening all messages is not able to understand what is such key?

Choose a secret key in public

We want Alice and Bob can choose a number that will be a **secret key** and we want they can do this without the knowledge of any shared information, they can determine such a secret key by public messages. But the public listening to the messages can’t determine what key is going to be fixed. The setting is the following.

Imagine **all people in a room**. Alice and Bob, they keep exchanging messages. There are many people in the room. Alice and Bob use an algorithm, all people know the algorithm, no secret at all. Can Alice and Bob fix a number k to be use as a **secret key**? Alice and Bob will know it, the public will be not.

Actually the 2 parties, they don’t need to exchange many messages. **2 messages** are sufficient. Let’s just see this simple drawing here. You see, Alice side and Bob side. Alice is choosing **a number p**, later we will talk about the properties of such a number. A number g and a number A. Alice will be computing g^a **mod p.** Easily. Just a choice made by Alice, she can choose such numbers. Then she sends a message to Bob containing g, p and A. Bob is getting such message, just choosing a number, b. Since Bob obtained such information, is able to compute g^b **mod p**. Easy. The result is named B. Alice is sending g, p, A. Bob is using g and p and sends back to Alice B. Now, Bob can compute A^b mod p. And Alice can compute B^a **mod p**. What is somewhat surprising is that the result of such a computation is **the same**

number K. The numbers chosen by Alice and Bob are not public. They are needed to compute K.

People listening to the conversation will be just able to know g, p, A, B. To compute A they need either a or b. Ok, let's **attack this protocol**. Given A, g, p, we want to find a. What operation do I need to perform? **Discrete log.**

Now we can see the Diffie-Hellman **key exchange**. The public informations are a prime p, an element g possibly a generator of a Z_p* . In this way, the procedure can continue by letting Alice choosing a number *a* at random, inside the interval 1..p-1. When choosing the number p, we want it to be a **safe prime.** Safe primes satisfy many nice properties. We should also get used to this terminology. Key exchange. What is it? The exchange of some information allowing the 2 parties to set a **secret key.** It's very frequently used for **session keys**.

A session key is, you have to do something between two parties, you just need a key to be used **that time**. You start the conversation, set up a key and use the key for your encryption. When you have finished you can just **drop the key.** It's used to encrypt messages during the transmission. Many protocols need to setup a session key. All the operations performing a session key are known like in the term of key exchange.

Perfect forward secrecy

Diffie-Hellman is **strong** because it's strong the discrete log. Now you may think this result was proposed long time ago, still today DH is used. Since the integers a and b are discarded at the end of the session DH is offering the **perfect forward secrecy**, there's no material. It means if for some reasons an attacker manages to find the key, he will get **no information** about the previous key and the future key to be established. That's the definition. The usage of perfect forward secrecy is encouraged today because it gives **strong robustness**. If the attacker succeeds in getting a secret key, no information to know older or future keys is obtained. You can easy understand why we write the property of perfect forward secrecy.

Ok, actually 2 years ago a group of researchers started studying an extensive manners the property of DH. What happened? They discovered that if the prime number p is **not too big** organisations having a strong computational power, like a government, can **precompute** many many values for discrete log, preparing lookup tables and they found that it can be manageable the attacker in such a case to get success against DH for **small sized keys**. How many bits the key to be exchanged? Is a number, you get the number mod p. Just the same number of bits as p. the possible keys are of course. **Pre-computation** can help the attack. There are some technical tools and notions that can be used for better understanding but we are not still ready. Later on when we will know details we will be able to understand why there are few attacks possible. This website here, in the paper introducing the issue, the fix to this attack is just use **big number**, having 1 thousand of bits at least.

Now we are getting closer to a very typical problem in the internet, because when you have 2 parties in the internet and these are just get to start knowing so they have no information, **no secret key established**, they start a conversation using encryption and other stronger properties for information security. They need to setup a key, to do what is called the **exchange**. So they can define a secret key session. A key for symmetric. Let's try to not make any confusion, such keys set up by DH are used as symmetric keys, the parties just fix the same key.

Also I want to make it clear that the public key is **not making obsolete** the symmetric key cryptography, still today Rijndael is a very strong. In all cases where 2 parties can use AES, that could be a very good choice, so public key cryptography is **not a replacement** of the symmetric key cryptography, just as a support. What is important to understand is that the DH key exchange is going to fix a symmetric key, but the mathematical properties using for setting up the same key **is inspired** to the public key cryptography. We will see better this property while studying.

What is important to say is that the key exchange by DH is what you expect to **man in the middle attack**. If we make the DH key exchange inside a room as I told you before, Bob and Alice can see each other, what if the parties are on the internet and they are **not sure about the identity** of the other part? If there is a man in the middle, he can easily understand that. The man in the middle could send a message to Bob claiming to be Alice. What happens? When the man in the middle sends the message to Bob he will be forwarding g, p. For sure **he won't send A**. Because he will choose another number. He will be able to fix a **secret key** shared between Alice and man in the middle. Allowing the man in the middle to setup another key shared between man in the middle and Bob. Easy. If we are under man in the middle attack, it can get success of fixing a secret key between himself and the 2 parties. The 2 parties will be **unaware**. Instead they fixed a secret key with the man in the middle.

Can we fix it? **Not much** to fix, you need to make **authentication** between the 2 parties. If you can make strong authentication, in presence of this DH will be a good solution for the key exchange and the definition. Without any authentication, the DH is **vulnerable** to the man in the middle attack. Since our first class I introduced the man in the middle attack, it's a very **typical** attack. Especially when you connect to a public wifi. The exchange obtained by DH after a strong authentication is strong. Exchange without strong authentication is **weak**. Even if as of today only governments can have the power to run such an attack, with a relatively small p. So many values of discrete log. Governments and maybe some few players on the internet having **strong computational power**.

I want to mention the fact that the idea of DH can be use basing the approach on whatever group structure. There is some reaches about the possible number solution as the discrete log problem, otherwise it's easy to attack the algorithm. In some cases the functions employed by DH are integrated as the so called **elliptic curve**. They are just an introduction to some special functions making all the mathematics more complex, obtaining the achievement that you can get the same security with **smaller size keys**. Last year I was not able to find the time to do that. Let's see. Today we need to make key exchange in every private network application and in any secure connection based on https and other secure protocols. By definition session keys are going to be dropped at the end of the conversation. Today there are typically 2 important approaches to key exchange, one is DH, the other is based on another.

Birthday attack and HMAC

Any question? I have another question to simulate something. There was a question about the **birthday attack against HMAC**. I ask you to focus on HMAC that is a possible approach for implementing **keyed hashing**. In many cases the hashing function that is employed for the implementation of HMAC is a **Merkle-Damgård** hashing function, a sequence of blocks to implement the hash.

This is the definition of HMAC. If you look you will find some variant of this definition, all variants are very similar. In order to compute a keyed hashing the proposal was compute the hashing using a key xored to a constant, **inner padding**, concatenate to the message. The result is a hash, such hash is concatenated with the result of the xor between a key and a constant. Again compute the hash. Here you see the constant. I also received by email a question, why **that particular choice** of 2 constants? Inner and outer padding. There is no a very important explanation why these 2 constants have been chosen, such constant are maximising the distance between the coding. This is somewhat guaranteeing **unexpected augment in entropy**.

Other point very interesting is, somebody may object hey you you are computing **twice the hashing function**, you have one computation, a second one, is HMAC less efficient because you are computing **twice** a hashing function? The original message is considered once. This is the real computation requiring some effort, if the message is long the MD construction will take **several steps.** Then the result of this hash is a **small digest**. This means that the last invocation of the hash is made on a very small argument. Not the same as the previous one, depending on the length of the message. You can imagine you have just one extra block.

As I said there's **no scientific explanation**, in general maximising the distance is providing two starting values that look very different and **very far** each other, there's no scientific explanation acceptable. When talking about HMAC we said that the build hack is somewhat **inefficient**, if the attacker runs its attack he has the problem of not knowing the key, this means that he **can't check for success**. But if we want to better understand what happens in the case of HMAC let's consider this setting, where you can imagine the attacker can compute many times the HMAC. Theoretical researchers say there's an **oracle**, you can query it asking many requests.

Suppose there's an oracle, this one is **computing the HMAC** using some **fixed key**, unknown, for any message you can ask, when you are asking your query to the oracle you provide the message, the oracle will compute the HMAC of the message by using some fixed key. So, since the output of HMAC is having a fixed number of bits, if such number of bits is n you expect that after $2^{n/2}$ different messages you expect a **collision with probability 1/2**. This is the meaning of the birthday attack. So, we call m_1, m_2, such messages providing collisions. Now you can choose random string let's name it x, and ask the oracle to **compute another HMAC**. You see here, the HMAC is computed over the message m_1 concatenated with x, you get the result p. Until now, we asked the oracle for this number of computations, now there is a very good probability that the authentication tag of message m_2 concatenate to x **is again t**, the same we got here, in particular this happens if the hash function is of the type obtained by iterating the computation over several block according go the MD construction. This is used by SHA1, SHA2, the new SHA3 is not based on MD. If it happens that both the messages m_1 and m_2 they end on some block boundary, over these 2 hypothesis MD construction and messages ending on block boundary, it happens that the authentication tag of m_2 concatenated with x is the same, you found another collision **without asking the oracle** for another query.

This is just a **theoretical attack**, the hypothesis behind this attack is asking many times a query to the oracle. It's important to comment that this number here is an **expected number,** if you run the attack you prepare your function and you make your function running this number of times, after this number of runs you **check for collision**, with probability 1 over 2 you will get a collision and you can continue your attack by the technique described here in this slide. What is important to say is that you may find the collision **before** $2^{n/2}$ **attempts**. When you are implementing your software running the attack you decide in advance how many iterations, that will be the number of iterations you choose, with probability one over two. After that you will check if you got some collision, it is **not efficient** to check over collision **for every attempt.**

Variant on DH approach

Last time we saw DH, what is the purpose? This is the purpose, DH is aiming to enable two parties at **defining a secret number**, a session key. The idea is compute this number, that will be the same number the two parties can agree. question: what if we want to agree a session key **over 3 parties**, not just 2? Can I use the same idea? The goal in this case is enabling 3 parties to choose the same key session. The same key for a session that is common for the 3 parties, this is not so frequent. Typically you have 2 parties. If we want to just theoretically imagine that 3 parties want to talk **using the same session key** can we use a simple variant of this DH approach? Any idea?

Even if maybe I'm not strong in math, I need the slide and I read that the secret is $g^{ab} \mod p$. A and b are chosen by the two parties. I understand nothing on math, but if I look at this formula I may try to say ok if parties are 3 what about trying $g^{abc} \mod p$? Is there any way to **compute a number like that**? If we can compute such a number and the messages exchanged are again public messages not showing a, b, c but just **partial computations** like g^aor something like that? So, the attacker has to compute a **discrete logarithm** for having success. Is it possible? Answer is yes, this is a **good choice**.

You can generalise DH by 3 parties by introducing a component here of the exponent for every part. You have to decide what is the **sequence of messages** because in this case the message is very simple, 2 messages, when you have 3 parties you can imagine several sequences of messages. If you just start by considering A sending $g^a \mod p$ to B, ok? Of course let's assume that g, p have been already chosen, because they are **public constants**. Modern attacks to DH suggest to change every time g and p because for a long time DH was based on the same choice of p and g and this is allowing **pre-computation** of discrete logarithm, but let's just skip this problem. If A is sending this number to B this is Alice, this is Bob, let me call this number A, now Bob can compute $A^b \mod p$. This is g^{ab}. He can send this message to Charlie can compute A^b, the message sent by Bob, to the power of c. This will be giving $g^{abc} \mod p$. So he get **the session key.** All parts needs to know the partial information, Charlie needs to send to Alice $g^c \mod p$ and then Alice needs to send to Bob a new message computing $g^{ca} \mod p$, many messages and you can understand in this way all parties will be **enable to compute the same number**.

You can imagine different schemes for sending messages. This is not the scheme minimising the number of messages but you have a very precise and simple order between he messages, in case of failure of the communication the process just **stops**. In other case, other schema of messaging, in case of failure the parties can get **wrong information**, or you should make the protocol much more complicated by inserting **sequence numbers** and other informations to be able to reconstruct the correct order of messages. This is a nice exercise I suggest you to do, try defining the sequence of messages between the 3 parties and compare **at least two possible choices** for the sequence of messages. Compare under a point of view of failure of one communication, **one failure**.

Another nice analysis is **compare** the messages scheme, the scheme behaviour when attacked by man in the middle. DH is **weak** respect to man in the middle. What happens in the case of 3 parties? **Still weak**, of course, but what is interesting is this. What is the power of the man in the middle? Maybe intercepting the communication between Alice and Charlie? Or he's able to intersect all communications between Alice and Charlie and between Charlie and Bob? Third case, he's able to **intercept all communications**. This is very interesting, in the case of two parties only you remember that the effect of the attack is defining two different session keys, between Alice and the attacker and then between the attacker and Bob. In this 3 different settings what is the power of the attack, how many different session keys are defined? This is a nice exercise to be done.

RSA

We are now mature for starting discussing about RSA. I want to remind you the main idea proposed by DH, that they were aiming at **subdividing the secret key** into 2 pieces, **one public and the other private.** The idea is that you can use both keys in some intermixed way, interchangeable, you can encrypt by a key and decrypt by the other, whatever is the key chosen. You can encrypt by the public key and decrypt by the private key or vice versa. Of course the algorithm for implementing this idea was **not found by DH**, it was found later, one or two years later, the idea is very strong because it contains a very nice intuition, we already commented about the fact that encrypting by a public key is enabling people to send an encrypted message to somebody for the purpose of **confidentiality**, right? Everybody can say I have this public key, whoever can send a message to this guy and encrypting this message by public key. Only the owner of the corresponding private key is able to decrypt.

Encrypting by private key is the base for **non repudiation**, if encryption is made by private by definition only one person can do it, and then check the author of original encryption was exactly the owner of the private key. This is the base for definition of digital signatures. In particular the researchers are trying to focus on the idea of using one way function to implement, easy to be computed in one direction, hard to be done in the other direction. We want a **injective function** because we want to be able to **invert** the encryption.

In order to work on this idea it's nice to consider the multiplicative group $Z_{pq}*$. This is defined as usual **multiplicative group** where p and q are 2 **prime numbers.** They should be chosen as large numbers. You can imagine you have chosen p and q and you are computing the product that is N. Now you can consider this multiplicative group Z_n*. N is not a prime number, you can factor N. N has 2 factors. What is the content of this set? By definition this multiplicative group is containing all integers **co-prime** to N. This is the definition. What it means co-prime? We know if a number is not co-prime to N, if m is not co-prime what happens? Either m = i•p, **multiple** of p, if m is a multiple of p it's **not co-prime**. Or m is a multiple of q. Also in this case it's not co-prime. Other possible cases? No. No other cases.

When it happens that m is a multiple of p, it means that m is not co-prime to p, if m is co-prime to p it's not a multiple of p of course. For not having this possibility m should be **co-prime to p**. For not having this other possibility m should be co-prime to q. So we can say that m is co-prime to N **if and only if is co-prime to p and to q**.

Now we want to know how many elements in $Z_N *$? The name of that cardinality is **Totient function**, denoted by $\phi(\bullet)$, introduced by Euler.

$|Z_N *| = \phi(N) = \phi(p \bullet q)$.

What are the possible candidates belonging to $Z_N *$? All members of $Z_N *$ should belong to this interval, [1, pq + 1]. Let me write **all possible numbers** now.

1, 2, … q

q+1 q+2 … 2q

Can we write p times q in the **last line**? Not because p is external to the interval. So we have **(p-1)q**. All these elements, how many? (p-1) elements. They are **multiple to q**, so they are not co-prime to q. They can't belong to $Z_N *$. If we would have some space.

Let's write again the set, using p. The last line is … (q-1)p. We can't have q•p, this is **external** to the interval. How many elements? They are (q-1) elements. These are all multiple of p, they **can't be co-prime to N**, they are not co-prime to p. We can write these elements not belonging to $Z_N *$.

Remind that (p-1) **doesn't belong** to the set, (q-1) neither. Is it clear why? The table p, 2p, 3p, (q-1)p, how many? Q-1. Q-1 forbidden, and again. Pq -1 - p + 1 - q + 1 =

Pq - q - p + 1 = p(p-1) - (q-1) = (q-1)(p-1)

This means that the totient function $\phi(pq)$ is equal to this number. P and q are 2 prime numbers. q-1, p-1, can they be prime? They are **not prime**. Even or odd? The totient function is **even.**

Now the idea is, ok can we compute the **exponentiation** for encrypting? I mean, can I work on the idea on taking a message m, considering it **like a number**, and for encrypting this number I compute m to some power, I compute a **power** that is m^x. Is this a good idea? A 1-to-1 function? Otherwise it's not good for encryption. If it's **1-to-1**, is it hard to invert? The question is when the exponentiation to some power is 1-to-1, over this multiplicative group? Even x, computed x^e, is this 1-to-1? If you choose e **internal** to this interval, answer is **yes**. You need an additional requirement. The theorem is the following, when you choose a number e inside this interval, between **1 and the totient function**, choosing a number e internal to the interval and co-prime to ϕ, then this is a 1-to-1 function in the multiplicative group Z_N*.

If you choose e co-prime to this number, internal to this interval here, then this is a **1-to-1 function**. Very strong property. By hypothesis the gcd between e and ϕ is 1, this is the meaning of co-prime. Since these 2 numbers are co-prime, then e is having a multiplicative inverse mod ϕ. **The Bezout identity**, do you remember?

< Slides >

This theorem is allowing to develop a **public key crypto system**. In order to do that you can follow the following step, you choose 2 prime numbers, p and q, you denote by N the product, you can choose a **number e** in this range, between 1 and $\phi(N)$, such number is co-prime to $\phi(N)$. Then let **d** be the inverse of e, the public key is the pair (e, N). The private key is the couple (d, N). N is public.

In order **to encrypt a number m** belonging to Z_N, not Z_N*, you compute the number $m^e \mod N$. Notice that we are having here **2 different numbers** for the modules, mod N and mod $\phi(N)$. Here we have $\phi(N)$ because we understood that the exponentiation is **one to one** when it happens something and it is the theorem, we are working in this set here. Now we want to be able to **encrypt every possible number** between 1 and N and we need some rule for computing the number. This is the statement, it says for encrypting compute $m^e \mod N$. For decrypting compute $C^d \mod N$. But what we have already seen, the statement previous, is allowing us to say that it works? Yes, it's allowing but it's **not so obvious**, we need to elaborate a little bit for understanding why this is **decrypting the cipher text** obtained by this type of encryption. Before seeing why, just notice that you are using the same operation for encryption and decryption, just **changing the exponent**. This means you can exchange the 2 exponents, you can swap and you can decide to encrypt by d and decrypt by e. D is the multiplicative inverse of e and vice versa. You can **swap** the 2 exponents.

Why decryption works? Before seeing this, let's be sure that encryption is well understood. You see the **encryption** here, the first step is take the message and map the message to a number. Then compute that power mod N, this is the **ciphertext**. Then you can decrypt by using the other exponent. Why **decryption works**? We have found that by definition d is the multiplicative inverse of e mod phi, right? This is the definition. This is the meaning. This means that there exists an integer, some **number k** here, such that e times d is equal to 1 plus k times phi. A certain number of time. The difference between ed and 1 is a multiple of phi, this is the definition of **congruence.**

Consider m and p, what is m? Is the **message**, the number associated to the message, p is a **prime number** we have chosen to run RSA. What is the relationship between m and p? Since p is a prime number, either m is co-prime to p, or m is a multiple of p, no other possibilities, right? Because p is a prime number. In the case m is co-prime to p, the greatest common divisor is 1, then by the Fermat theorem that is in this case equivalent to the **Euler theorem**, m^{p-1} is congruent to 1 mod p. Starting from this congruence, we can raise both sides to the same power, k(q-1). Then multiply **both sides by m**. We get (p-1)(q-1)q and by the effect of this multiplication we get 1. On the right side we multiply this by we raise number 1 to this power, we get number 1, then multiply by m and we get m. We get this **equivalence**. Such congruence still holds in this case, if it happens that the gcd between m and p is just p you have that both sides are congruent to 0 mod p. M is a multiple of p, if this m here is **congruent to 0** and if having a multiple of p you can write m like j•p and you write again replacing m with jp and still get the same value, this is congruent to 0 again. M is a multiple of p.

In both cases it holds what? Notice that this exponent here is the same as this number here, just the same. It's equal to ed. It happens that $m^e d$ **is congruent to m mod p**. Now you can run again the same reasoning comparing m to number q. You can write similar formulas and in the same reasoning you get the symmetric formula $m^e d$ congruent to m, **this time mod q**. Since p and q are two prime numbers, they are co-prime of course, it happens that if you are considering the module computing by multiplying p and q you can combine together to congruencies to get the more general congruence $m^e d$ mod N where N = pq. This is just what we want, you see here that starting from the message m you have an exponent here that is giving a **number congruent to m**. The last congruence is the proof that the decryption **is working.**

So we are using the property that e and d are the **multiplicative inverse** of each other. By stating these properties and exploiting that they are co-prime we can get the combined congruence, this is the process. This is the **explanation**. This is giving a raise to the RSA algorithm, well known and still today employed in so many settings. It works with different key lengths, of course **the longer the higher security**. We consider this size good today but in the last years there are increasing in computational power, suggesting **RSA longer keys.**

We want that the number encoding the plaintext is no greater than N, and this means we want **avoid a lot of weak cases**, we are going to see these cases. Since all the cases where you have some rules for transforming the message to a number. What if you get as a result of this transformation 1? **Is this good**? You compute 1 to the power of whatever and you get **still 1**. This is not a good encryption, what if you get as a result 0? Again, not a good encryption. This beautiful theoretical construction needs to be refined to avoid some particular cases that could make the construction **weak in some cases.**

The algorithm is **not so fast**, you have to run the exponentiation having the exponent very big, thousand of bits. Even if we know fast algorithm for exponentiation if we compare the typical running time to Rijndael or DES you notice that RSA is significantly slower, very slower. This is a **general property**. We now start discovering that public key encryptors are slower than symmetric key encryptors. Is this important? **Yes, it is**. Think of what happens when you use encryption on your smartphone. You want light, efficient one. You don't want the **consumption of your battery.**

Today we choose e and get d, in the original paper they choose d and get e, **it's the same,** we can exchange those. If you choose d to be equal to this number, can you choose equally an even or odd number or do you have **some constraints?** Can we try d = 150? Could work? Remember that this choice of exponent should be a number belonging to the interval, 1 to totient function, and co-prime to p. This is the proof that this is ϕ, this is the computation of the gcd. How getting 156? Just computing the **remainder** of the integer division between this number and the smaller one. And again you compute the remainder and get this one. And so on. This is the prove that **d and ϕ are co-prime.**

Then by some magic tools it's possible to discover that this number is equal to 1. E = 17 is the inverse of this number here. How can we find the **multiplicative inverse**? Just accept for now the two numbers, d and e, now we can choose some encoding for the letters so you see here a very simple proposal, to encode the blank by a **double zero** and the capital letters from A to Z by **consecutive numbers**. We get something like that, this is the encoding, you see that the message is subdivided in 2 blocks. You can of course compute the every number here raised to the power of d or e and you **encrypt the message** and choose public and private key. Typically we use the symbol e for the public key chosen in the interval and d is the private key obtained by inverting e.

How to compute the multiplicative inverse? We start by the **Bezout identity.**

Au + bv = 1. The theorem is not stating the pair u and v is unique, **there exists**. For instance, a times u is equal to 1 - vb. So this is 1 + pv.

Au = 1 + pv. I read that a•u is 1 + something. If I remember the definition of **congruence** I can compute |au - 1| = pb, this is a multiple of b. By definition a•u is congruent to 1 mod b. This is the definition of congruence. v = $b^{-1} \mod a$. You can express a number having a property of multiplicative inverse of b. Notice that in the case where you have 1 here, the pair here u and v is unique because the multiplicative inverse **is unique**. Can I swap the meaning between a and b? Yes. It's the same. It means that u is the inverse of a mod b and also v is the inverse of b mod a. With the Bezout identity we can write a formula that starting from 2 numbers co-prime you can express a number having the property of the multiplicative inverse.

Euclide algorithm for gcd

And multiplicative inverse

Let's remind the Euclide algorithm for computing the gcd. We can extend such an algorithm, it's interesting to remind the **traditional algorithm**. I guess you all know such an algorithm, just few steps.

It's easy to prove that the gcd(a, b) if you assume **a > b** is = gcd(a - b, b). This is very easy. Imagine that a is much greater than b, you can subtract b many times from a. You can do that in a **fast way**, and obtain gcd (a%b, b). This is making the algorithm faster. In order to make the complexity analysis of the algorithm we should make a **deeper analysis** and I don't want to run into that. You can prove that the time complexity of this algorithm is $\theta(|b|)$. This means the algorithm is fast enough.

We want to use this algorithm for obtaining **more information** and actually we can compute an equality that is converging to the Bezout identity. We can write this one, we compute the next remainder by computing the remainder between these 2 items here. You start by r-2 = x, r-1 = y, **you assume x > y.** You start by computing the remainder of the **integer division of x by y** and then you continue by computing the remainder of integer division between y and remainder you got and so on. You can write this iterative relationship, when you are repeating the same steps. When you find that **you get remainder 0** you have found the gcd that is the previous one. This is just the mathematical reformulation of the Euripidean algorithm. If we write Bezout identity for x and y, if they are co-prime and according to what we have already discussed, u is the multiplicative inverse of x mod y, v is the multiplicative inverse of y mod x.

The idea for computing the multiplicative inverse is **extending the algorithm** for maintaining such an equality, you compute triples, many triples. Having this equality maintained. If you can manage to have at some time rn equal to 1 the current value of un and vn **will satisfy the Bezout identity**. The idea is modify the original Euclide algorithm for maintaining this equality, you have to understand how to maintain correct values for un, rn, vn, starting according to the original formulation, you see here. You start by n = -2, then -1 and so on. You want to **maintain this equality.** Is it possible to maintain for many values of n?

You can prove that by **induction**, in a very easy way. In order to make the proof by induction you have to show how to start, the base of the induction. If you choose for n = -2, u = 1, you get 1 times x, and you choose v = 0 and you get here just x. Is it r-2, accords to the formulation of the traditional euclidean algorithm r-2 is equal to x, the greater between the 2 numbers. If you choose u = 1 v = 0 we get x = x here. In the case n = -2. This is a good start. If we choose u - 1 to be equal to 0 we get 0 times x = 0 plus we choose v - 1 = 1, 1 times y you get y, what is r-1? Is y. So the **beginning is easy**, I mean, it's easy to prove that this equality is holding if you start by these values. The notation is the **traditional one**, when you are computing remainders you divide some number by another number, you take the floor of the result, you may need qn, floor, you get the integer number. If this is the position of the notation we are using such qn multiplied by r(n-1) is almost the original number, almost, the difference is just **the remainder of the integer division**. This is just the definition of integer division. This is the **base** of the induction. The equality is holding at the base of induction for n = -2 and n = -1. We need 2 initials steps, at the beginning you have 2 numbers. Now, here you see 2 rules for **updating** the number. This is one rule, you get un from the previous un - 2, un - 1, in this way. Un - 1 times qn, this is subtracted to u(n-2). This is the rule for updating the number. This is just the 2 rules for taking the next values.

Now by the **inductive hypothesis** this is true. We are making an induction. You first prove the base of the induction, then you make the inductive hypothesis and say ok the thesis is holding **up to some number**, using this as hypothesis you show that for the next number the property is still true, this is the original schema. Assume by inductive hypothesis that the equality is holding for (n-1).

Instance of RSA

Some steps are described here. Alice can start constructing the **instance** of RSA by taking 2 large primes, p and q, is it easy to compute the product? Yes, it is. Because the traditional algorithm for computing a multiplication is requiring a time that is squared over the number of digits of the arguments. It is **easy**. Then Alice chooses at random a number co-prime to $\phi(N)$, smaller of it of course. Then computes the multiplicative inverse of such number mod $\phi(N)$. This makes possible to define a **public and private key**. How we can compute the multiplicative inverse? Let's check the algorithm. 2 numbers are co-prime, you need to know the $\phi(N)$. If the adversary is able to know p and q he can compute the multiplicative inverse, this is important to understand. If the adversary manages to find p and q, **starting from the public key** he can compute the multiplicative inverse of the exponent and manages to find the private key. Knowing p and q enables the adversary to compute the private key from the public key, he can run the same algorithm. He should not be aware of p and q.

When you are making the encryption you use N, so what is important to say is that given N can the adversary find p and q such that pq = N? **Factoring** a number. This is belied to be hard, there is no polynomial time algorithm to do such a computation, not even in the average case. If you find such an algorithm you break RSA, there is **no theoretical result**, it's just hard to factor the number. The multiplication is believed to be a one way function. You can easily multiply, but finding the original terms it looks hard, it's a good candidate for a **one way function**.

So, p and q are **sensitive information**, Alice should store them in a safe place, and same for the totient function and $\phi(N)$. Talking about a possible implementation for RSA we can start from some steps. We want such numbers having **thousands of digits**. To have a strong RSA we have a very long key. When you are implementing RSA you have to take care that you are generating p and q **every time**. After choosing e you can use the Euclide algorithm, extended, it looks we are ready but let's look more into details. Find 2 random primes, how to find them? 2 big random primes? Is it easy to find a prime number? Not so easy. If you try the most stupid algorithm you can imagine, it happens that you can be more lucky than what you can expect. This is the **typical approach**. Let's generate a huge number of bits, many bits. The last bit is 1. This is an odd integer. Test if the integer is a prime number, if not restart. Is it **convergent?**

This is a probabilistic algorithm, but you can imagine you never generate a prime number, you could converge after a few steps, there is a very **strong property**, mathematicians know a very important theorem, the **prime number theorem.** If you consider this interval from n to 2n inside this interval there are many primes. How many? More than n / ln(n). This is much bigger than the other one. In any interval you can expect to find **many prime numbers.** This means you can quickly generate random numbers just you need that they are odd, you can expect to find a prime number every log n attends. **Expected number.** If you want a prime number of 4 thousands bits again the number of attempts it's not so high, you compute the logarithm and you are in it.

Still today this is a good approach. Why no other? You can imagine you could precompute some tables of prime numbers and choose at random in these tables, since we are considering prime numbers having **thousands of bits** the number of different prime numbers is huge. You can't handle tables having such numbers. It's more secure to keep generating new random numbers, this way you find different values for number N, the product of p and q, this is ensuring some stronger property in RSA. Now soon we are going to see what are the attacks against RSA and we need some nice **further properties** to have an approach.

You have to encode, encrypting is different from encoding today, encrypting means to compose the ciphertext, encoding means using a code to represent such an information. You have to **encode the string** like a number, ok? This is you encoding, then you encrypt. Here the step of encoding means encrypting, this was used in the paper. We can compute exponentiation in some a fast way. Requiring a linear number of operation. You can see log N because N it's not the size of the input, just the value. The size of the input is the log of the value. This is linear. We would like to make more clear the idea of one way function. Starting from x, computing $x^e \mod N$ is easy.

We think it's hard to compute **x starting from this information**. The one way trapdoor function has been defined. What is a one way **trapdoor** function? A function behaving like a one way function, very hard to be inverted, but if you are provided with some **extra information,** the computation becomes easy. This is fixing the idea of a one way function, the exponentiation made in this way is considered to be a one way function, you remember that no formal proof about the existence of **any one way function** exists. Anyway, if you provide with a trapdoor, an extra information, what particular extra information? An extra information that makes it easy the computation. We can go back to the original number x. This is very important. If you want to use a one way function it needs to be a one way function **with a trapdoor,** so the extra information makes it easy to invert the function.

Attacks to RSA

Now we introduced RSA. It looks nice and somewhat easy under an implementation point of view. Can we attack it?

Now we see some examples for **attacking RSA.** Some will be discussed, other just proposed for you to make other analysis. The goal now is to specify some implementation details about RSA so that all the weaknesses of RSA **are removed.** For attacking RSA we can find p and q given N. This is believed **to be hard**. There are some suggestions but it's good that p and q are not too close together, it's also good if these 2 numbers here (p-1) and (q-1) they have large prime factors.

A first attack is **factor N**, so you can compute the private key starting from the public one. Some messages may be easy to decrypt, if you just apply RSA according to the definition you can get some ciphertext that are easy to decrypt, they are just **particular cases.** About the factoring it's nice to say if you can manage to factor number N you can break the security of RSA, you can compute the private key. The vice versa is an interesting open problem, if you consider RSA like a function, if you invert RSA is that implying that you are also able to factor N? **Open problem.**

For several years they have been proposing challenges with money to factorise some numbers. It's 10 years the challenge is **no more active**.

We can attack RSA if we can factor, but we can assume the attacker **is not able to factor**, as long as N is big enough. Other attacks, there are messages very easy to be decoded. If message is just number 0, 0 to the power of e is 0. So ciphertext will be equal to the plaintext. If the message is number 1, 1 raised to whatever exponent is still 1. **Again it's very easy to understand**. This strange property in the case where the message is number (N-1), in this case the message you get the encryption is again (N-1), message itself. This is easy to be seen. This is easy to understand if message is (N-1) is encryption is still (N-1). Exponent e must be odd and greater than or equal to 3. If you are computing the exponent e on this base N-1 mod N, it happens that the result will be again N-1 because notice that the square of N-1 is equal to 1 mod N.

First, there is not so frequent that we are encrypting these numbers, we use an approach based on **sorting,** adding extra bits using some approach, also random bits. Instead of encrypting the message you add extra bits. A practical implementation will be seen next time. If both N and e are both small, if you choose e = 3, it happens that m^e is smaller than N. Computing the root of this number is very easy in arithmetic. In this case the solution is adding some **extra bits** to the original number to make it big enough, we don't like small numbers.

Good morning. I try to do what I'm expected to do. Today is the day I planned for the **first assignment for homework**. I want to organise this time by making the topic and then by describing assignment for the first homework, if we have time we will start talking about digital signatures.

If you remember we were talking some **theoretical attacks against RSA**. Related to the fact that in some particular cases the approach of RSA it's not robust, because the attacker can exploit some mathematical properties. It turned out that to add extra bits to the message in many cases just add random bits to your message, that's not a small number. There are other variants of attacks that can be exploited by the attacker and there's also a very strong property that can be exploited, we need to be aware. If we continue to have a look the list of attacks we're going to describe **is not complete**, there are more advanced attacks, more or less the way we choose to implement in the practise RSA is able to **contrast all these attacks**.

So in this slide we have another example, where the public exponent has been chosen to be a small number, number 3, if it happens that there are **two similar messages**, for example a message is number m, another message is number m+1, so if we are going to encrypt both messages we get two different ciphertext, c1 and c2, you can very easily write a **system equation** fully defining such information that is allowing to extract the plaintext message m by **simple computation** like this one. If you just want to try, you take this formula here to the place, c1 and c2 by the definition and you find. Some people managed to find a way for **extracting number m from the encryption** of number m and some other numbers related to m.

Also the more general case of message m and another message that's linear depending on m, has been **faced and solved**, this is just longer under a mathematical point of view, but it's very similar to the previous case. When the adversary manages to have the ciphertext associated to message m, or some message linearly depending on m, according to **2 constants e and d,** you can again find a formula that is being the value of m starting from the 2 ciphertext, the next slide is just showing the **front page of the paper**, I don't know if it's clear enough, here you see messages m1, m2, they are related by some relationships, if you have some other information like this one, the **encryption of other numbers**, you can write a **set of equalities**, a way to find the original message. This is the last passage.

So I’m not asking to completely dominating such a technique of attack just because it’s like another statement for enforcing the idea of **small exponents are dangerous**. Despite of this knowledge many people today still underestimate the danger of small exponents, very people still like using e equal to free, it’s not a good idea, attacks like this one **become possible** and people can easily take formulas like this one and **encrypt the original plaintext**. This is the same issue of the previous slide, the solution is to choose a large e. Notice that while all solutions either talk about adding extra bits to the message or talk about using larger values for exponent e, there are other attacks based on the **Chinese remainder theorem**, well known in integer algebra.

I don’t know wether you already know this property, again we are not going to go deep in these properties, this is basic, so if you have some numbers, they are all positive integers, they are co-prime, whatever you choose 2 of them they are co-prime, this is the hypothesis. For any given integers you choose there exists an integer x that is **solving the system of simultaneous congruencies** like these one. Every congruence mod n_i. It also happens that all solutions of this type are congruent module the product $N = n_1 n_2 \dots n_k$. This is a very nice property, you can use such a property for another type of attack, again to make a simple example assume **the exponent is small** and then assume you are sending the same message to 3 different users. You can expect that different users will have different numbers N, they are choosing different p and q. The public key will be such pairs you see here, **different pairs** all of them however using the same exponent. So the same message m is obtained by computing $m^3 \mod n_1, \mod n_2, \mod n_3$, etc. by using the Chinese remainder theorem you can compute $m^3 \mod n_1 n_2 n_3$.

There is a very high probability that the **numbers are co-prime**, each of them is the product of two large primes. Message m is smaller than m1, smaller than m2, and than m3. As a consequence m^3 will be smaller than the product of the 3 integers. For the attacker it’s easy to compute the cubic root and get the original message. Adding random bits to the message will **avoid equal messages**. The attacker can use cryptography like this one.

So, again, adding random bits, another possible vulnerability is related to the fact that the **message space is small,** it means that even if your message is composed by many bits maybe the message should respect some framework, some pattern, just because the message is a word belonging to some language, the grammar is providing special patterns provided to every language. If the legal words of the language are few numbers, the attacker **can compute all of them** and then can encrypt all of them **using the public key**, it’s easy for the attacker to check if the ciphertext is one of the computed.

The solution is **add random bits** to the message. Other possible vulnerability is related to the fact that 2 different users are having the same number N, if they can have e and d. So, if it happens, the owner of the private key can write a set of equations, since he’s the owner of the private key he knows both e and d, and knows all the relations between e and d, and this means he can manage to find p and q **without factoring number N**, he will get p and q by exploiting the knowledge of N. Such person gets to know p and q, so if some other person is having the same number N, the first person is **enabled to attack** the second person, he’s knowing the factors p and q, he can find the private key starting from the public key.

The solution of this issue is let people choose by themselves number N. It means that users are **not really choosing number N**, but in many practical solutions the piece of software providing good choices of p and q, is depending some actions like moving the mouse or typing on the keyboard, the probability that 2 different people get the same p and q is **very very low**, you can expect some issue like the one described in this slide is not dangerous, because of its extremely low probability.

RSA properties

One of the most famous properties of RSA is the **multiplicative property**, if you have a message m, and this number happens to be the product of 2 numbers, m1 and m2, when you are encrypting message m1 times m2, computing such by using **exponent e mod N**, the result will be, by the basic properties of such arithmetic, the product of the encryption of m1 and the encryption of m2 mod N. Just because you can distribute the exponent e to the two.

So, if 2 messages are 2 ciphertext the product of such messages will be to the product of the corresponding ciphertext. This is allowing the attacker to run dangerous attack, of course this property depending on the property of the exponent can be applied in the case **you have many messages**, not just 2, so what is the attack? This is an attack belonging to the family of the **chosen ciphertext attack**, suppose the adversary want to decrypt some ciphertext, the adversary knows that the ciphertext has been obtained by some unknown message m raised to the power of e mod N, the adversary can very easily compute a new number x, the product of the ciphertext above, and a factor here, $2^e \mod N$.

If the adversary computes such a number he can run the chosen ciphertext attack and can **ask the oracle** for decrypting such message while once it has been encrypted, so in practise he can ask the oracle to obtain number Y, equal to $X^d \mod N$. According to what we know about the original message X, it happens that you see here the definition of X, you can write it in this way, this means that **X is the encryption of 2M**, so while decrypting X the adversary will get the knowledge of 2M.

This is just a chosen ciphertext attack, the ingredient the adversary needs has the ability of asking some oracle for doing some computation, notice please that this is not just an instance of **lunch time attack**, if the adversary manages to find the decryptor free he can use the decryptor, this is more tricky because the adversary can ask the oracle to **make the decryption** and the ciphertext decrypted is appearing to some unknown ciphertext, the officer that is working on the machine operating on decryption can say ok I can do that because this message is not corresponding to no message I have encrypted, I'm not doing any damage, **this is dangerous.**

So more in general the adversary can do that, in the previous slide the adversary was choosing number 2 to make the attack, in general the adversary can just **choose other number** to run the attack, here the number is X and then ask the oracle to decrypt the new ciphertext. The typical solution for contrasting this type of attack is requesting that the message should **respect some structure**, it means the idea is make the message be composed by some parts and every part should satisfy some rule, maybe also by introducing some other measure or some other numbers for formalising the dependency between the parts, if you introduce some pattern, the oracle is no longer working because ok, the adversary can easily compute this number X here, by multiplying the ciphertext by another number, it's very easy, but if the original message m is satisfying the property of having such structure, the probability that the message 2M is satisfying the same property about this structure is **very very low**, this is breaking with high probability the tools available for the attacker, if he wants to use such type of attack, the adversary will choose a number 2 and will try to ask the query to the oracle, since with high probability the structure of the new message is not correct the oracle will **refuse to work**, he will detect some anomaly, so the adversary can choose another number, we expect the adversary can run many many queries to the oracle, making the attack complex and can't be run in a batch, it becomes an online adaptive attack, it needs more resources.

Other attacks on RSA

Ok, other attacks. There are other attacks that are related to the implementation of RSA, once you have understood all the issues discussed and you decide a pattern, extra bits, random bits, to use larger exponents and so on, you make your **implementation**. If the adversary is able to analyse your implementation he can try to analyse the way your implementation is working by checking **the time required** to compute some values. Or just checking how much energy has been used for computing such an encryption. Decryption is **much more difficult**. Encryption is using a not so big number e as exponent, we have decided that 3 is not a good number, but there is another very used that is 2^{16} that is in practise used and **not yet big**, when we are talking about keys having thousands of digits, the decryption will be very demanding in terms of energy, if the adversary gets information about the consumption of energy of running time can use such information to get more knowledge about the original message, the solution is adding some **random steps**.

Other properties

After this quick discussion about possible attacks we understand that the classical textbook implementation RSA **is not safe**. Several attacks are possible in many cases. The idea is ok, you want to encrypt message m, preprocess m so you get a new message m' and then you can apply RSA to such m'. Of course you need that m and m' **have the same meaning** or you have a very easy function that gives you m' starting from m and vice versa.

Here you can see a short list of main properties of RSA. Don't forget the exponent e must be co-prime to $\phi(N)$ and that the knowledge of p and q is enabling **everyone** to know the private key. For making short computations is good that exponent e is chosen small. This is a **free choice**. Decryption is longer than encryption. In any case if you try to compare the efficiency of Rijndael there are several **order magnitude of difference.** It's not true that Rijndael is just "faster" than RSA.

Different order of magnitude. People would like using RSA for encryption, if they need privacy or confidentiality they would prefer to use Rijndael, **much faster**, providing **high security**. We need to understand the role of RSA and other algorithms like this. I think I already informed you that when the 3 guys they designed the RSA then they found a company named RSA that started to do a lot of applicative research for defining standards for a **good implementation** of RSA.

Public key infrastructure

The result of this standard is known as **public key infrastructure.** Because exactly an infrastructure, you need a lot of standards defining how to do the steps for a good implementation of RSA. One of the first result proposed by the company RSA and **immediately standardised** as internet standard, was called **public key cryptography standards**, PKCS. If you check documentation you will find up to number 15 I think.

Then during time the several standard have been evolved and some numbers have been just **retired** because no longer needed in some standards. It's interesting to see the first standard number 1, that is defining a standard to send messages using RSA. If you start from the original message M, you are going to encrypt another message that is m, this is the m definition, and then you see the structure of m, it happens that **only the last part is the original message m**, the message m is starting by 0 , so that you can make it sure that its value it's smaller than N, where N is the product pq.

Then there is just a code saying you want to do encryption number 2 if you use number 1 means **you want to do signatures.** Some random bits here, a pattern 0 for denoting and then the message. In this way you obtain that if you are encrypting any time the real number you are going to encrypt is different, this is providing a good property to the encryption because the attacker **can't even check** wether Alice is sending the **same message** Bob. Even if the space of message is small, in this way it's big enough so the attack based on the idea of analysing all possible messages is useless, the space is too big.

This is the original version of PKCS, number 1. This is made **obsolete.** We are discussing just for historical reason. Today there are stronger approaches to the preprocessing of message m and that is a very nice proposal given in the 90s the name of this proposal is known with **OAEP**. This is a way of **padding** the original message for contrasting all the types of attack we have seen as examples in the previous slide and it's also providing a **fast way** for encoding and decoding the message.

Again the idea is take your message, **preprocess** the message, get a new message, then encrypt the new message, of course this should be **completely invertible** and easy to invert, you can see here the schema. At the beginning the authors were **not completely aware** of the power of such an approach, anyway you see here that the term that is used here is **optimal asymmetric encryption padding**. This is the meaning of the acronymous here. This slide is completely describing the approach, you can see here the original message m and here in the below part you see what is **really going to be encrypted** that is a message composed by both X and Y.

In the left part of the slide you see the meaning of different symbols, in particular n is the number of bits, numbers k_0 and k_1 are just integer that are fixed by the implementation and according to what you are seeing in this picture you see the original message m is composed by **such a number of bits**. You are assuming some rule saying ok, I know the number of bits of RSA, then I decide to **reserve some of them** for the message and when fixing the implementation you choose a number to be subtracted to m, this is the number of bits representing the original message, then you just **pad this message** by a sequence of 0s, how many 0s? k_1. Here you have the pattern message.

Here you have another string *r*, just a **random string**. You must have a good random generator and we will be talking about the quality of random generators in the next lessons, good random bits, how many? k_0. It happens that in this schema we are going to use **2 functions**, G, H. Typically they are just **hashing functions** of crypto-graphical quality. Here it happens that you use the **random bits as input** for the hashing function, the output of the function is xored to the padded message, ok? This is giving raise to the part X of the message to be encrypted, remember the message that is really encrypted is composed by the parts **X and Y**.

Of course the hashing function G and H should be chosen so the **number of bits** of the output should be **correct**, you are xoring here, the result of the hashing functions G should have the same number of bits of the **padded message**, that is m - k_0. This is defining the part X, in order to obtain the part Y you take the part X and again you compute a hashing function, a different function providing the output with the **size of k_0 bits**. This is the exact size of the hashing function, so that the part Y is computed by just xoring the output of H and the original string of random bits, this is part Y.

It looks very complicated but actually not so much once you have understood some requirements about the number of bits and the size of the output of the 2 hashing functions. This is a way you **encode the original message**, you get an encoded message, this is the message really encrypted, imagine you are encrypting this message and then the receiver will run the decryption algorithm, once he has run the decryption algorithm he's getting back X and Y, the message that was encrypted, anyway he can recover the original message very easily, in order to find the random string r he can, since this is Y obtained by xoring, he needs to **find r**, so by exploiting the properties of the xor operation it's sufficient to compute the xor between Y and the hash of X, the value here, this is the value Y, if xoring such 2 values **you get r** of course, by elementary property of the xor.

Once you get the **knowledge of the random string r** you can recover the message as the original message padded and this is just the xor between X and G(r). Indeed if you analyse the schema you see here that X is the result of such xor operation, if you want to find its argument over here you just need to compute the xor between such X and the number here but the number here is G on the input r. Since we have just found the **way to recover** r, G(r) is known and you can compute the xor to find the original message and the padding made with unknown number of 0s.

This is the proposal. Is giving some evolution in the standards. This is a very **strong and good result**, making things for developers much more complicated, not because the schema is complicated, but if you start checking **libraries offering encryption** and decryption functions when you begin with your crypto-graphical security **you won't write the libraries again**, you will choose good libraries. Only in special cases people in big organisations need to write libraries for security reasons. When you are finding a library it becomes complicated to understand such library, when was it developed? At that time what time of padding for the original message? It's supporting what versions?

Good libraries have been made **up to date**, supporting both the original versions and the last version of PKCS1. In other cases the answer is not. Good libraries provided maybe **15-20 years ago** they are not made up to date. You have to check **very carefully** what version is supported by your library. Finding a good library providing the good cryptography, not so simple, it requires **good ability for searching** for good information. We will be talking again about it later.

All or nothing security, this is the security provided by the security with padding, the meaning is in order to find the original message m you must recover the entire message X and the entire message Y. We need both of them because we have seen the way to extract m starting from such numbers. So by just modifying one bit of X and Y you get the result of **destroying the possibility** to completely recover m, because you have the role of the hashing functions, in other approaches you can recover some wrong bits while doing decryption, wrong bits are not completely affecting the original message, but in this case wrong bits are destroying the possibility to get the original message, this is **the meaning of all or nothing**, you need to have all bits correct or you get nothing.

When we are employing some encryption **public key based** we need a public key, private key, the encryption algorithm, this is **more general than RSA**, you remember the trap one way function, we need it. You make encryption by using the public key, anybody can make it, you decrypt with private key. All standards agree about the fact that private key should be kept by the owner in a **safe way**. If it happens that for some reason the private key is compromised, starting from the time of compromising all the usage of the private key will be made not legal, no longer legal. Even if the private key is compromised at some time, all the usage before that time will be still legal, having a legal impact on normal human operations, for instance signing contracts with **digital signatures** or making sensitive data encrypted.

ElGamal encryption

About 10 years later, a mathematical from Egypt, was able to find the **answer from DH**. We defined the proposal of DH, they proposed the public key schema. Then, trying to develop such schema they found another way to setup the session key, the DH key exchange. Later, Rivest and other manage to find a solution to the problem of DH by using a completely **different approach**. What ElGamal was able to do is find a solution to DH approach using the same approach, mathematical approach. The same mathematics that was being used for the **key exchange**.

He found the correct solution using the same math that DH were using for solving the problem themselves were proposing. ElGamal is a very important researcher because he provided a lot of practical standards for **implementing the encryption**, he’s also somewhat the person who designed the **SSL protocol**, security protocol that is used by http for making secure connections on websites. He worked for some time in the netscape corporation, during that time there was the so called **browser war**, in the first browser war the fight was between **internet explorer** and **netscape**.

Without the contribution of netscape, the security of the web **could not really evolve** because it was netscape corporation the one that introduced many security solutions that became **internet standards** later implemented by all modern browsers. How to make an encryption by the approach of ElGamal that completed the idea of the approach of DH? Again, this slide is describing a way so that Bob can send an encrypted message to Alice, this is the goal. So Alice starts publishing public information that are p, g, the same information that we have seen in the DH key exchange, **g is a generator** for a cyclic group of order b. P is a prime number, g is a generator for Z_p.

Now Alice chooses a number just saying in **the DH key exchange**, such number is denoted by x. This is a private key, then Alice publishes public information, result of $g^x \mod p$. This is exactly the same information Alice is publishing during the DH key exchange process. Until now just the same mathematics. Where x is **chosen at random**. Now Bob wants to encrypt a message for Alice, this is the message, it should be a number of the Z_p set. What is the process at Bob side? Bob starts by choosing a random number, y, again **in the same interval**. This is just the same of DH, Bob chooses a random number.

Now Bob is using such a random number for making some computations, here you can see the computation. First Bob computes $g^y \mod p$, again this is the number computed by Bob in the DH key exchange and sent by Bob to Alice in order to find the g^x times y, you remember I hope. Then Bob makes the **real encryption**, he computes such a number, m times $g^{xy} \mod p$, the private key. To make decryption Alice **starts computing** the secret key. In this way Alice can compute the key and invert the key computing the multiplicative inverse, if she's able to compute such multiplicative inverse she can **get the original message** by just multiplying the message to the multiplicative inverse.

It's heavy under a **computational** point of view. The validity of this approach is limited by all this math and **computational effort**. Actually this was a first proposal, not really employed, what happens in the reality is that all the asymmetric algorithms proposed by in the literature based on the public key, are somewhat limited and they should be compared to great schemes of symmetric encryption. Practical users, they all of them **prefer symmetric encryption** rather than asymmetric encryption.

Hybrid cryptography

So what is the **real current usage of RSA** if you want also of ElGamal encryption, the real usage is ok, we can use such tools for **digital signatures**, we have already introduced the concept, we need to develop the concept more. For the confidentiality purpose what is the real current usage of RSA? Are we really encrypting a message with such optimal asymmetric encryption padding for the need of secrecy while sending a message? Only in a **few cases**. In reality what is called **hybrid cryptography** is chosen.

The hybrid cryptography is an approach to **confidentiality** where you use public key cryptography for sending a key, and then you use symmetric cryptography for encrypting the messages, in other words in hybrid cryptography you are using **public key cryptography** for encrypting keys, and **symmetric key cryptography** for encrypting messages. This is very very important. Of course public key encryption is **not only useful** for hybrid cryptography. But now the main role of RSA is use RSA to encrypt a key.

So a very **practical approach** is this, Alice generates a key, using her preferred generator, then she sends the key to Bob using public key cryptography, it's easy to do that, she needs to just encrypt the key using public key of Bob and then sending ciphertext to Bob. We have told that public key encryption and decryption is **heavy computational**, while you are just encrypting a key, the **message** being sent is **very small**, the numbers involved in the computation are not so big as in the case you have a long message.

I want to ask you now, does it make sense to use modes of operation like **CBC, OFB in public key encryption?** In the case we are just public key cryptography for just sending the key, the size of the key will be **smaller** than the size of the block. So we don't have the problem of encrypt many blocks. Block encryption, the size of the block here is much bigger than the size of the block in symmetric encryption, because **few bytes** in symmetric, many bytes in asymmetric.

Theoretically speaking you **can use** if you want to encrypt long messages. If the size of the message is greater than the maximum size you will need that. But as I was saying this is **not really employed,** it's not really efficient. There's another problem, if you are using public key encryption for confidentiality you will need to use **padding** for every block. Not just for the last block, I mean, you remember, in the case of CBC you have the original message split in many blocks and you need **padding** in the last one. If you want to use modes of operation for public key cryptography **every block** will be requiring a padding, according to a correct usage of public key cryptography. This is not only a problem of efficiency in running the algorithm on longer message but it's a waste of time introducing this. You may see this last argument as one more reason for **discouraging** people for using modes of operations while using RSA. Nobody is using RSA for encrypting long messages, only one block. Theoretically speaking is again a block based encryption, RSA, never used over more than one block. This is the main usage of public key encryption for **confidentiality purposes** in practical cases.

The message I give you, even if public key is providing new powerful result it's much more convenient using symmetric key cryptography, and you can use public key cryptography for sharing and establishing keys to be employed for the encryption.

Homework program

Let me introduce the **homework program**. This is an optional program for students, enabling them to get extra points in the exam. It's important to immediately say, the homework program is designed **just for people attending** the course. Such slides will not be published in the website. Only people physically here will be enabled to see that.

It's **strictly forbidden** to distribute information outside this class about the homeworks. You see some rules, you are expected to provide **6 contributions**, you will be obtaining **up to 2 extra points**. Every time I'll give an assignment I will also say to you what is the deadline. I invite you to **use latex** to write your report. The homework consists in some work you have to do and you have to provide a report about your work. You can see some **basic requirements** about the usage of latex.

The **article class**, 11 pt. in english, since latex is allowing several types of formatting I ask you **not to use low level formatting**. Don't use packages, packages are just extensions to latex providing a way for making smarter formatting and special effects, **don't use packages**. For bibliography you can use bibtex but it's not a package. For bibliography I suggest using bibtex, a part of latex suite.

Since latex is looking like a program, it's the same as ok you are writing a java program, don't use bit operations. I think I already told you about the origin of latex, the original word was tex. It was like using assembler for **formatting a document**. Then some macros were provided. People can just format documents using macros, still is possible using original tex commands, **don't use them**. In my mind I want to collect all contributions in one book, I need **one consistent formatting**.

The only **package allowed** is the graphicx package, allowing people to insert pictures in the document, if needed. If you need to include a picture you can use the \includegraphics. I expect that you will be using **pdf latex** that is that part of the suite able to provide the output in pdf format. Your contributions will be **valid until the exam of September 2018**. You may think of the scenario where you give the exam in January, you repeat the exam and your homework is still valid.

The requirement is **attending the class**. I will use the compilation software to detect similarities in different contributions, standard used. Email contributions to cns@dis.uniroma1.it, use your sapienza mailbox, write in a standard way. I'll try to use some scripts to get emails. The subject of the email should be a standard subject like **HW1-012345**, matriculation number. That's enough. No spaces in the subject. You will be attach the contribution to email, it's good attach tex document, pdf, png, jpg, txt, don't compress the attachment, give a **standard name** to the main latex file, like hw1-012345.tex

Other files: *-hw1-012345.*

Don't submit folders. Email body should be the output of SHA-256 run over the content of the main latex file hw1-012345.tex

My evaluation will be available **only when you are taking the exam**. Here I just describe a possible scheme for writing your report.

Title: <the given title>

(Possible) Subtitle: <HW1 - CNS Sapienza>

Author: first, middle, last name and matricola ID

Date: Opening of the assignment

Article class, no abstract needed here.

Provide a fully detailed solution, including bibliography (if any)

Homework number 1

Homework number 1. **Attacking DES**, this is the content of homework number 1. The deadline is strict, I want it until 2 November.

Title: **Attacking weak symmetric ciphers**.

Description: What I'm expecting, the homework is focused on DES, encryptor using a key of this small size. Choose whatever programming language. While choosing your favourite language I expect you don't want to write functions making the encryptions and decryptions. You need to find the libraries supporting you. Many people like using java, because is very **well known framework for developing**. Java is offering many frameworks for security and encryption it offers them.

You see here a possible **list of libraries**. I want sources and I need to link. Don't under estimate the difficulty. You must understand if you can use the library. In many cases they are provided for spectacular engagements with cryptography. You just need to use DES and DES is provided by all libraries. I don't know if modern libraries still offer DES because of its weakness. Once you have decided maybe you may choose the padding for the library, let me find a good library. You should just implement the **basic operations** for encryption and decryption. You can just use CBC or other modes.

For the most important part of the homework choose or **generate a 56-bits DES key**. Plus 8 parity bits you get what looks like a 64 bits key. Choose a plaintext and encrypt it getting the ciphertext. You are expected to test encryption and decryption.

This is the attack: setup a **brute force attack** based on generations of all possible 56 bits keys, for recovering the original plaintext. How you should generate all possible keys? It's just a **string**. The DES key, imagine this is just a part of the DES key. When you are looking at a DES key you see a 64 bits string. You see something whose length is 64. It means 8 bytes. This is just one of the first of such bytes. Actually the key is containing good information in **the first 7 bits** of the first byte.

What is the usage of the 8th bit? The usage is related to some **very old issues** about errors while doing the transmission, is used to compute a parity bit. The parity bit is an extra bit allowing the total number of 1 bit in the string will be an odd number. In this case the parity bit will be 0. It's **completely determined** by the first 7 bits. You get the next bit as a **consequence**, this is just what is expected to be here, you have obtained the first byte of the 8 bytes. You need to do it 7 more times to find all possible keys. Not all possible configurations of bits are legal DES key.

You must **respect the pattern** for parity bit. This is the property. You have to generate 64 bits keys but the number of different cases is lower. You can start from whatever result. **Don't start** from a string with all 0s, start from a string with the first bit equal to 1. I want you to **measure performance**, in particular cpu time. You have to run and get next key, check wether if your experiment is successful, you do that again, several **billions of time**. I suggest using C but I want to say something about real performances.

I expect you running this generation **taking times**, measuring times. You just compare what you have as result with the original plaintext. Somebody could say, ok this is the time I need to generate the correct key. It's also including the time for **making many decryptions**. Try to repeat the initial for initial keys and some plaintext, depends on how fast the code is running on your machine. Then you should **report about project** that is the writing of some lines of coding and the experiment you carry out and also try to have a look at the result at performance result and try to **give a comment** about performance result, it may happen that some of you may be somewhat heavily affected by a slow machine, may happen that you have a **slow machine.** If the speed of the code is not enough for running such experiments you can make it simpler by just fixing part of the key. So, if it happens that it's too slow to run this experiment try the approach of blocking the first byte of the key. Fixed. You are assuming the adversary is knowing the first byte and then you try a **smaller number** of keys. If you are using the pc of some granddad you need to block 2 bytes. For modern notebooks I expect you can do it.

FAQ about homework

I got some questions since I recognise some **FAQ** I now want to say something about that. You should know that the homework is focusing on **brute force attacking DES.** It's a cryptographic symmetric approach using a small key, 56 bits. Now typical questions are 2: my computer is too slow, it takes years for breaking, 2nd type of question, what is the strange **00000000**.

Ok, what if your computer is too slow? Well, first I want to say that breaking like 70 billions of billion of keys, you can do that but it's **not exactly a short game**. So, even if you have a good hardware your time can be long. In order to prevent useless time consuming elaborations during days, weeks, I told you that you can decide to make constant some bytes of the key. How many you want to make them constant, just have a check, you can **very easily estimate** the time for running 100 attends. You understand how long time it will take trying all possible keys? Then you decide you want to keep running your processor one day or the whole all night, a given number of hours. You can easily compute how many bits you can brute force in that amount of time, you can estimate how long it takes. 100 keys is a **good number**. It means that you prepare the attack, you make running your approach generating 100 keys, you take the time, you understand how long time it will take generating all possible keys, ok that's not visible, I want to reserve 1 day of elaboration, of computing. And using a proportion you can just determine how many keys you can test in one day, you take the logarithm and you have the number of bits.

All the other bits can be made fixed, constant, the focus here is not to win over DES but it's to setup a brute force. It's important you understand what steps the attacker is asked to carry on when setting up a brute force attack. Even if I mean, 8 bytes, you keep constant 4 bytes, remains other 4 bytes. It's ok. Of course you are making the encryption very weak but it's important the **methodology** and the way you take the time and what consideration, what comments you can add after seeing the time. The other point, at some time I told you may decide to start generating key not from a string of 0 but from another string, and I wrote another string. You can start by whatever string, initial string. There's no good choice. If you are using a generator for keys, if the generator is **really random**, every key will have the **same probability**, starting from a 0 or other will have the same probability, the expected time is not changing.

Why I proposed to start not from 0? I proposed that just for having as first numbers big numbers. If you have big numbers you will find more easily your errors. Just that. So, you can start from whatever bit, just be consistent with your choice of keeping frozen a few bytes. How many of you is participating with homework? For over 30 years I keep saying **I'm superman** because never getting a cold, flue, nothing. This year I'm dead. Please be patient about that. Also I'm not having my usually energy, I need energy when discussing topics. Digital signature. Before starting I want to try an introduction

Digital signatures

Digital signatures are for guaranteeing **data integrity and non repudiation**. Why we need digital signatures for data integrity? Isn't MAC good enough? We studied hashing functions, you can use them for computing the tag, authentication tag and we can use **keyed hashing functions**, so that Alice and bob will be guaranteed that it's the counter part sending the files. In a perfect world everything is ok but the world isn't perfect. What if a dispute between Alice and Bob? If Alice says it was Bob sending that file, and Bob replies not, it has been Alice sending the authentication tag for that file. They are fighting and a judge how can establish who of the two has been sending the authentication tag?

For sending it it's required to **know the key**, but both Alice and Bob they know the key. How to solve this problem? We want to be sure about the originator of the message. This is a **limit** of the approach based on the MAC. Just a limit. It's important to understand the limit and to understand why we need some **extra effort** for ensuring both data integrity and non repudiation, so that the subject sending the message can't say some day it wasn't me sending the message.

Now next step, do you remember when we were introducing RSA and in general we were introducing the **public key approach**? We described a nice setting where we kept saying you can make encryption by using a public key in order to get what? **Confidentiality**.

But if you make encryption by using your private key what happens? Who is enabled to decrypt? **Everyone**. Because it's just useful using the public key for decrypting such ciphertext. There is no confidentiality of course, you are obtaining an important information, everybody that is decrypting the file will know that one only person could encrypt it, by using a private key. Since by definition the private key is **secret** and the owner of the key must keep in a safe place the private key if you can decrypt the ciphertext you can be sure about the fact it was the legitimate owner of the private key making the encryption.

So we can base some approach to digital signatures over the encryption made using the private key? This is the **basic question**, we already introduced this question and the idea is maybe yes, it looks a good approach. Now I want to show you what happens if we in a simple way just try to define this way a digital signature. Now I'm going to describe an approach that **it's not working**, I describe it just for **teaching purposes**, to make you aware why we need an extra ingredient. Let's just assume that we are introducing the idea of digital signature and we decide to define what a digital signature is in the following way.

We are not assuming that now we want confidentiality, but we want data integrity, non repudiation. Assume that this is the message M, and assume that Bob is the owner of a private key k_s and a public key k_p. So the hypothesis we are making is to define a digital signature for file M, message M, in the following way, a digital signature of M, the **encryption** made using the private key of Bob of message M. This is a ciphertext C. I want to repeat, this is **not the real definition**. We are just examining a possible definition for discovering that is not good. We need an extra ingredient. If Bob wants to prove it was really Bob the sender, he can send to Alice the **pair** message and signature.

Alice takes the signature, it's ciphertext, Alice decrypts the signature, can she? Yes, she can because for decrypting this ciphertext is needed the public key of Bob. So she can obtain the decryption and then she can check wether the decryption is the same as the message sent here in plaintext. Also in this case we are introducing a model where the sender is sending **a pair of information**, a pair of files, the receiver is getting such information and it's running a **verification algorithm** saying accept or reject. Assume that the verification says accept. It means Alice will think that the originator of the message has been Bob sending such a message. If the message is changing, if some attacker wants to change one bit of the message, the verification **will fail**, why?

Because if the attacker is changing a bit of the message the verification, the decryption will be giving a different message. If the attacker changes a bit of the signature, the verification will fail. With high probability. Are you ok with this basic introduction? Perfect. I want to show you an attack, against this model. Let me write again the definition of this **hypothetical approach**.

Sig(M) = $K_S(M) = C$ and verification is checking the decryption of C.

$Verif : K_P(C) = M$ if equal or not. I want to show you an **existential forgery**, do you remember forgery? We discussed about it. I would have some slides reminding the concept. Existential forgery we discussed about that, it's important to remind the existential forgery is the ability for the adversary to **find a pair**, like message and signature, such that who is verifying the signature will **obtain accept.** Existential forgery is a minimum level of attack, it's not requested that this message here should have any particular meaning, it can be even meaningless, a totally stupid message, a random sequence of bits. Why?

Because if you send to Bob a random sequence of bits and a digital signature and Bob makes the verification and it says ok accept, Bob will say Alice was the sender of this message. Now I can't understand the meaning of this message, what I'm missing? Since the **security of digital signature is very high**, the highest security I would say, for authenticating the sender. Bob will be sure about the originator of the message, so an existential forgery seems a weak attack but it's very bad. We don't want it.

There are two other types of forgery, where the adversary is having more power? The **selective forgery** is an attack where the adversary is having the ability of creating a pair and such a pair is composed by message and signature, the message has been decided in advance before running the attack, of course this is not meaning the message can be whatever message, maybe it's satisfying some **mathematical property**, the adversary got to manage some tweaks, mathematical features about the encryption, by exploiting such information the adversary is able to **create some particular pairs**, not many, just a few. This is the selective forgery. Also, if he can make a selective forgery he's also able to make an existential forgery, this is implied. **Strong attack**.

Of course the strongest attack is the **universal forgery**, this is the full power from the adversary who is having the **full ability** to decide whatever message and to show the correct signature for that message. This is the universal forgery. Strongest attack. Being able to run the universal forgery is implying to run the selective forgery that implies the existential forgery. If you want to prevent the adversary running such forgeries, what is the particular forgery you have to focus? If you prepare some solution, algorithm, such that you can **prevent** the existential forgery, will you be happy or maybe you will be more happy if you are able to prevent the universal forgery? What is your **real enemy**?

If you have an algorithm that can contrast the universal forgery is it able to contrast the existential forgery? I am able to contrast the universal, the adversary **cannot choose whatever message** and sign the message, but maybe he can choose one message. I have to focus the **existential forgery**, since universal forgery implies the ability to execute the selective forgery who is implying the ability to make existential forgery, if you deny this you deny all of them. Because not existential is implying not selective that is implying not universal. This is not difficult, it's very important that you can understand this methodology of reasoning. We need an approach to **contrast the existential forgery.**

I want to show you an attack, the attacker is generating a random file. R a random file. Can the attacker do that? Of course he can. The attacker is now computing what? The encryption by using the public key of Bob. He can because this is the public key, available to everybody by definition. Let's call this T. Pretending that T is a message, and R is the digital signature of the message, made by Bob. When Alice receives this pair she would like to verify. She verifies. Encrypt the ciphertext, you should get **the first argument**. In this case the verification is what? Take the ciphertext, encrypt this using the public key and get the first item. If we encrypt the second item using the public key by construction **we get T.**

If we define the digital signature in this way it's easy for some attacker to run an existential forgery attack, generating at random the signature and obtaining the original plaintext associated with the signature. This is why the idea of such a definition of digital signature **is not good.** The solution to this problem is the introduction of another step, making usage of **hashing**.

When running the verification this equality is holding, so the verification step will say accept. We define digital signature, **secure by definition**, providing a way to run the verification step so that the user of this service when seeing a digital signature just needs to do the verification. Verify the message according to the approach. So if you run your algorithm and get accept or reject. In this case if the definition of digital signature is the one we introduced we see there is an **easy way** for the attacker to run an existential forgery attack so the verification will say accept this pair, message and signature, of course it's **not true** that digital signature we defined is secure.

Hashing in digital signature

If we introduce hashing in this process we obtain 2 good results. First, the previous attack is **no longer working**. Second, the efficiency of the signing algorithm is better. We like it. The most important aspect is to ensure that the attack we have seen is no longer working. Let's revise the definition of **digital signature**.

By the new definition, the good one, digital signature of message M is the encryption by using the private key of Bob made on what? On the **hash of the message.** This is a new definition. What is the verification step in this case? Of course when Bob is sending the message to Alice he's sending message M and Sig(M). The verification process made by Alice is take M, compute hash of M, easy. Everybody could do that, the hashing function is **public**, not a keyed function. We need to use a good hashing function of course, maybe of crypto-graphical quality. A cryptographic function.

So the process is take M, compute the hash, now take the signature and **decrypt the signature** using the public key of Bob. We should get from this step the hash because this was the encryption of the hash. Now just compare these 2 items, are they equal? If yes, verification will say accept. If not, verification will say reject. You understand the structure? I think it's enough simple, it's very important you are able to focus the topic. This is the **most important part,** if you get this part you get digital signatures.

The verification step requires to compute the hash, decrypt the signature, the decryption should give the same hash. The decryption is made using the public key of the signer, of Bob. In the previous attack the attacker was generating a random file. This random file, after seeing the analysis, the random file was **the signature**, according to the previous definition. The original message was obtained by computing the decryption, with public key of such random file. If the attacker starts the same process, what happens? The attacker is generating a random file R. He's trying to generate the signature, next step is **find a message**, so starting from this random file the attacker can of course compute the encryption by using the public key of Bob of this random file, he will get something, what is this? According to the new definition? It's the hash.

$$R -> K_p(R) = T$$

The attacker should be able to send a pair message, signature and the message should be hashed to this value here. So what type of problem is requested to be **solved by the attacker**? Given the hash, found a pre-image.

You should **invert the hash.** Find any of the original message that are having such a **digest**, you remember the terminology. Such hash is named digest, or fingerprint of the document. You are given the digest, now the attacker should find a message whose digest is this one. Now you understand why we need to request a cryptographic quality of hashing function, so inverting it is very hard, even under an average case. This is the problem that should be faced by the attacker, why the existential forgery attack easy in the previous case now it's **not looking easy**, there is one extra step, you have to **invert the hashing function**. This is stopping the existential forgery we have seen.

What is the requirement? The hashing function should be a good hashing function, of course if the attacker is able to find collisions he can run **very bad attacks**. Why? Now Bob is sending a pair to Alice. The verification step is easy, $(M, K_S(H(M)))$ she expects to find the same digest here, compare, if equal ok, if different reject. easy. What if the attacker wants to make a **forgery attack**? Suppose the attacker is inventing some message M', let's call the signature S. (M', S). What happens if the attacker just takes the signature with another file? This attack is **not working**, the verification step is easily finding that the digest of this message is not corresponding to the encrypted digest. What if the attacker is able to find **another message M"** such that the hash of this message is the same of M? Different message, same digest. H(M") = H(M).

In this case the attack is easy, it's just required to have a pair like this one, that is a legal pair, now what the attacker needs, to find a colliding message, if he manages to find a collision, he finds M" having the same digest as M, this signature will be good also for M". You agree? So of course the full power is obtaining the moment where the attacker gets the private key of Bob, with it he can sign whatever. But if doesn't get the key but he's just **collecting legal pairs** like this one, he can take one of the signatures and try to find a collision to this message, the message is known, given a message find another message that is colliding.

This is why we need hashing functions **strongly resistant** to collisions, if the hashing function is strongly resistant is also weakly resistant. We found 2 reasons why we need a cryptographic hashing function, the attacker given the digest he should **not be able to find a pre-image**. Second requirement, hard to find a collision. These are two qualities of hashing functions, they make hashing functions having crypto-graphical quality. We have another interesting result, we have already discussed the fact that public key cryptography is more demanding under a computational point of view, when you are making encryption by using private key, your private key is very long, you have to run **many computations,** lot of time.

So if you want to sign a long message, according to the previous definition, the bad one, you have to compute the encryption of possible long message, now according to the new definition, the correct one, the encryption is made on a digest, the size of digest is small, cannot be long, whatever is the length of the message, **the size of the digest is a fixed**. SHA-1 is no longer secure, 160 bits are no longer ok. SHA-2 in the case of 256 bits for the result of the hash, the digest, it's **secure** so if we are using SHA-2 here or computing such a digest we have to compute the encryption of 256 bits, it's a small portion, even if the algorithm are demanding lot of computational effort, under an **asymptotic analysis**, the size of the input is very small here, you can get a quick result.

There is another interesting consequence, if we remember the original definition of RSA, there is the **multiplicative property**, remember it? So, you can somewhat build a case like this is the message M1 saying that I owe Bob 20$ and the message 2 is 100$. If you encode such letters you may get that the product of the 2 numbers m1m2 is just corresponding to the concatenation of the 2 messages. If you can manage that it means that by starting from 2 digital signatures you may compute a new digital signature. If you introduce the hashing this is no longer possible, you remember? The introduction of the solution is **solving the problem**.

So the general way to manage this tool, digital signature, is we need some way to generate public keys, we need an algorithm for signing and for the verification. The signature can be made by **randomise algorithm,** introducing random bits in the signature. We said we need to introduce some random. The verification is normally deterministic. Today there are several things for using digital signatures, I would say they are most used but there are also RSA and DSS. Still today RSA is a good sign algorithm. Of course, with time passing, we need **longer keys**, because of the computational power growing.

In particular case of RSA I remember the original proposal for using RSA for computing the digital signature was ok, use SHA-1. Today if you want to use a modern security proposal just use **not SHA-1,** but SHA-2 or SHA-3, and a longer key, at least 2 kilobits. We can refer to PKCS number 1, to make encryption for signatures and confidentiality. In this way you see this byte here is changed to 1, in the previous time we saw this was 2 this number for confidentiality, but the general schema is very similar to the previous one, start by 1, a string of 1 bits byte 0 and then some integer number that is denoting **the hashing function** we are using and the digest. You compute the pattern, this is the real message you are going to sign. You want to sign message M, you compute this string here and you have the digest of M. Then the signature will be the encryption of this string made by private key of the signer.

You understand there is also some **small details**, making this difference from encryption for confidentiality. There is an extra field here, specification of used hashing function, in this way if you allow a family of hashing functions you can compute **different signatures** for the same document, using different hashing functions. In many cases the choice is not just random but it's made according to some general list of preferences where the parties are trying to pick the top part of the list.

I want to say that during time the standard for RSA has evolved. Last time we saw an approach, we call it **optimal**, for encryption. Security **all or nothing**. It was providing a new version of PKCS1. Also in this case there are new versions, but if we want to follow all details of modern versions we have to keep into account **many details**, somewhat boring in some cases, you add some extra bits and so on, it's very important you know where you can read if you want to **know more**, maybe suppose that for some reason you are involved in some project where you are required to contribute to the new version, some pattern of the encryption based on RSA, well this is I think the last version of PKCS number 1, a public document you can obtain from the web, the version is 2.2, the document is offering the icon of the RSA company, if I'm not wrong, the previous version 2.1 was received by the internet engineering task force that encapsulated this version inside some document.

I think this one is not standardised but I may be wrong, you see here some detail, where there are some I guess you can recognise some acronymous there are also other that we have never discussed, while you are going into details you have to face some **practical issues**, for example I give you my message and the message is let's play cryptography. A string. Ok, we have to transform this **string into a number.** How we do that? Ok encode in some way, yes, you can encode in many different ways such a string, the easiest approach is consider the representation of string like a sequence of bits and this is just a number, you have some implicit encoding, there are several other considerations that we can carry on so that there are **better choice for** running encodings, several types.

We just need to be standard, if you want to see all details have a look at this document, not so big, the particular I want to say the most interesting part is just 20 pages, where you see the **encryption schemas** and signature screens. In some extra details, we are not going into details about that, this is I'm saying ok we stop here about such details on RSA, if you want to go behind you should have a check to this type of document. Notice there is a chapter whose title is **data conversion.** It's a public document, you can google.

At the beginning there was PKCS number 2, number 1 and 2 were for encryption and signature, after some time number 2 has been **deleted** and just number 1 defined the details for both confidentiality and signature. As a concluding for RSA I want to stress the concept that when you are **using RSA** you want to sign, the other typical usage of RSA is send an encrypted key. You don't use RSA for confidentiality purpose, for encrypt a document, you encrypt just a key. Because of its complexity, computational effort, much bigger than Rijndael.

Signature scheme by ElGamal

Last time we saw a scheme for confidentiality based on ElGamal approach. The scheme was somewhat finalising the search made by DH for a solution to the problem they introduced. Now we have a very similar approach, still by ElGamal, defining a **signature scheme**. I want to say that now both ElGamal and next algorithm we will consider are implying some mathematical parts that are more demanding than the previous ones. We are not going into details about such mathematics.

Let's see the ElGamal signature scheme. You see an algorithm here, because first you have to **generate your keys**, private and public key. Choose a prime, this is number p, of **many bits**. We want the discrete logarithm is hard in this group. The problem of solving a question, it is the problem that is hard to be solved and makes DH a good approach for **generating a session key.** Find a generator, let's call this g. This is similar to DH standard, when you are looking for a session key. Now choose number x at random in this range. This is again similar to what DH were proposing because both parties are doing the same, compute $g^x \mod p$ and the number will be denoted by Y. Now you assume the public key is **pgy**. This is the same we saw for DH. The private information is the exponent x chosen here. This is **generation.**

If you want to sign a message m this is the approach proposed by ElGamal. We have defined a public and private key, now we define new keys, again public and private. Why we need an **extra pair?** Because the proposal is let's make a standard usage of our keys. Let's also generate **temporary keys** for running the encryption, for every signature you are generating extra keys. If you do that, every time you sign the same document you will be getting **different signatures**, but the process of verification must work in any case.

So, the approach is ok, we have a public key and a private key, they are **permanent**. For every signature let's generate a further pair of keys, public and private. What is the process for **running signatures**? Ok, start by considering the original message and start by computing the digest of the message, we hash first, again this is a **crypto-graphical quality** hashing function. Now we can pick a number key in this range to be co-prime to p-1. Pick this number **at random** so there is a new choice of a random number, and compute $g^k \mod p$, where k is the number just picked. This is number r. Now compute another number, look at the number, notice in number r while computing it the digest of the message is **not used,** this is not depending on the message, so what is the consequence if it's not depending on the message?

You can do that in advance, you can do what is called **preprocessing**, if you like it, you can prepare a **table of numbers**. Not really recommended, you will have the problem of putting in a safe way such information, anyway compute this number r, is the result of another exponentiation, and now compute number s, defined like that. What is that? You see here the contribution of the digest. Small n is the digest, is the output of the hashing, and you are requesting to compute such a number where you see the contribution r signs x, and then you are multiplying this number for the **multiplicative inverse of k**, mod p-1. If it happens you get 0 from this computation try again. Try again means pick another random number.

And by definition the signature of this ElGamal approach is the pair (r, s). If you are using this type of signature when you will be sending your message you will be **sending a pair**, the message, second item of the pair is this pair. The signature is a pair, the second item of the pair. Ok, you can see here there is somewhat a tricky way of computing number s, and this is strongly related to the way we use for running the verification, so ok **we want to verify**, suppose somebody is computing such a signature and he's sending a message, M signature. Starting from this information you want to **run the verification**, start by computing the hash.

Now, number r and s, they should be not arbitrary numbers, number r should be less than p, of course **we are not considering 0**, when we get 0 we get a **trivial case** and we want to restart the process. Number s for the same reason is obtained by this other computation mod p-1, so number s should be less than p-1. What is more interesting is that for accepting we want to check that this equality, is y^r times r^s equal to $g^m \mod p$, if they are equal we accept, otherwise we reject. Why we decide to **accept or reject** on the base of such test? Ok, you see here in this slide there is such a formula marked by the trivial start, this is the formula saying how to get s. How we get s?

If we multiply both members here left side and right side by number k, we get s times k, here is it. And on the right side k^{-1} is the inverse. This term is becoming into m - rx. We get in the right side it's just number m. After such multiplication we know we are going to exploit this definition, since this is the definition of r it happens that $r^s = g^x s$ and y, defined as g^x, x is the **secret number**, means that $y^r = g^{rx}$. We are considering that because the verification is asking to compute this here. $y^r - r^s$. We have to multiply these 2 terms here. The official body in the states for defining the standard chooses a **variant of ElGamal.**

ElGamal variant

Passwords are a very **bad way** to make authentication. But people still want password. Now we talk about **DSS**, digital signature standard. There are other bodies choosing DSS rather RSA. DSS is inspired to the ElGamal approach and according to the original formulation this is using SHA because the standard hashing function chosen is SHA family. Ok, you may need **two different acronyms**, people are talking about DSS and about DSA.

You should understand what is the difference, the **DSA** is just the algorithm to **achieve digital signature**, the standard is a set of further details, the algorithm so we can practically employ such an approach. Now we focus on the algorithm for making digital signature, notice that we have seen RSA you can use it for **confidentiality and for signature**. We saw a first proposal by ElGamal, just for confidentiality, today we assume another proposal by ElGamal just for signatures and again the DSA algorithm is just for signatures, so the property of running an approach where you can very easily interchange the keys is well defined for RSA and variants.

Ok this is the basic of the approach, so the idea is working with two prime numbers, the name of these prime numbers you see p and q, and you have to work with 2 prime numbers that are **satisfying this equality**, you have a bigger one p, a multiple of q, plus 1. Or in other words, **(p-1) should be a multiple of q**. Number p is just a prime number, again we want that the discrete log problem mod p is not the same as the previous approach for ElGamal. Once we have agreed on these 2 numbers p and q we need to focus on another number here denoted as alpha. That particular number such that if you compute alpha raised to the exponent q, you get 1 mod p.

So you may think of some mathematical problem, given p, given q, how to find alpha, the root of 1 mod p or that number such that to the power of q is congruent to 1 mod p. Ok? **Does this alpha exist?** Can you find it? It's **easy** to compute such an alpha number, next slide is on how to compute such a number. For I would say that in my opinion if I'm not wrong in 3 times **I asked at the exam** the question "**compute the 7th root of 1**". Just asking to use this approach, in all cases students were not able to compute a root of number 1. So, the problem is given p and q satisfying this equality find alpha such that α^q is congruent to 1 mod p.

Generate a random number, h, within this range here, greater than 1, less than p-1. Once we generate such a random number we can compute another number that is g that is equal to $h^{(p-1)/2}$. The properties of p and q are such that p - 1 divided by q is just number j, here. It's this coefficient saying proportionality between p-1 and p. Generate such a **random h,** compute such number g, it may happen than g is equal to 1, it's useless, just try **a different h**. The successive steps of algorithm will be not secure. Try again so that we find a g different from 1. Now if you compute g^q, is equal to h^{p-1}.

Now by Fermat theorem you have that $h^{p-1} \mod p$ is congruent to 1. We found a number g such that raised to the power of q is congruent to 1 mod p. We found alpha. Choose alpha to be equal to g, there are **many solutions** of course. It's easy to deal with the qth root of number 1 mod p. Now let's go the **algorithm**.

We start from p and q, prime numbers, related in this way we described, p-1 is a multiple of q, then also consider number alpha that we can compute by the approach we saw in the previous slide, the qth root of number 1 mo p. Starting from such information you generate **private key and public key**. What is the private key? Again it's a random number belonging to this interval, number s is a secret belonging to the interval 1, from 1 to q-1, q is the **smaller** between prime numbers. This is your private key. The public key? Number p, number q, alpha and the result of this computation here, take alpha, raise to the power of s mod p, the result is y. So the public key is pqalphay. Is it easy or hard to find s? It's the problem of finding a **discrete log.** So it's hard by hypothesis to find s.

What is the signature of such message m? Again the DSA algorithm is inspired to ElGamal generating extra keys, I want to compute a signature. Of course the consequence is that every time you sign about generating extra keys the signature will be different. There is a built in mechanism to introduce **mechanisation** in the generation of the signature. So choose a random k, is kept as secret, and the signature is composed by **2 parts**. It's a pair. Part 1, part 2. First argument, second argument. This is the signature. Now we are going to see the **definition**.

This is the definition of part 1, the first item of signature. Look at the first item, we compute $\alpha^k \mod p$ and such result is computed mod q. P is the bigger so the result is a number between 0 and p-1. This number is **transformed in a smaller number** between 0 and q-1. Part 1 again is not depending on message, is just depending on number p and q that are constant. On number alpha, and the secret number q.

The **second item** is depending on the message, you get the digest of the message using an hash function and then how to compute the second part? You compute the sum between digest and what? You see here the product between number s, number s is a **secret** generated at the beginning, here. Chosen here, and you multiply s by part 1, such a number between 0 and p-1 and the outcome is multiplied by the **multiplicative inverse of k**, mod q. This is what we define as signature by using the digital signature algorithm. Again we notice that part 1 is not depending on the message, you can allow preprocessing, part 2 is somewhat **faster to be computed**, it's a hash and a product and you can quickly compute this value, maybe the hardest part is compute the multiplicative inverse of k.

You use the algorithm, and here in this slide you can see some on the practical steps the for putting signature and for running the verification step, don't forget whenever you are defining a digital signature scheme **you need to define this verification step.** So again you see here the definition of the signature that is such a pair. Now the verification is defined as follows, compute such 2 numbers denoted by **E1 and E2**. What are these numbers? These are 2 numbers mod q, and one number is just obtained by the digest times the **multiplicative inverse of part 2**.

The second number is obtained by computing mod q again, the product between part 1 and the multiplicative inverse of the second part, after computing such numbers the verification step is saying accept the signature if and only if it happens that such expression. Such equality and if equality holds your verification step will be accepted, otherwise it will be rejected. Ok, this is **looking strange** because the verification step. And again we can see why this is working, by just following the step of ElGamal was using that. Accept if it happens that **this expression here is equal to part 1**, where E1 and E2 are defined in this way.

Why is this working? We just use definition. Look, the first one, we are trying to make it **explicit** the value of the digest. Can we do that? The digest is here, in order to make it explicit from this formula you have to multiply **both parts by k**, you get k times part 2, then you can solve. And you get this value. Now you can multiply by part 2 multiplicative inverse. In this way we get such a relation. Now, we use the definition of y so if we consider alpha to this exponent and y to this exponent we get what? We get the other alpha and you can sum the 2 exponents of alpha, and you get what? What is this exponent here?

This is just the expression for **number k**. We are just replacing the original definition and we get alpha to the power of k plus c times q. Why c times q? This is congruent to k mod q, it means its value will be k + a multiple of q. What multiple? We don't care, just **any integer coefficient**. Since we know that alpha to the power of q is 1, this part here will be disappearing, this will be equal to $\alpha^k \mod p$. Now if we have to **compute mod q**, we found that this parenthesis we are getting is just part 1. It's like magic but no special passage is exploited. The hard part of mathematics is to ensure the existence of a multiplicative inverse. This is a simple explanation why if you compute these terms here you should get back **the part 1**.

Summarising, again we have a permanent private key and we have a temporary private key. Permanent is S, temporary is Q. For every signature you are computing a new number k. Be careful, number k should be **protected,** kept as secret, otherwise the attacker knowing the public key and k can setup a system and get sensitive information. In particular if the attacker manages to know number k he can write 2 equations. When the adversary is knowing number k he will be able to run new signatures for new messages.

The ability of knowing key is allowing the adversary to sign, the power of **universal forgery**. The analysis of the mathematical security of DSA is strong enough. We are just following this simple passages but we are not going into details. This is good material for a master thesis. In practise we say ok we know RSA, we know ElGamal, we know DSA. DSA is obtained by ElGamal. Not use ElGamal, is good, let's **improve ElGamal** to obtain a standard, this means in practise ElGamal is not used. The signature verification step for DSS is more demanding than RSA. The signature verification is asking to compute this number here. The running time is **more demanding** with respect to RSA.

For what concerns signing documents, more or less the two approaches are giving the same speed. Still use preprocessing, you will be a little faster than RSA in making signature. DSS is requesting to generate **random numbers.** They are used as private keys, they should be chosen in a good way. It means that not only the number should look like random, but the adversary should not be able to predict such a random number. Don't forger the random number is the output of an algorithm, since an algorithm is deterministic, if the adversary is knowing the algorithm and the input **he's able to generate** a random number. You understand it's very sensitive the generation of random number. It will be the next stop we will discuss, so we can focus some details.

In many cases **preprocessing** is used when you want to implement signatures with smart cards, this makes the process faster and you want to save energy for the process of running the algorithm.

DSS is used for just signature, RSA both for signature and key management, DH just for key management. This is just a table summarising the main properties of the approaches. There is another point, when you are signing an analogic contract, they give you a paper and you sign. You need to **specify the time**. When you have an important signature, 2 parties are agreeing of something, in many cases it's requested the time and we have seen in the process of taking a digital signature there is no mention of time and day. Actually this is very important and for solving this topic the approach is very simple, once we have agreed what is a digital signature.

Ok, in those cases where we need to **timestamp** a document, to associate in a secure way a timestamp. What does it mean make association? I don't know when it has been saved, but at some time the document was existing and I'm certifying that. If you want to associate an official time the typical approach is using a third party providing the service of generating timestamps. Suppose you **just want to timestamp** a document, of course while time-stamping Alice will have a secure hash, cryptographical quality, and send such hash to the **authority.** It adds timestamp, a timestamp is just a string containing the month, year, day, time, hour, seconds. In some cases also more. This is the timestamp. Now the authority, after adding the timestamps, computes the hash of the received hash and the timestamp added. After computing such new hash the authority signs the value and sends information back to Alice.

Alice is obtaining statement and the timestamp authority associates a timestamp to the hash. This association is certified by a digital signature. You should understand the power of this approach. What you just need is having a trustable timestamp authority. This is a **general problem.** Actually, we saw several types. All digital signatures are requesting the usage of a public key. When I verify your signature I need your public key. This is the **weak part.** How can I be sure about it? Who is providing to me your public key? Is it just you? Or maybe the attacker can try to send a public key to me. We will see soon there are some libraries very easy to be used from command line and you generate a pair, public and private key for DSS or for RSA.

Let's assume I want to make an attack, I write a document, I want that this document appears to be signed by **some other part**, I generate my pair, I make the signature and then I take the public key, I go to the victim and I say this is the public key of him. Trust me. Check the signature, so that the victim will check the signature using the wrong public key. It's not wrong, is the public key associate to the private, the verification will say ok, the problem is associating the identity of the signer to the author. This is the weak part, where attackers are strong, they try to cheat about the identity of the owner of a key. There are some mechanisms for certifying the owner of a public key. There is a standard, we call **digital certificate**, there are entities that are certifying the ownership of public keys.

This is a very **practical problem**, in Italy there is a list of certification authority, managed by an agency of the government, the name is changed, today is **AGID,** in the website you will find a list of certification authorities that are trusted for Italy, because Italian regulations are saying you should setup your organisation, your operating manual, submit your manual, so that digital signatures generated by the trust of certification authority will lead to legal values.

Random numbers

Question time. We talked about symmetric encryption, then we talked about data integrity and **MAC codes**, HMAC, hashing. After that we considered the **public key encryption** and also described what is called the **public key infrastructure**, and we saw some usage of public key for confidentiality, but especially for key exchanging, in addition to digital sign. We concluded our remarks about public key infrastructure addressing one of the most important issues, where attackers try to get some advantage.

I want to remind you this important issue, the association between a **public key and an identity**. This is the weakest point, such an association should be certified, or guaranteed. We need trust, there are several ideas that are aiming at providing trust towards public keys. Both centralised, asking for the help of certification authorities, and distributed. In any case, the trust is important because if you check the list of the **most dangerous attacks**, cipher attacks, concepts related to trust are always in the top 10, **trust, mistrust, phishing**, is also related to trust, and so on.

It's very important you start building your **own model of trust**, by knowing the current model of trust, for finalising this concept and analysing all details we should wait for the study of digital certificates that will come soon, but now it's time to open a parenthesis for discussing a few characteristics of **generating random numbers**.

We have already seen we continuously ask to generate random number. After random number generation we will be considering several points of view about **authentication**, where authentication is understanding who is our partner, associating the parter of our communication to some identity, both in case of human and application partner. I ask you try not to loose the main path, the learning path, so that now it's the good moment for opening a parenthesis about random numbers generation, but we will back to our previous path after studying the 2 main protocols.

So, we today we consider random numbers and in the remaining time I will be giving the **next homework**, how many of you sent the homework yesterday? Let's address the main points about random numbers. In many cases we want random numbers, let me make it clear that when you want a random number because you are **testing your software**, you are making some **test for correctness**, measuring performance, you will be generating random inputs. That is another problem.

Random numbers we are needing should be showing **higher qualities**, if such qualities are not present the attacker will have successful approaches to **break our security**, think of the case you have the best encryption software. Unbreakable. Now you generate your key, what if your generator is weak and the adversary can predict the key? You understand the robust encryption software becomes **useless**. The weak point can be the **random number generation**. This is why we should consider this problem very carefully, in the history of attacks and vulnerability there are so many examples of issues related to a **bad generation** of random numbers or predictable random numbers. We need to understand the requirements, what we can do, how to approach the problem in a robust way, so that we are not introducing weaknesses inside our framework.

In the **DH key exchange** Alice is choosing a random number, her private key, Bob will be doing the same. In many practical cases both partners will be using the same tools for generating random numbers. Even if this is not true they use different tools, the weakest point of DH is guessing one of the two numbers. If the adversary is able to **see all communication**, this is the basic hypothesis, and one of the two generators is weak, ok, if the adversary is able to predict the next number he will be able to understand the next session key. I want to make you aware about the more general problem, the general problem is this, how to allow 2 different parties to have a **secure communication** if the 2 parties **never met before**? They have no possibility to share information.

In cryptography people are using **pre-sharing.** It means sharing in advance, before you actually need to make your communication, you are already organised because you share your secret information. In the general case, when you start your conversation, and you have no pre-sharing, how can you start **communicating in a secure way?** This is a very important problem, not easy to be set. It seems to be a good step in this direction using DH, you can start exchanging public information and then you set a **session key** and you can encrypt sensitive material using a session key, you can use also other keys, but the problem in DH is vulnerable with respect to the **man in the middle attack**, if you are not sure about identity of the partner you may be triggered in generating the session key agreed not with the real partner but with the attacker that is the man in the middle.

All these problems are addressed by using in some ways random numbers. I want to anticipate the final remark we will be doing at the end of our classes. There is **no real secure approach**. There are very good approaches, none of them is able to guarantee real complete security. Still we are not able to do that. What does it mean to have a random number generator? You see here two acronyms, **random** number generator, **pseudo random** number generator. What is the difference? It's related to the fact that if you want to discuss **what is randomness**, you may talk long time not so easy to define in a way not making using any mathematics what is randomness.

Intuitively randomness is often very **present in the current nature.** So, there are some physical quantities depending on some parameters we can't control or predict, you can think about temperatures or small particles or other unpredictable phenomenal. And so on. They can be considered as **source of randomness**, there is no practical way for determining the real next temperature for instance. Also you should be aware of some **limits** of this approach. I can't predict next temperature, but I can bound the temperature, I expect the temperature tomorrow will be not under 0, not higher than 30 degrees. Is this a small interval or a big range? All is relative, depends on the **number of significant digits** we are using. If we are using 20-30 significant digits it's a very small range, all possible values are not so many.

You understand that, so we now have to keep into account the power of the attacker and he's able to use to try **all possible numbers** belonging to a range. You need to make the range almost like infinite to make the approach secure. There are other practical problems in getting random values for the nature, because you don't know when they are available, how many bits of randomness you can get at any time, also in the way you get random information you are maybe introducing some information, you are using some tools for measuring the temperature or whatever, you are using human designed and it's introducing something.

Of course computer scientists considering the approach of using **n algorithms** for generating random numbers. An algorithm is like a **black box** and you give some input to the black box and get some output this is completely deterministic, even randomised algorithms are really they are deterministic, you know what is a **randomised algorithm**. How can we define it? It's like an algorithm taking some input for solving some problem and also taking an extra input, this extra input you can imagine the process of flip a coin, you get 2 possible results with probability 1/2. The algorithms are taking an extra input that is string of bits **you don't know in advance** but of course if you have the same input the output will be the same because an algorithm is deterministic by definition. Is it clear? The problem is how to get something that looks like randomness from a deterministic algorithm? This is a very tough problem. This is why in order to be correct we talk about **pseudorandom number generation**.

The best we can expect is being able to generate a sequence of bits that **looks like a random sequence**. Since the algorithm is working in the way we have described, completely deterministic, we need some initial input for initialising the state of the algorithm, for obtaining the output that is unpredictable. I have the algorithm again black box approach, if I give some input to the algorithm it produces a sequence of bits, and it will look random. You understand it's important the adversary is **not knowing the initial input**, otherwise you give the seed to the adversary, you give all the generator. In this case the seed should be **kept as a secret**. In some cases you give 2 inputs, a seed and a key. A seed is just an initial value, the key is what we call secret key, symmetric encryption is using secret keys, we can try to exploit the power of symmetric encryption for obtaining random numbers, let me open a very small parenthesis.

If you try to **zip a file** you obtain a zipped file. If you try to zip a zipped file you get another file **not really shrunk**, more or less will have the same size of the zipped file. You are already aware about that. What does it mean? The possibility to compress a file is depending on the nature of the file, there exists files you can't compress. The output of some encryptors is looking like what? If you try to analyse the statistical properties of the output of an encryptor you have a result that looks like a compressed file, you can't compress it, I don't give a theorem but we expect a good encryption algorithm gives you a **output you can't compress**.

The file you can't compress is no offering properties for **further compression**, higher entropy, chaos, this will be looking like randomness. In practise what we are talking about, random number generator should use some measure of some physical parameters, not practical, very difficult, we use random number, pseudorandom number generators, they are **algorithms**, we initialise the algorithm using some **secret input**, that should be kept as a secret otherwise the adversary will be able to predict the same secrets.

Again I want to remind we are not believing in **security by obscurity**, our random number generator or pseudorandom is well known and its property we study and somewhat **approved by the community**, so our security is based on **secrecy of the key** and the initial state of the algorithm. Any question?

I want to say that we are meeting the same problem as before, why? What we told Random number generators are not practical, so we use algorithms, ok, pseudorandom, ok, since this is an algorithm we should be able to generate a **random seed** for initialising the algorithm. Again you need some random input so how do you handle this problem? Random input? It looks like the same problem. Maybe you can afford a smaller input, but if the input, the seed belongs to a **range too small**, you understand the adversary **tries to generate all possible seeds**. This is making hard the problem.

In order to introduce the process of generating numbers looking like random numbers it's possible to use some **extra information** like system data, data coming from the environment, the computational environment we are using, extra information, external source of information, all of above, I mean several sources you can mix several sources, try to use maybe hashing functions for obtaining strange information from this external source and so on. We want a possible **long sequence of numbers**. I want to point out that in cyber security it's not sufficient we have a random number generator of high quality.

Good randomness, **difficult to predict**, ok, this is well come of course. Is this sufficient? **Not considered so**. Of course this is necessary. We want another characteristic from the random numbers we are using. Attackers are always attacking and they try all possible attacks. What happens if **an attacker finds a session key**? For some weakness he's able to find a session key the result of our random number generator. So if at some time the adversary knows the output of the pseudo random generator what is the power of such an adversary? If the algorithm is generating the next random number by using **only information based on the previous one** the attacker will be able to predict the next ones.

Also, the compromised system should allow the attacker to know **all information used** by the system but not future information. Not even the previous random numbers, because the previous ones are been used for generating keys, we don't know if some keys are still used, we need another important property, even if the adversary manages to find random information generated, he should **not be able to predict** the next one and not even to know the previous one. Do you understand the importance of this property? Not related to the quality of randomness of the bits. This is **another property**.

When we are starting generating the pseudorandom numbers we need a **seed** for initialising the state of our generator, how can we establish random seed? You see here this sequence of lists of possible sources. Information coming from disk, fragmentation, free space, other properties. Other information coming from the OS, like the input output process, user information, what is now user space? How many windows open? What is doing the user with keyboard, mouse? And so on. Other information related to the network, because of course what happens to your network adapter is looking like unpredictable, how many packets are arriving, what is the time adjacent packets and so on.

These are all sources of information **looking unpredictable**, for our purposes they look random, remember we need many bits of randomness, if information provided by the system is providing I don't know 10 bits of randomness it means the adversary can generate all possible combinations. This is why if you want to initialise your random number generator, sorry it's too long say pseudorandom number generator, I keep saying random generator but **I'm talking about pseudorandom number generator**.

In many cases the **current time is not good** as a source of randomness, not sufficient. The current time is providing a very small number of bits of randomness. In addition if the adversary is knowing we are generating a random number using current time as seed, the number of bits is very small, the adversary knows what day is today, so he should just guess how many seconds and some extra digits, one to extra digits, no more. So the number of the **possible attempts is very limited**, a small number. It's good to keep into account such an information, it should be mixed with **other sources of information**, usage of cryptography, I mean encryptors, decryptor and hashing functions is well come to produce a high number of bits, random bits.

If you check for attacks to random number generators you will find very nice stories, I just mention some well known vulnerabilities, in the last 20 years in many cases people use as source small randomness, an integer number, having a limited number of bits. You understand it's very easy for the adversary to **try all possible seeds**. About using current time, you can easily compute how many attempts are expected for the adversary, if your clock granularity is 10 ms if the attacker knows the current hour, not knowing the minute, this is providing a small number of different choices.

In other cases very **funny mistakes** like not keeping secret the seed value, if the seed value is not secret you can use no secret at all, no comment about that. In particular this type of issues has been met many times, not because people were stupid, saying ok this is the initial seed, but because the same information was provided in some header of some message, according to some protocol, the same information used for initialising the random number generator was **provided as plaintext**, is standard to provide the complete timestamp when you are generating the packet, this means that the adversary can consider all plaintext information in the protocol as initial state of the random generator.

Also there was a **very significant vulnerability in Netscape**, one of the first versions. This is the portion of code that was used at that time for managing the generation of random numbers. The function generating random numbers was a function like this one, **x = MD5(seed)**, the result of an hashing function, run over a seed, the seed is manages like a counter here, this is the value returned by the hashing function. If you just keep hashing some value you need to introduce some variability, otherwise even the adversary will be able to do the same.

So, you see here a few lines of code for obtaining what is called a seed, according to original approach Netscape was using the current time, and it was also using such conformation like the pid, some superficial trivial analysis. So by using a strange function looking like this one, this is a function just computing a value from known values, this is **not significant under the crypto-graphical** point of view, they were computing 2 parameters, a and b, by using process ids, and time, mixing such information in this way and obtaining such a seed by the hashing function own as a message digest, md5. This is the way for **obtaining the initial seed**, and then they just use the same hashing function on the seed and so on.

In this approach there were two vulnerabilities, one is the initial seed very easy to predict, the second is md5, that is **broken hashing function**, at that time not completely broken, today it's completely broken. Just for providing some information, if I'm not wrong we are just not giving any details. **MD5** was employed since the 90s and this is the size of the output, **128 bits**. Just looking at this number today we understand it's weak, small output size, it means it's **vulnerable to the birthday attack**, because 2^{60} is considered to be very practical. Actually today we know the function is **not collision resistant**, not suitable for practical protocols we need to employ high level of security.

The philosophy used to design md5 was very similar to the philosophy for designing SHA-1. If you check this drawing here this is the initial state of the algorithm, performing many rounds of operation, intermixing values by simple rules, using xor and similar operations obtaining the next step. The md5 was proposed just few years later first collisions were found and today, until 10 years ago researchers were working on proving the vulnerability of md5, most important result was obtained 10 years ago, some researchers were able to **create a digital certificate**, document associating an identity to a public key.

In order to do that some digital signatures are needed, based on md5 are hashing function was they were easily breakable. People are still using md5. Still today the experts while doing **digital forensics** they extract information from your disk and to prove data integrity they offer md5. Netscape was **strongly attacked** by many.

Because you remember the information used for generating the next random numbers. Even if the attacker is not exploiting vulnerability of md5 there are **vulnerabilities in the generation of the seed**, if the attacker knows the seed he's able to predict all random numbers. If the attacker wants to try to attack **an host using netscape**, since the source code of netscape was public, they can try to guess the process id and parent process id, is this difficult?

Ok, the parent process id in many cases is a very small number, the parent in many cases is just the process activated at the boot of machine. You know **UNIX machines** are complete under the point of view of the tools for making operations and operating over the internet, in particular there is a complete system for running the service in every UNIX machine. One technique very nice for the attacker to obtain precise information about the process id is ok, let me **send an email** to that machine, that machine is having the software for having the email, let me connect on the correct port and simulate the sending of email, let me just **generate a name** that is not reasonable, I will be sending email to abcde0123 etc. ok, the user is not existing, the daemon used by the machine will be generating a reply automatic saying the **user doesn't exist**.

That will be studied in another course, **web security and privacy.** But in the automatic generation of headers of email there is an information called **message id**, this information is generated by the system to associate a unique id to the current message. In order to associate a unique id the daemon is taking information like the host name, the internet address, is using the current time and process id, so all this information becomes into a string, present in the reply, so the attacker sends the email, gets the automatic reply **showing the process id**, it's an upper bound, it's a process serving the email.

The process id of netscape is an upper bound of parent process id, this is making the **range very very small.** The attacker was not so happy that the internet task force decided to change the standard for generating message id associated to email, other information are **not exposed to attackers**. Even without this attack, the number of bits of randomness present in this approach is very limited because you have 2 numbers, 2 integers, even if they have **15 bits of randomness** the 2 numbers are not having the bits just because the parent process is much smaller of the process id of the actual process. Even if you admit 20 random bits for the time you get 47 bits of randomness, a small number. Then you can hack the process id by using the email trick.

There are other funny stories you can check, this is very funny because 10 years ago the **Debian distribution** was going to release a new version, you know the open ssl package, a library important for making available to the user all the cryptographical function, the library was checked and after a call like that, you know the C language, you can pass parameters by address or value, and the purify tool, a debugger, was complaining, for this line of code the purify tool was saying warning. You could have **some variables not initialised**. The analysis of code was showing this was not really true. The person running the tool decided to remove this line, so that the generation of the **seed for the library was broken** and for some months the library was using a broken security because the random generator was broken.

There are lot of mistakes that can be introduced, we should just be careful. There is an interesting literature, you can check in the web a lot of **folklore.** This is written 10 years ago by an important researcher. To make more significant what we are talking remember that for many years the national security agency made actions so that **weak random number generators** were approved as standards. Why? Because it was easy for the agency to break the security of protocols, guessing sessions keys and being able to decrypt all the information.

More technical details

They defined some criteria for estimating quality of pseudo random number generator. They introduced **4 criteria** for measuring the quality. You can define them under a mathematical point of view, actually it's a list of very intuitive requirements, have a look to K1. A sequence of random numbers with a low probability of containing **identical consecutive elements**. A sequence of identical elements is called run, the K1 is we want low probability of having runs, repetition of same bits. K3, it should be **impossible** to determine by observing a subsequence of generated information previous of future values in the sequence. This is another requirement not related to the output of the output, related to the robustness against attack.

There is a symmetric requirement, asking that it should be possible for the attacker determine by knowing **the internal state of the generator,** means knowing the values of all variables, knowing such information should be impossible to determine previous values of the sequence. If you know internal state you can simulate the production and you can **predict next values**, unless extra input is provided.

About criteria K2, this is a **list of requirements** making reference to well known statistical test. Among them there are a few requirements, we would like there is more or less an equal number of 1 and 0s in the sequence. 2 test about the frequency of runs and long runs, when length is more than fixed parameters. There are **indicators for measuring the quality** of random number generators.

When you are formally defining such requirements for the pseudo random number generators you are requesting strong properties to your algorithm. This is leading to the definition of what we really need. We need not just a pseudorandom number generator, we need a **cryptographically secure** pseudorandom number generator. What is the requirement for this security? Ok, all the requirements of any ordinary pseudorandom number generator but since we want to have the extra quality of the security under a cryptographical point of view.

This means **you can't predict** next values with a polynomial time algorithm that can be successful with a probability different from 1/2. The other quality we are requiring is the resistance to the compromising. It must be possible to reconstruct the string of random numbers **previously generated.** Of course we understand you can't go back, but can you predict? This is why it's a good practise to give an input to the random generator at **any generation**. If not providing extra input at every generation the adversary will be able to predict next bits after the compromising.

There is no polynomial time algorithm that can make distinction with two outputs with probability better than 1/2 a **truly random sequence** from a pseudo random. If you construct your generator and try to analyse the output of the generator by comparing the output to a truly random source, wherever you have found a random source, there's no polynomial time algorithm able to make a distinction between the two outputs with probability better than 1/2. This is a **very theoretical requirement**.

It's an open problem, if you want to stretch your mind on some nice exercise you can consider this open problem. Is there any way to distinguish an output of a high quality pseudorandom generator from a truly random sequence **without knowing the algorithm** and the state with which it was initialised? Theoretical open problem, you can study, this is good material for a master thesis.

Now we can conclude with a discussion about random generator by showing **some examples** of random generator. This is a random generator that has been employed for a long time, still today the architecture of such a generator is considered to be good, you have just a counter, incrementing mod something. A limited number of pseudorandom number, determined by the mod. The approach is **encrypt an integer number,** the encryptor will be making use of some encryption key and this is just the output you see from the encryptor and this is the next random number. You encrypt one, two, three, and so on.

You can even decide to **make known the initial seed**, the security of this approach is based on the symmetric key used for the encryption, right? Also good the fact that for next generation you make an extra encryption and you need an **extra input** that is secret key, even in the case some generated number is compromised the next number will be unknown, as well the previous numbers. You can analyse this simple schema, from the schema you understand some properties, how many numbers can be produced from this schema?

You may start from 0, when you get N you restart. You understand there is a **period** in the generating values. In your opinion should this encryption make use of any **mode of operation?** Do we need to use some mode of operation to make encryption? It depends. When we need a mode of operation? When in input for the encryption is larger than 1 block, right? Ok, how large is this input? It depends on the mod N you choose. If you choose N **so big** so that you can exceed the size of input, ok, **you need some mode of operation**.

This is why very often people just choose a **very small number N** like 64 bits so this will be the size for I mean, 64 bits is the number of bits to represent N, this is the size of input for the encryptor, you will have a period equal to 2^{64}, is this good enough? In many cases yes, otherwise you can reorganise your generation. There is an approach based on RSA for generating random numbers, have you ever heard about **token generated** by a generator distributing such tokens, they are based on RSA, when you have to order a money transfer by internet you use a token generator, based on this generator here.

Today there is a difference because today the **modern token generator** are also keeping into account the time, they have a clock internal clock, expected to be somewhat synchronised with the official time but the initial idea was very simple, you setup the RSA algorithm by deciding the prime numbers p and q, you compute the product, you choose integer keys such it’s co-prime, this is just the beginning of the algorithm. So that you can initialise some values z, then you keep generating, encrypting by the RSA algorithm the value of z, you get the next from the previous one, many encryptions. You can change the seed whenever you want. You can decide your policy about the employment of some seed generator.

This is **another standard generator,** standard ANSI, it’s based in triple des, setup with EDE approach, encryptor, decryptor, encryptor, application of 3 times of DES. The input is a seed, 64 bits, an integer m, the key for triple DES and some date and time. The output is a sequence of N strings, you can use your triple DES **over the encoding** of time, then you have some cycle generating the values you want computing triple DES. This is providing the next number, then you update the seed using triple DES applied to the last generator number xored to the initial seed. This provides a **good sequence** of integers.

There is another generator, the approach is **choosing two prime numbers** p and q that are offering such nice property. N is the product of the 2 prime numbers, choose a random number s co-prime to N and you can build a sequence of bits,

$s^2 \mod N$ is providing an initial value, you have a cycle providing random bits and you see at any iteration this cycle is offering in output a bit under the point of view of the description this is not proper, this is **not really return**, this provides an output. You compute square mod n, after that you consider **the last bit**, to be returned. Then you obtain the next value x by squaring. This is **very fast.** The speed of random number generator is a quality very often important, because obviously if you want to use an encryptor for your generator you can do that, you will get an high quality generator but you must remember that random number generators are used when two parties **are setting up** a communication.

They need to generate random numbers. If there is some log operation the user will feel some lag, some delay not good under a practical interaction. In some cases you don't want lag, you want a very **quick generator**, still offering high quality, like this one, maybe the best. I have already mentioned that the library offered by modern variant are offering random generator but they are not having a cryptographical quality, if you want to use the random class of java be aware that is **not cryptographical secure.**

Last slide is the following, why this generator is **not good?** Easy to understand how it works. Using a **good hashing function**. If there is a compromission and the adversary manages to find one generated key he will find the next generated key.

In the last exam I asked the students to valuate the quality of some generators. First you have two cases, the initial seed **public or secret.** So you have 8 cases. A public seed is a mistake whenever you don't have extra input. Otherwise you can afford to keep public your generator. The first and second generator are the same idea, hashing twice is not introducing any new strength in the generator, this is just **double hashing**. The third and the fourth are more interesting, here we get the next random number by encrypting the previous random number using some secret key x. You encrypt by a secret key, it's a good choice, we already discussed, you understand here you can make the initial seed **both public and secret**, no difference. This is the most complicated one, I just invented it so it's different from the other proposal.

My idea was ok, you have the initial key, use the initial key x as a number to be encrypted and use as official key the result of the hash. You want to generate the next i+1 random number, ok, **take the seed,** hash the seed many times, i+1, after hashing the seed many times you get a string, use it as **secret key** for encrypting the secret key. You have the contribution of a secret key, kept as secret, you can have both public or secret your seed, the result will be good, because he won't be able to predict next keys.

Homework number 2

CMAC: Cipher-based MAC. Homework on authentication tags. Deadline 9 November h 11:59 pm.

We have seen 2 approaches, CBC and hashing function, the strongest approach based on hashing functions is probably the HMAC. This is opening today.

Description of the homework, I ask you to study, to find documentation about the **CMAC**, it's another approach, I have **not mentioned in our lectures**, this is a variant for obtaining authentication tag, CMAC is cipher based message authentication code, you can consider CBC MAC like one instance of CMAC, actually CMAC is more secure than CBC MAC and it's based on the introduction of **some encryptor** for computing properties over your authentication tag, there is an official documentation, here you see the link, a **nest document**, an official standard, what I ask you is **describe the CMAC as a MAC algorithm**, you should use your own words, it's easy to copy wikipedia or just take some text from the official standards, I want you describing this approach using your words but they should be aligned to the methodology we have used to describe message authentication tags. We considered several parameters, using some key, no key at all, I want you considering this CMAC like a **general MAC algorithm**, consistent to the approach we took some weeks ago.

You are expected to **describe the CMAC**, to **make a comparison** between CMAC and HMAC and in particular we are interested in comparing **efficiency, the security, the usability**, last but not least find some **official reference for java** or C or C++ for using CMAC libraries, I mean open libraries. This is just a small requirement for the next homework.

So the description of the homework is, you see the title, don't forget to use this title while delivering your contribution, and 2 slides for describing the homework, just **addressing the CMAC** and making some comparison between CMAC and HMAC. Any question? Have a nice weekend.

Small exercise (exam)

Any question? I'd like to make a small exercise. I'm thinking of problem I gave in the **last written exam**, we can discuss about this problem, so I want to share with you. In my problem I asked, imagine the case you are downloading an application for your smartphone, the download is made from a market PlayStore, AppleStore, you can imagine you are downloading the app from a trusted repository, the problem I'm focusing is the following:

The organisation who developed the app of course is providing the **backend**, you can imagine the app like a client server application, your smartphone will contact some server from some remote side. And they have the following concern, they want all users using the **original app**. Not any compatible app. You understand that somebody could develop some different app but compatible with the protocol that is specifying how to exchange messages between client and server. If you know the protocol you can write a compatible app. The organisation wants to make **authentication of the app**, not of the user, first authenticate the app, it means recognise **the app is the original one** downloaded from the market, PlayStore or AppleStore, the official store.

Second, if ok, you will continue by **authenticating the user**. This is the global setting. We are not going to look for a perfect solution, the attacker can be very powerful and run some tools to support reverse engineering. We are not yet able to design a very strong solution but we are able to design a fair solution. Since we have just discussed how to try to **digital signatures** I want to suggest a possible scenario and I give you some hint and I ask you to think about this problem, it's a very modern problem. You can also make some business about your idea. I am making some simple remarks. first, ok I can accept the market where I downloaded the app from is trusted. I accept that apple store, google store they are trusted.

When I'm downloading the app I can imagine the server in the store is able to **generate a pair of keys** public and private key to be associated with the app. What app? With the particular instance that some user is going to download, for any download of the app a new pair of public and private key. The smartphone is downloading both the app and the public and private keys. The idea is that the app is also **knowing the public key of the server**. It's built in the app, so that when the app starts working it can start some handshaking with the server making authentication based on the private key, in order to do that it's sufficient that the server while generating the pair of keys, private and public, is giving the pair to the app and it's also sending the public key to the server, so server gets a list of **all public keys of official clients**.

This means you can easily design some exchange of messages where the app is authenticating itself just by providing some hash of the code encrypted by the private key, so that the server can easily check by using the correct public key. No need of certification authority because the server from the store is trusted, so that both parties receive the keys from the server, if the connection is secure **no need for a certification authority.** As an exercise I suggest you to consider this setting, you can easily design some exchange of messages, this scenario is not completely robust because if the attacker is taking the app and is able to make **reverse engineering** of the code the attacker may find the private key, maybe not stored as plain text, it's encrypted by some other key, but some other keys of course, if you encrypt the private key by some other keys the software should be able to use these keys so you may try to hide the correct key in the software among the part in many cases there is the good choice of using as a key strings **extracted from the interface** between the app and the user, because such an interface is present in different languages people can switch languages, so there is some schema for extracting information so that you can hide the key. There is **no completely robust solution**, to be more robust you need other mechanism.

Authentication

It was a good moment for suggesting a problem of authentication, we are starting the part about **authentication**, we will see several techniques for implementing authentication, you will have immediately lots of question about authentication based on **password**, just be patient because before talking about passwords we need to talk about other techniques, the password one will be the last one, a very stupid technique even if it's very popular.

We start by some scenario based on ok we first give an introduction to the topic and then we start by considering a scenario based on challenge response. So, we start talking about authentication. It's a process you know very well, you need to **authenticate in many cases.** Alice needs to show the identity to Bob because she wants a service, and Bob will give the service only to authorised people, Bob wants to know wether Alice is really Alice, if she belongs to the list of people having the right to use the service. They are just two users of **dating website**, they want to be sure there is really what he meant by writing the profile, you can imagine many different scenarios. How to be sure about the identity of some part? There's another question.

What is the identity? The digital identity is a very complex concept, you know that we are not really covering this topic because it's related to the privacy topic and we will be considering such topics in the web security and privacy course, in short what is a digital identity? You can consider **like your credentials** for reading your emails. You are a user of an email system registered user, ok that's a digital identity. Is that associated with some physical person? What is its identity? Analogic identity. We don't care in this moment. What we care is ok this digital identity is **well defined**, because when I am registering on facebook I need to give my email address and also for other social networks, in other cases you can **register to some services** by providing your facebook identity because facebook has become so important in handling digital identities that it can be used by protocols.

It happens that it becomes a standard almost a standard. So, for now a digital identity is just what is defined according to some registered email, the importance of protecting the emails is so **critical**, still today many procedures from recovering lost passwords are based on email. Any adversary controlling your email can trick Facebook or other networks to issue a new password requirement, I can simulate a lost password, facebook will be providing an email with a link, if email is broken I can steal all digital identities. There are many possibilities. There are special applications for supporting the process of password recovery and so on. Still the people using such smart procedure is very small set, in the average age of such people is very low.

So, Bob wants to be sure about Alice, so Alice can be just a pair, I mean, just an email address, associated to a real person that has been verified. In other case we want first and last name of Alice, **something more**, we will see in order to have first and last name we need something like a digital certificate or global service, like the **Italian SPID**, a digital identity allowing in Italy to sign in in public administration portal. Not completely supported but the process is on going. Have a look. We discussed that the attacker can be running several types of attacks but there are 2 special attacks, **man in the middle and spoofing attack**, the 2 attacks somewhat overlap because in some cases it’s not easy to say they are different, in some cases they are the same attack.

For what concerns authentication the word most frequently used is **spoofing**, means I’m cheating about the identity of the originator of some messages, an email message or a datagram, sending a packet or whatever, at different levels of the stack ISO/OSI. There are many spoofing attacks, so that the intruder can try to impersonate Alice once or many times. We want to **prevent** such type of attack by using **authentication**. On one hand we need to authenticate our communication partners, we need to make this usable, I don’t want to authenticate my partner for every message, otherwise there will be a big overhead and I don’t like that. We have to **design carefully** what we are doing dependably on the application domain. From now we are not making distinction between digital and real identity, we will add some remarks about that when starting digital certificates. Of course you can consider some simple cases where in practise in human interaction you can recognise people, the face, the voice, or some other practical property you know very well, or you can use a badge for authenticating your identity.

We are considering the authentication problem in a general framework, you can authenticate yourself because you know some particular information about your credit card, described by some digits that should be kept secret and so if you know part of this secret you are the correct owner of such an information. There are 2 interesting cases in this general framework, where a computer is needing to **be authenticating from another computer**. Is that the real computer I am thinking it is? I try to authenticate a computer or a software, not a user, and the example I started today is the **authentication problem**, I want to authenticate a program running on the smartphone.

Another special case is the case where somebody is using a **public workstation**, used by many different users, imagine internet cafe, you can go there, you pay a few coins and you get the workstation. It can be running in very smart operations but you can't expect that workstation is storing your private key, how to manage such authentication when accessing a public workstation? If the workstation is public you can expect only the user is storing information in his mind allowing to run the authentication in a successful way. Under a philosophical point of view we like to use the **closed world assumption**. When running the authentication if the process is not completely successful you can assume it's failing.

The opposite is the **open world assumption**, if you don't know it's not meaning that the false identity. In this case you want to be secure and you prefer to use closed world assumption, and many practical people don't like to use such because too philosophical, many people just use words like black list and white list. **Black list** is a list of forbidden behaviours, all that is not explicitly mentioned in the black list is allowed. **White list** is the opposite, everything is forbidden except what is mentioned in the white list. White list is more secure than black list but more limitative of freedom. In the department there are some professors doing research on security and they know that security in order to have security you have to **accept limits to your freedom**, all professors including myself are saying ok I don't want any limit, I'm aware about the fact I can be attacked but I don't want any restriction to my freedom, we are not completely consistent. We are just humans.

In many cases when you are considering the problem of authentication your environment can be considered as closed. You can think of the network of some enterprise that is having many employed, many people and all of them is authenticate when in the morning they reach the office, they make the boot of the computer and should authenticate in the domain associated with enterprise. They are authenticating themselves as human users within a **closed environment**. Since this is a very setting very frequent is interesting to study this particular case, authenticating users within a closed environment. The larger enterprise, more important such a secure authentication and typically enterprises can afford spending money, invest money on such type of security because they understand the importance of the cyber risk, in other cases small enterprises, authentication will be very weak.

When you use a closed environment you consider, somewhat deciding you have one special server here it's called a **third party carol**, a trusted server, one special host in the network massively protected by many measures, logical and physical, to protect such unit server and this server can support the authentication of several users, we will see that **kerberos** is a very secure environment with a single approach it's based on a trusted third party. So in this setting it's very very common that you have the trusted server, you have Alice and Bob belonging to the organisation, they need to use services, the attacker is also belonging to the same organisation having its credentials for accessing the network enterprise, this is very common, **the attacker is an insider**, with this setting the insider is having many powers with respect to powers had by some external attackers.

Is very frequent that in important organisations with high level of security **attacks are coming from inside**, not outside, in the top 10 you will always read insider. When authenticating a party is this human or a computer? Is there any difference? In practise there are several differences, you easily understand a computer can store a very good secret, a very long key, password having thousands of digit. A computer can easily store such a secret information and run cryptographic operations. Humans they are not able by definition to store in the brain long keys very complicated, they at most they can afford some passwords, what is the difference between a **password and a key**?

First is that since humans are very weak and very lazy they choose as password words belonging to some dictionary or simple variations. Can I use such a password as a key for running some encryption? I use a very strong encryption, AES. I need a secret key, can I just invent a password from some dictionary of the right length? Then use it as key for encryption my information? I could do that, **completely insecure.** A more secure is: I invent a password then I use a hashing function providing what is the hash of the password, if you start from a weak password or hard **it's the same**, the result will be an hash having a length, you can use that hash as key for running the encryption. Not because I want to introduce a standard, I want to introduce the idea of good information derived by a possibly bad password, you can derive good information from a password.

There are important libraries such openSSL that are allowing you to choose a password and to use it as a seed for generator, pseudo random generating a secret key, this is good enough. If your password is stupid because you choose your birthday the attacker can guess it very easily. We understand an important difference between authenticating a program or human is that the computer can use very complicated secrets and run very tough operations, humans start by passwords or by passphrase. A password, when they ask you to generate a passphrase, they are suggesting you to choose many words from a dictionary and concatenate such words so you get a phrase. We will talk about that.

We want to focus on the problem of authenticating people. You can choose between different approaches, you can base your approach on **what people know,** such secret, some password, on what people **are having**, a smart card, a smartphone, they get a code on it, any of you is using **2 factor authentication?** Good. You can base authentication on who you are and who you have.

Talking about passwords

Ok, humans can afford short passwords. Sometimes they are **easy** to be guessed. Computers can afford long password, store in an encrypted way somewhere. There is some authenticating process for decrypting password. They also can use **one time passwords**. One of the first attack is the so called **trojan horse** running the trick against the user. **I did that**.

In the case of biometric authentication, you can examine the retina, analyse the voice, handwriting and so on. In many cases there are **false positive** and **false negative.** What is considered safe today is **pairing** that type of authentication with other types. Fingerprints change in time, the attacker can record your voice, and so on. Other biometric tools for authentication, the style of typing your keyboard. It's more robust than you can imagine. In particular his speed is ranging in a very short interval, after some time by using some machine learning approach it's possible to **learn the way** a user is typing the keyboard, this can be a good factor. Another possibility could be when you are recording your signature on your tablet. **Analogic signature**. What is the tablet recording? Not only the shape of the signature, also the speed, acceleration, strength which is determining the pressure of the pen on the tablet, many physical factors that are considered all together and from this collection of information is possible to run some good authentication. When you are designing systems you need to **prevent the false positive**, that is the real aim. Also good to prevent false negative but it's important to have clear in our mind such parameters, such options.

We start considering some real cases. They are very many cases. We start considering some important settings. For instance, a very nice setting is the one where the 2 users, Alice and Bob, are **sharing a secret key**. They can use that for making a good authentication. Another possibility is that every user is sharing a key with a trusted authority. Or when users are using public key, this is another different possibility for making authentication, you can consider separately. You can define many techniques, when in the exam I ask to you **design some authentication system** you should understand it's a design, so you can find your own solution that is good as my solution, in the reality solutions are never unique, you can approach the same problem by different techniques and get good solutions, a keyword in our learning path is the **engineering**, engineers are doing designs. You should get used at designing. Designing means understanding the environment, knowing the main techniques, main methodologies and making choices. The choice should be the **result of some analysis**.

Here you can read a list of possible ingredients for implementing authentication, a possible ingredient is a timestamp. A nonce, a random number to be used **once.** One time. In many cases such random number is used as a challenge, to challenge the other part to do something he's expected being able to do. I just generate a random number, so you understand the importance of being able to **generate a random number,** I communicate it and expect the other part is able to use such random number, according to some agreement, some action that is **well defined** and takes as input the random number. When I see the output I understand that only one person can give the input and provide the right output.

Other techniques are **sequence numbers,** for example TCP is using sequence numbers to implement two things, correctly use the packets and implements windows. I want to point out again that the ability of using cryptography for confidentiality doesn't mean authentication, **not implies** authentication. The attacker can be using cryptography but the attacker could be just an attacker, not the right person. We need to design special techniques. We start by considering the case where we implement the authentication based on a symmetric key, it means that the 2 parties **are sharing a secret key.** Symmetric encryption. We want to use encryption for implementing authentication, we should be able to understand we are not using confidentiality. We are not aiming at confidentiality in this moment, we are aiming at **authentication**, this is just an usage of encryption for authentication, if you want confidentiality you must add another layer.

So this is a setting where the 2 parties are sharing a secret key. Alice and Bob are the only parties knowing such a key. This notations means we are encrypting message M by using this secret key. Of course you don't see here what is the encrypting algorithm, we don't care. Any algorithm is good, as long it is a **strong algorithm**. Another important detail is, be careful, one way authentication is different from two way one.

One way means one part is providing information about its identity, but not vice versa. So maybe Alice is giving information about her to Bob, but if she wants to be sure about Bob's identity she will need another authentication. There are some approaches where you can implement both. So we see some examples, one way authentication based on challenge, where the challenge is a nonce, one way authentication based on timestamp, two ways mutual authentication based on nonce. We see some examples, we see some attacks, we start understanding the **new dangers** existing.

Look at this simple schema, 3 messages for implementing one way authentication. Alice is proving her identity to Bob. Such messages are being sent as plaintext, you can ask why plaintext? You can do that but we want to design something that is already secure based on plaintext. why? look, the first message Alice is presenting herself just stating her identity to Bob. Bob is challenging such party by sending a nonce, means that Bob is using a cryptographical secure pseudo random number generator, sending such a number to Alice and Alice is expected to encrypt such a number by using the **shared key**. Of course Bob can check the last message from Alice, can check it in 2 possible different ways, they are equivalent.

One way is take R and **encrypt R** by using the secret key and get the same result being sent by Alice here. The other verification is just decrypt this message and check the decryption is number R. You also understand that the 2 possible practical implementations are different, in one case you need to be able to encrypt-decrypt, in the other case you don't need to decrypt because Bob is running the same process, Bob can take R and make encryption for R. This is providing the possibility for using a good **hashing function**. Not reversible. If you are using a symmetric encryption you can use whatever.

As a result Bob is authenticating Alice, not vice versa. Recording all messages is able to collect many pairs R and encryption of R, they are all pairs plaintext - ciphertext. By using this information the attacker can run a **password guessing attack offline.** It means the attacker knows just one pair and starts generating all keys being able to give such associations. The adversary can mount an offline password attack. The **limits** are, if the password is not strong the adversary can try an offline attack and the other limit is that this is a **long way authentication**. Of course we don't need to remark some things, of course we need a good generator because if the adversary is able to predict a nonce.

This is a variant, very interesting, in this variant you see that Alice again starts with a message stating her identity. The challenge by Bob is not R but it's the encryption of R, by using the shared key. Alice is expected to provide the original plaintext R. In this case you need to use reversible cryptography, real encryption, of course the attacker can try again to mount up an **offline attack,** he's knowing a pair plaintext ciphertext. What is nice to observer is that the nonce R is having a limited lifetime, it means we can consider it valid for 20 seconds, 10 seconds, very small amount of time, after that no longer valid. If the nonce has small lifetime imagine when Bob is generating the challenge using a nonce concatenated with a timestamp. In that case he sends the message here and Alice is replying but there are 2 interesting consequences, because Alice is understanding the other part is Bob, why? Because with timestamps Alice understands that only Bob could send the message, she's receiving in the correct time, real time, and only Bob could do that. The time interval for the validity should be very small, otherwise the adversary could try a **reply attack**. Seeing a secret transmitted over the network, I don't understand such secret, I just record it and I send it again pretending to be Alice or Bob.

Here we can see an example where the authentication of Alice is implemented with **one simple message**. Where Alice is just sending the encryption of the current timestamp by using the shared key. In order to do this approach we need the 2 systems should have synchronised clocks of course, the clock can be subject to attacks. Then also we have to define the **time limit**, if you send such a message maybe the validity should be 1 second, otherwise it could be attacked. This is very efficient, in some cases also useful. This is interesting enough, it could be attacker but this is the base, you can **improve it** using extra ingredients. When Bob is the service, like a printer, and Alice should be authenticated as authorised user of the printer, this is just a string that can be sent to the printer to show that I am authorised to print. What if there are many printers? They can share the same keys. The identity of one printer must **concatenate** with the timestamp.

Two ways authentication

We want to implement a good authentication, two ways authentication, the first idea is ok let's just try **twice the one-way authentication**. In this case you see that it may be implemented in this way, first message I'm Alice, Bob sends me the challenge, Alice is encrypting some information that is ok this is a different notation, this is just the encryption of the challenge sent by Bob, so Bob is **completing the authentication of Alice**, now Alice is challenging Bob for the other part and Bob is expected to encrypt new challenge by sending the correct answer to Alice.

You see here a mutual authentication that is originated by one of the two parties but after the starting of the authentication the actions are completely symmetric. Challenge - response, challenge - response. This is a simple way to implement a

mutual authentication. Of course we can say ok many messages for implementing this mutual authentication, can we save some messages? Ok this is a compact way for implementing a mutual authentication. We can start by analysing the content of the message, if we look at the first slide what information is Alice sending to Bob? I am Alice, an encryption that is a response to a challenge and a challenge, you can **compact the message** by letting Alice sending as a one only message the identity and the challenge to Bob, Bob is sending one only message containing the response to the challenge and the challenge, you can compact.

The same content in a **fewer messages**. In this case we can obtain 3 messages only. You agree this is an equivalent to the previous case? Just saving messages. Ok, but now we should be getting used to the reflection attack, now I'm going to explain a new type of attack, a very dangerous attack. It is based on the fact that in current modern frameworks, in the case where authentications are made many times by software, software agents implementing many authentications, the user that is using the laptop is **not aware about the number** of authentications carried out by the computer, in many cases the computer is talking to several parties or just to one party only making **many authentications** for different reasons. This is quite normal.

This is making very dangerous the **reflection attack** that is based on the construction of an extra session of authentication that is not really useful, just used by the attacker for obtaining good information to be used in another session authentication. This is the schema. In this reflection attack Trudy, the attacker, is **pretending to be Alice**. Messages are spoofed, you see the first message, I'm Alice and this is my challenge R2. Bob, this is an attack based on the previous version of the protocol, we consider this protocol where Alice is providing identity and challenge, Bob challenge and response and Alice is providing response.

You see here only 2 messages of the 3, Trudy is sending a spoofed message saying I'm Alice this is my challenge, Bob is correctly responding, with the correct response here that is the **encryption challenge** and this is the challenge prompted to Trudy, if it's Alice getting the challenge she's able to respond by encrypting R1 but Trudy is not able to correctly respond because she **doesn't know the shared key.** Trudy starts a new session authentication, saying ok I'm Alice and this is my challenge, but Trudy is choosing number R1, the same challenge, the same chosen by Bob here and in the first session it was Alice or better Trudy expected to respond, so starts a new session authentication.

Bob will reply according to the protocol, **encrypting R1** using the correct key and will be providing also another challenge. Now Trudy is knowing the **correct response** to the challenge, now is able to complete the first session by providing this information as a response. The second session will be not completed and will go in **timeout**. The second session is not completed, not analysed, but the second session is not finalised, maybe this can setup an alarm, it's anomalous, Bob is expecting something. Sessions not completed are really frequent, this could happen for many reasons and this is considered normal, **not sufficient for generating an alarm.**

We can prevent this attack by some simple tricks, for instance you can setup a **challenge rule**, saying that the challenge that is generated by the initiator of the connection should belong to some set of numbers and the challenge generated by the responder should belong to another set of numbers. Odd numbers, even numbers, for instance. This means if there is some challenge generated by one party the same challenge **cannot be generated** by the other party. This is preventing the type of attack we have just seen. Another possible measure is **change the key** so if Alice is sending a message to Bob the key is the shared one but is Bob sending the message. Or whatever you can imagine, simple rules for changing the key, the effect of the encryption will be different.

The reflection attack for that particular example is not working, be aware that reflection attacks are **general schema** for running attacks, not only for authentication, the measure we are seeing here are good in these particular cases but **not in all possible cases**, we need to study carefully some cases when considering different frameworks, also in the case of authentication, we have some other framework for authentications, we will see some interesting examples. I want to remark that this schema for implementing authentication are just a beginning.

I mean that in practise we are **not really implementing this schema**, the schema implemented are inspired to that but they are also including other ingredients to make authentication more robust. It's a learning process converging to more practical implementations of the authentication. This is **another possible mutual authentication** and in this case messages are not 3 but 4. Not 5 as the previous slide, not 3 like the optimised case. Alice replies to providing the response and by providing the challenge in the same message, this is the response by Bob. I suggest you trying as an exercise to check **wether** while using this scheme **is possible to setup the same type of attack**, just try as an exercise. Keep in mind that even in this case the attacker can mount up an **offline password guessing attack** and what happens here is that the attacker will be storing recording the information exchanged here and he will be able to save pair of correct associations of plaintext ciphertext.

If the plaintext is belonging to a small set like a dictionary or derived from it the attacker will be able to **break the security** of course. Also consider where in many cases the attacker is an **insider**, don't forget this detail, in many cases the attacker is an insider, so in order to find to get to collect pairs of plaintext ciphertext the insider can **push colleagues to start conversations** with other colleagues by inventing some simple reasons, since there is basically trust between work mates people will start conversation with other people and the adversary **collects proper pairs** plaintext ciphertext. Now we can somewhat see reviewing this process of basing all the challenges response authentication in a more general setting where we introduce a **third party**.

This is a very typical scenario used by organisation and enterprises, as I have already mentioned in this case we have one particular server internal to the network that is protected and it's **considered trusted** host and very typically this is called key server or also authentication server and what happens in most cases is that every participant in the network is **sharing a secret key** with this server. So if we have n parties participating in the network there will be the same number of secret keys shared between the participant and the server. This is a very **standard setting**.

So you can consider the case where Alice and Bob are sharing the secret key between the server. All members of this network can be humans but also machines, programs and so on. Every user should be registered in order to use the services and the administrator should generate a new key for this user. The key will be shared between the new participant and the server. The goal is given the user to participate in the network we want to implement **single or mutual authentication** between Alice or Bob. Not so optional but I would consider it important is not only finalise the authentication but also generate a session key to be used between the two parties. For now what we know for generating session keys, you remember the **definition of session key**? Is a non permanent key to be used for securing some messages during some short session. After the session the key is just dropped. This is a session key. DH allows to generate a session key, also a good random number generator allows to generate a session key you can share with other parties by sending a message encrypted by RSA.

So the idea is that when you are using some **authentication server** Alice and Bob they have to start a secure conversation, they need to implement the authentication, to have a session key between them for some **short time**. We want some authentication protocol that is guaranteeing some facts. Only Alice and Bob are knowing the session key, also the authentication server, trusted server is knowing the session key, and we want to be sure about that and we want that the session key is a new key every time we start a session so it should be a **random chosen key** not used before, when you are generating a new random key you are not testing wether this key **is actually really new** or maybe already generated, the probability of generating the same key while using long keys is **very low**, you accept the risk of reusing the same key after many generation because due to the high number of keys the adversary is not able to save all of them.

In many cases the attacker is an insider in many cases, it means he is also having a shared key between the attacker itself and key server. This is a standard insider, the attacker could be an user having conversation with Alice and with Bob. I'm talking about **standard secure conversation** between the attacker and Alice or Bob, the attacker and Bob. After that the attacker can store information about that conversation in order to run **future attack**. The power of this attacker is just ok it's a regular user of the system and may store some old session keys. The limit for the attacker is ok **not able to guess the random number** generated by the parties, not able to get the shared key between Alice and the server, between Bob and the server, the unique shared key is knowing is the shared key between the attacker itself and the server, because in these cases we are considering the very important particular case where **the attacker is an insider**.

Also the attacker is expected to have a **normal computational power**, if the attacker is seeing some encrypted text, he's not able to decrypt in a short time. How to setup an authentication with a trusted server? We can start considering some schemes for implementing this authentication. We will see **many schemes** and slowly converge towards new and more robust schemes but we need to understand why we need some choices, so we start from using simple schemes, how to attack them, **patch vulnerability** and so on. Just as for learning purposes you can consider this very simple scheme.

Ok, Alice wants to talk to Bob, so Alice sends a message to the key server saying ok I'm Alice I want to talk to Bob, the server chooses the session keys and sends to Alice 2 messages, the first one is encrypted by Alice secret key, the second one by Bob secret key, the content of the message is the same, **the session key**. The server is sending twice the session key to Alice but one message is encrypted by Alice secret key, the second one encrypted by Bob secret key, Alice can't check the content of this message but she gets it. Alice is getting this message here and she's able to decrypt this message and **obtain the session key**, now she can send to Bob the following message, I'm Alice, I obtained the information from the server and this is what the server sent to me meant to be used by you.

Ok, so just Alice is somewhat repeating the same information obtained by the server sending it by Alice to bob. Bob is able to decrypt this message because it's encrypted by its secret key, finds k and can send a message to Alice, like hello Alice I'm Bob, encrypting the message by **session key**, Alice will be sure that it's really Bob that is sending this message. Does Alice need to send a similar message to

Bob like hello Bob this is Alice? Encrypted by the session key? Actually not because Alice is having a very important role in this scheme because it's Alice sending the session key to Bob, Bob is using a session key obtained by Alice and it's encrypted by the secret key of bob, ok this can be attacked, we have to **make some steps**.

You can assume the attacker trudy can spoof the man in the middle attack, this means that the attacker is able to be between Alice and the server or Alice and Bob. Imagine the conversation starts by sending to the service I'm Alice I want to talk to Bob, but **it's Trudy** intercepting the message. Now what happens, Trudy after intercepting the message can send a message to the serve saying I'm Alice, I want to talk with Trudy. The server chooses a session key, encrypted key, one time with the secret key of Alice and the second time with the secret key of Trudy, because the message received by the server is Alice wants to talk with Trudy.

What happens, now Alice is trying to send to Bob the message like I've got the key from server C, I'm Alice, this is the encryption of the session key. Again T is intercepting and after that Trudy can reply to Alice saying I'm Bob, hello Alice, encrypting this message using the session key, after that Alice she needs a session key between Alice and Bob and in reality the session key is **between Alice and Trudy.** A first vulnerability, there are many in this protocol, when starting the conversation Alice is talking to the server, saying I'm Alice I want to talk to Bob, what about the case of replacing that first message by this other? I am Alice, identity of Bob is encrypted. There is this possibility, Alice can encrypt the identity of Bob and the server will be able to decrypt, to understand who is the other party wanted by Alice.

We get a **modified protocol**, this is the modified protocol. Alice sends to the server I'm Alice, I want to talk to somebody, the identity is encrypted. The server is able to decrypt, can choose the session key and send to Alice the double encryption of the key. Again you can imagine the protocol is complete in the same way as before, Alice is able to decrypt the session key and can send a message to Bob, able to decrypt the session key. Notice that in this second case the attacker is not knowing the beginning that Alice wants to talk to Bob, the identity of the other party is **not known** to the attacker, will become known later, when Alice will try to send a message to Bob.

It's possible to run a more complicated attack in this protocol. Imagine that Alice wants to send a message to the server, I want to talk to Bob and using the encrypted identity. Trudy is an attacker, running the **man in the middle,** is able to get the message from Alice and sends to the server a different message, the message is "I'm Alice I want to talk to Trudy, this is encrypted by using the key of Alice", this is not easy, the attacker is not knowing the secret key of Alice, this is just a reply attack, we can just imagine that some time before Alice talked to Trudy. While this, the information has been sent to the server. Trudy is storing some old message and can reply some part of the old message. Trudy can send this information here as long as in the past at least one Alice talked to Trudy.

Since trudy is an **insider** it's easy for her, can very easily push Alice to start a conversation with Trudy. In this case the server will be choosing a session key and will be sending to Alice the 2 versions of the session key, by Alice and Trudy. Alice is decrypting the message so she gets the session key and sends a message to Bob, I'm Alice and this is the encryption of the session key. Trudy is **getting the message** and now Trudy is obtaining the information that Alice wants to talk to Bob. Since Trudy now understands is Bob the other party, Trudy can reply to Alice telling hello Alice I'm Bob. Trudy obtains the session key here.

We can recognise many vulnerabilities. The approach is a set of protocols, many versions. There are original versions, there are versions of the protocol based on symmetric keys, on public keys, so it's a very important approach. This is the **first important scheme** you should learn. NS. This is the first proposal and is vulnerable to an attack. Then we see the attack. This is very important, the kerberos protocol is based on some protocol derived from Schroeder. At the beginning Alice wants to talk to Bob, she's choosing a nonce, sends a message to the server, I'm Alice, I want to talk to Bob, this is my nonce.

The session key, identity of Bob, encryption of session key to be decrypted by Bob offering the same session key, identity of the other party, Bob we know he will use this session key with Alice. Alice is able to encrypt this message, check the nonce n, check the identity and check the message to Bob, this is **session key encrypted.** The content of the message is providing not only the session key but also the identity of Alice. Bob is able to decrypt, now Bob can choose another nonce.

Ok, Alice is choosing the nonce, I'm Alice, I want to talk to Bob, this is the nonce, this is the reply from the server, we know the server is replying the message encrypted by the secret key of Alice, now the attacker is in case of Alice. Alice gets the message from the server, checks information. The **session key.** Actually there is Trudy running, intercepts the message, sending to Bob another message, that is containing a different session key, K1. This session key encrypted by the secret key of Bob, this means it's a old session key that has been actually used, it's you see encrypted so only the key server can do that. It's a session key used by the 2 parties Alice and Bob for conversation. If the attacker is already monitoring the conversation and have the possibility to store such information. In this case what happens, the attacker is **able to send to Bob** this old session key.

Bob can decrypt, actually the message is intersected by the attacker, is sending to Alice the correct message, I'm Bob this is nonce, encrypted by the key Bob understands should be the right key, K', he sends to Alice but obtains by Trudy, sending to Bob the correct reply, notice that in order to do that it's requested the ability to know all the session key, so be careful, not only trudy is knowing all the information stored here, but it's requested knowing all the session key. It's possible, yet not so difficult to get such session key. This is a possible attack, not so easy, it's requesting that there is a key owner key has been **compromised**, for that key it must be known the attacker the encryption of such key using the correct secret key of Bob, after compromising a key you can use such key later.

Needham - Schroeder protocol

We start talking of the family of the **NS protocol**. For implementing authentication in this case there are **2 parties**, let's see again this version of the protocol. There are several versions, and last time we introduced this version of the protocol and discussed how to make an attack to this version.

So we quickly describe the initial version of the protocol where Alice wants to talk to Bob so that **she chooses a nonce** and sends something to Bob. The server is storing a key shared between any participant and the server, this is **secret key** for implementing **confidential messages**. So that the server will choose a session key for Alice and Bob, will send to Alice a message encrypted by the key of Alice containing session key, the nonce, the identity of Bob, and another message to be sent later to Bob containing the session key and the identity of Alice **encrypted by the key of Bob** so that Alice while receiving this message decrypts the message, gets information, checks the nonce, checks the identity of Bob, sends information to Bob, this is called **ticket.**

Bob decrypts and gets the session key and **sees the identity of Alice** so that Bob can choose another nonce N' and sends a message to Alice, encrypted by the session key and Alice can reply another message, the 2 parties implement for authentication. We already seen this attack to the protocol, now I want to see the same attack described in a **more detailed way.** We can assume that the attack is running to 2 different sessions, the attacker has recording one session and the key of that session has been **compromised**. Why the key has been compromised? Several possible reasons, we don't want to go into details, there can be many reason why a server can be compromised. So in the next slide you see **2**

fragments, a blue fragment and a red one, the blue fragment belonging to session 1, the red belonging to session 2.

In **session 1** you see Alice starts the protocol by sending all the information to the server, the server is replying to Alice by encrypting information by using the **key of Alice**, the same information stated before, now Alice is sending to Bob the ticket but in this moment the attacker is intercepting the message so that the message is obtained by the attacker. Now you can assume that later, we don't know when, later all this information is stored and in particular **the key has been compromised** so that later the attacker can use this key in order to **cheat** by claiming to be Alice sending the message, actually the attacker sends the message and is replying the same message so Bob thinks he's getting a message from Alice and he finds by decrypting this message a **session key**.

I want to repeat that this **is not effective to store** session keys because every host, every user in the network continuously makes authentication so the overhead of **storing session keys is not convenient**. In no case session keys are stored. Some keys are compromised and can be used very effectively by the attacker. In this way Bob will think that he has to talk with Alice so that he replies to the message by sending a message to Alice, the message is intercepted by attacker and according to the protocol Bob is **encrypting a new nonce** and the attacker will reply to Bob in the correct way. Now Bob is thinking he's sharing a session key with Alice, actually he's doing that **with the attacker**.

There are many possible variants of this attack, all these variants today are important just for letting us better understanding why we need to **introduce some more ingredients** in the messages two parties are exchanging. Again introduce such ingredients, timestamps, sequence numbers and nonce. There is a general methodology that can be used here, that is using in a smart way both timestamps and nonces, making the approach robust against reply attacks. Anyway I want to show you a more practical, a more important variant of the protocol, this is one of the many variants. In this variant a timestamp has been introduced and this is known as the **modified challenge response variant**. So that you see the messages here, the slide, Alice starts the conversation, wants to talk to Bob, in this variant Alice instead of sending a message to the server she sends it directly to Bob, so I am Alice, this is my nonce.

Now Bob is choosing **another nonce**, N', and it's Bob contacting the server, he says I'm Bob, this is my nonce, and then he sends an encrypted message, Alice wants to talk to me, this is Alice nonce and this is the current timestamp. This part of the message is encrypted by using the **secret key of Bob** and the server is knowing this secret key. When obtaining this message the server understands there will be a conversation between Alice and Bob. **He generates a session key** for this conversation and he's sending to Alice so remind Alice sends a message to Bob, Bob sends a message to the server, now the server sends a message to Alice, the message is containing such parts, an encrypted part containing information about the identity of Bob, the original nonce of Alice, the session key and the timestamp. The timestamp has been **generated by Bob**. Such information is encrypted by using Alice key, so that Alice can decrypt.

The **second part of the message** contains the identity of Alice, the session key and again the timestamp, second part is encrypted using secret key of Bob, only Bob can decrypt, then there is nonce N'. No need to send N' as an encrypted way, because in the last message Alice is sending to Bob **the encryption of the N'** by needs of the session key and the ticket to Bob because it's some information that **only Bob can decrypt** and understand, it is prepared by the server and it's containing the session key so that Bob can check it's the same timestamp and the identity of Alice. In this way with 4 messages we have a **variant of the protocol** implementing the authentication between parties, there is no emphasis on authentication, just setting up a session key but the so called expanded version of the protocol, this is the **last version** we see for now, it's in this picture, it's showing also providing emphasis to the last part of the conversation where is the mutual authentication.

Again Alice wants to talk to Bob, Bob is **generating a nonce** N_B and sends the nonce to Alice, it's encrypted using the secret key of Bob so that Alice is not knowing N_B. Now Alice sends a message to the server, she's sending a message to the server: I'm Alice, I want to talk to Bob, I got this from Bob, the encryption of a nonce, this is another nonce generated by Alice. The server will now **generate a session key** between Alice and Bob and decrypt the message encrypted by the key of Bob and **finds the nonce** N_B.

Now the server sends a message to Alice containing several parts of information and the whole message is encrypted using the **secret key of Alice**, and what is Alice getting in this message, she's getting the nonce she sent to the server, the identity of Bob, the session key and the ticket to Bob, information that Alice is **expected to be send to Bob**. The ticket to Bob is some information encrypted by the secret key of Bob and this ticket contains the identify of Alice, the session key, the nonce N_B so that Bob can check the ticket and realise it is the nonce generated. After that Alice **sends the ticket to Bob** and sends the encryption, now she's sending information encrypted by the session key this is the encryption of nonce N_2, another nonce.

Bob can decrypt this information using the information contained in the ticket, encrypted by the key of Bob, now Bob can reply to the message encrypting using the session key further information, a variant of the nonce, another nonce, and then Alice can send some **confirmation** that by encrypting using a session key a variant of the last nonce. If you analyse the message you can understand the last message is **not really necessary**, because after the 6th message the authentication is already implemented, there is no need to send other information.

How to analyse such protocols? This is a more robust version. This is harder to be attacked. Of course when you are implementing this protocol you write software introducing some vulnerability, they are vulnerability of the software, not of the protocol, while analysing such a protocol like this one we have to carefully look at some particular details, **how many messages?** This is important. Because the number of messages is determined some delay in operations that in many cases you know delays are **not welcome by humans**. Number of messages, how much encryption? Because ok, encryption is a good thing but also introduces **some delay** respect to the other possibility of no encryption.

This is why information not really needed to be encrypted is **never encrypted**, encryption here is not really heavy because many cases just numbers are encrypted here, only a more structured message here like this one is encrypted. But this is what we expect in every implementation of protocol of this type, we can expect we are encrypting a number, we need to integrate structure information at some step in order to reach our goal of authenticating and **generating a session key**. This is a protocol inspiring the way the **kerberos** protocol is implemented, the implementation of kerberos is based on that and now the next slide pack we will consider kerberos, and we will see under a **philosophical point of view** what we obtain using a protocol like that. With a protocol like NS we obtain a session key, authentication, this should be introduced in some larger environment, where these steps are just steps and users are thinking at running more complex operations, this is a good example now see, in the kerberos protocol the new pack of slides. Any question about that?

Kerberos

Let's talk of kerberos. Is the most prominent user. How many of you are familiar or somewhat know kerberos? The way today we see kerberos is the **B plan**, I don't know if you remember I asked if anybody would contribute **supporting me** in kerberos, nobody contacted me, I shift in the B plan, in my idea introducing kerberos by **using 2 characters talking,** exchanging ideas and needs and solutions and their discussion evolves to the final solution, under a teaching point of view, the detail are easy understood in any case so that I will just describe the steps. Ok, let's go to the point.

Kerberos is a **general framework** for providing authentication in distributed systems and it's used in network enterprise, but not only for enterprise but because in any case people are using kerberos for LAN so that, what is important to say is that this solution is implemented in a good way for **all operating systems**. Even if you carefully look at implementations you will see some variants introduced by the implementers so introduce some new ingredients in the framework just to reflect the **need associated** to the brand.

Anyway today I'd say that all implementations are fair implementations even if my favourite is the one available in the **Linux framework**, this framework is providing a **safe access to the resource** of a network, you can imagine you have a network very complicated, with many users and with many services, and the general needing ok when you are very complicated network you may want to implement some polices, for instance this is the **laser printer**, every page you are printing is having a cost of 2 cents, this is the **colour laser printer** and every page is having the cost of 1 euro. So you want to implement polices for defining what people are **authorised** to use such laser printers.

Not every human user but also not every program **should be authorised** to use every resource in the framework. There is some policy written by some administrator I'm not focusing on the way of writing a policy, that will be done at the end of this course but it's important to say that **the policy is well known** and stored in some database. I want to say that when talking about these kind of topics the **world database** is very often employed, in many cases database are just text files. We want that the framework supports the functions of the network allowing people to use resources, allowing programs to access resources.

The **original idea** is coming from well known MIT, this is the link to the dialog, it was the A plan for talking about kerberos and today is very largely employed worldwide. There were some imitations in usability. I want to say we will be describing now kerberos version 4, today there is version 5, general philosophy is the introduced version 4, we will see what differences there are, in version 5 we can recognise the introduction of some technical details, a richer environment but not so important under a methodological point of view.

Kerberos is considering the problem that Alice wants to access **a service provided by Bob**, to do that we must authenticate Alice, Alice can use the service, in some cases also Bob should be authenticated, not always necessary, in some cases the conversation can take several messages, in those cases is important to have **confidentiality**, to have the possibility of having a session key between the 2 parties, Kerberos is introducing the **trusted server** and this server is a trusted authority and the authority is of course a reference server in the kerberos framework, trusted by every part it's known with this name, KDC, **key distribution server**, why this name?

Because it's the authority distributing the session keys when they are requested. There is the **KDC**, trusted authority and the hypothesis a trusted authority is storing a **master key for every user** in the kerberos network. Every participant of the kerberos network is having a master key, configured when you are registering to the framework, and this is a secret key **only the KDC and the participant know** such a master key. The approach is generating tickets allowing users to access services. **What is a ticket?** We have already seen what is a ticket, we will see again but a ticket is some information that is provided from one side to the other side and what we remember from the tickets we saw is that the person providing the ticket is not able to understand the content of the ticket, **only the receiver can decrypt** the ticket and check the session key and the identity of the person, this is approach, also tickets are valid in a given time window, so a ticket that expires, you can decide the **expire time** of a ticket and different expire time for different types of service. The main idea is that Alice is obtaining from KDC a ticket and then she will use the ticket to the conversation with Bob, this is one of the versions of the protocol. That is kept in a physical safeness. All messages exchanged are **safe** with respect to confidentiality and data integrity, and they are basic requirements of information security, and a framework like kerberos is providing security for applications well designed in some unsafe mode, like telnet.

You can use kerberos for communications to remove hosts from making network distributed file systems secure, for application level like emails and other needs you can imagine. Every participant in the kerberos network is known as **principal**, so every human user program user, service, is a principal, and every principal is **registered with KDC** and he's sharing a secret key that is a master secret key with such trusted authority. In the case of humans typically the master key is derived from the password, so that the human can choose a password, then from the password there is some cryptographical good algorithm to **obtain the master key**, you can use some hashing functions or other variants of such an approach, in the case of computers or programs you can use some tools for generating master keys and storing directly in the configuration such information. Local database. Is important to say that I repeat that the KDC is **massively protected**, in particular is having a master key, used for encrypting all the master keys stored at the server.

Implementing Kerberos

If you want to implement Kerberos in the house you will have to configure your server and require you to write a password, ok, this is the first idea, just a **preliminary version** of the idea.

Alice wants to **get a service from Bob**, so that Alice is sending a request to the KDC and the KDC is providing some reply, after that Alice will use that information with Bob. This information provided by the **trusted server**, that is encrypted by the master key of Alice and such information is the session key the ticket to Bob, the ticket to Bob is information encrypted by using the master key of Bob and such information is the identity of Alice, the session key and other information that can be used in several case. **Only Bob can decrypt such a ticket**, so Alice asks the server, it provides information to Alice, for Bob, this is the ticket and then Alice is sending something to Bob in order to implement the setup of session key between Alice and bob.

Typically a ticket is **encrypted with master key** associated with the receiver of the ticket and in practical cases tickets can contain some information like not only a session key, not only username, username, username is the name of Alice, service name is the name of Bob, maybe the network address of Alice, the lifetime is a very important information because it's defining the **expiring time**, starting from the timestamp and starting from the timestamp the ticket will be valid, for the lifetime, that is a **number defined** that contained into the ticket. You understand that in order to have a good implementation you need all principals they are having a **synchronised clock**. This can be a vulnerability. Attackers can try to attack the clocks for making particularly smart attacks violating the rules by lifetime.

So let's see a little better **how starts the interaction**, and how it works. This is still a simplified version, Alice wants to use a service from bob, she needs a ticket from bob. Alice sends to the server I'm Alice, I want to talk to Bob, this is my nonce. The server sends to Alice the ticket to Bob and encrypted information for Alice that is the session key, the confirmation of the nonce, the lifetime of the ticket, the identity of Bob. The ticket for Bob of course is **encrypted** by using the master key of Bob and contains the session key, the identity of Alice, the lifetime and the timestamp. Alice when getting the message can **check the nonce** and learns about the lifetime. Alice can send to Bob the ticket for Bob, when Alice wants to use a service in most cases she wants immediately but it's important she asks for the service **within the**

lifetime. When asking for the service Alice sends to Bob the ticket to Bob and a new information here that is known as the **authenticator**, what is this?

Not really new, looks like information provided. In the kerberos framework this is called authenticator and we will understand step by step the role of such information, such authenticator is the identity of Alice and a timestamp generated by Alice. The meaning of this authenticator is ok I provide to Bob my identity and the current time, I'm sending the message, encrypted by the session key, Bob can decrypt such authenticator once he has decrypted the **ticket to Bob**, so Bob gets the session key and other information like the identity of other party and Bob can decrypt the authenticator and check the identity of Alice, also checks if the **timestamp is consistent** with the current time known by Bob, again I say that kerberos is requesting a **synchronised clock** for all systems.

This can be a **vulnerability**. Bob now can send to Alice the encryption of the timestamp the original timestamp sent by Alice this time by the session key, this is a message different from this other, now Alice can understand that other party was able to decrypt the ticket, this is implementing the **authentication of Bob**. Now we have some practical problems, we need to imagine such a process in operating environments where people keep working a whole day. When there is a message between a host and key server the message is protected by encryption, the encryption is obtained by the master key, imagine that is Alice a human and Alice is sending a request to the server for several tickets, every time she asks for a ticket she needs to send an **encrypted message to the server**, the encryption is made with the master key of Alice, how can we manage that?

We have **2 possible solutions**, one solution is ok every time Alice needs a server she provides the password, the workstation of Alice derives the master key from the password and encrypts the message. This means that for any possible service she has to **retype the password**. Other possibility is that the master key is **stored locally**. This is considered very insecure. None of the 2 solutions is considered secure. Is it clear? We don't like to retype the password every time we need to send a message to the server, we don't like to store the master key. Ok, this is introducing a new type of solution that is a **meta ticket**, but in the Kerberos world people are using **ticket granting ticket.**

For some reason this ticket granting ticket impacts on students of our course and students implementing the popular app Telegram, implementing stickers? Some students implemented a set of sticker for the department, so **I got my sticker**. This is exactly my sticker. I'm still asking **why**, my students know me for several reasons. Emphasising the ticket granting ticket on my face, ok, I like it, but still don't understand.

Ticket granting ticket

Let's talk about this ticket granting ticket. This is a **meta ticket**. The main idea is, Alice is having a ticket to ask for tickets. Every time she needs a ticket she's not requested to retype the password, she just shows the ticket. This is under a philosophical point of view, considering the service of providing a ticket like a **standard service**. Alice can have the service from the printer, network file sharing, from the server providing tickets.

To implement a TGT you can imagine some particular moment, in the morning Alice gets the office, she opens the workstation and **performs a login**, she needs to type the password and from this password some messages are exchanging, and a session key is derived for Alice. This is a new session key, a session key used for the let me say for **a session**. For several hours the work session of Alice is continuing, it's using a session key, this is the symbol used here. The session key is having a **fixed lifetime**. While doing the job we give Alice a TGT, including such session key, and other useful information. If the server wants to give Alice a special session key the server is **encrypting** such session key using some master key.

Later, Alice sends request to the key server saying ok I want to talk to Bob, when sending such a request Alice will be sending also the TGT, the ticket for having the ticket, since this is a ticket, who is the **receiver**? The KDC. According to what we have told, what is the key for encrypting such TGT? Master key of the server. This is a ticket meant to be sent to the server. Alice gets TGT containing several information, in particular such TGT is **encrypted by using the KDC master key**, so that when Alice wants to use server V the KDC will obtain the TGT and will be generating a ticket.

You can imagine the Kerberos framework like an office providing services and there are **2 special offices** because there will be an office making the authentication of all users, authentication server and then another office providing tickets, ticket granting server, these 2 acronymous are ofter employed in kerberos framework so that is useful to see such acronymous. In this picture you see some steps, they are number, 1, 2, 3, 4, 5, 6. **Step one**, we have Alice logging on the workstation and requesting service. This is sending message to the kerberos in particular to the authentication server and you see this message that is used once per user login session and also the reply for this message is one only, so that in this first message there is the **request for ticket granting ticket**, the authentication server performs authentication in some way, you see here some schema so that the server is accessing some local storage or some database for **checking the identity of Alice**, if authentication is going in a good way the server is providing a good response, and the session key for Alice.

The session key that is having lifetime for the **whole day**. Now Alice wants to use a particular service, in order to use a particular service Alice is sending a **message to the office** for obtaining the ticket for the services and in order to do that she has to show the ticket granting ticket and the identity of the service she wants to use, so she sends a message and obtains a reply, the reply is a ticket for the service and a session key to be used in the services of course. This type of exchange request reply is once for type of service. This is not completely precise, because **tickets are expiring**. When Alice wants to use the service she sends the request to Bob, Bob will get the ticket to Bob and the authenticator from Alice, this will be **starting the service** whatever is the service, maybe implying a conversation between 2 parties, just a first general view, of what actually happens in the Kerberos framework, in the next slide we see more in details the content of the messages.

Alice is using the workstation, so she needs to write the type of workstation. First message is ok I'm Alice and I want ticket granting ticket. This is the message, a format accepting the protocol, AS request, **authentication server request**. The authentication server generically here indicated as KDC is generating a session key for Alice, then he finds the Alice master key, and he's providing ticket granting ticket like the encryption of such information, identity of Alice, session key for Alice, the **content of ticket granting ticket**, the encryption by KDC and the reply to Alice will be the session key. The password of Alice is important, here it's not clear, you can imagine at the beginning Alice can claim ok I'm Alice, so the server is doing all its job. Even if Alice is not Alice but is **an attacker** all his job is providing an encrypted message to result. To decrypt this message the master key of Alice should be used at Alice time. This is the time where the password is really important. Now the workstation will use the password for **deriving the master key of Alice** and will be able to decrypt such information. If this is not possible every attempt to login will be useless because no ticket granting ticket because no session key for Alice.

Now Alice wants to use a service from Bob. She needs a ticket for that, so that Alice asks now for a ticket to Bob. This is a request to the **office providing tickets**. The message is a message like I'm Alice I want to talk with Bob, this is my TGT. The **TGT** is some information **encrypted** by using the master key of the server, Alice is using a ticket now, it's the ticket granting ticket but it's a ticket, she needs to provide also the **authenticator**, some information that is encrypted by using the session key so Alice is using the session key for encrypting the time stamp, so that the key server is **not storing locally the session key** generated at the beginning the server will be able to obtain the session key by decrypting the TGT so will be able to check.

At **server side** the server generating the session key verifies, of course gets all the checks, then extracts the master key of Bob and prepares the ticket for Bob, containing the session key and the identity of Alice. The reply of the server is encrypted by the session key generated for Alice and containing the identity of Bob, the session key and the ticket to Bob. All these content are important, you can imagine in reality Alice **is asking for 10 tickets** because she needs to perform a very complicated operation, so she's acting at application level, she may push a button that is requiring many actions, many conversations with several parties so **many messages** like that are originated at Alice side at the same time. You understand that is very important to focus on the identity of the other party and to use timestamps.

For using a ticket Alice is sending a message to the service, this is AP request, AP I guess I don't remember like **another principle**, the message is the ticket to Bob and the authenticator, again because she's providing a ticket with the ticket she provides the authenticator, the encryption of timestamps by using session key, again Bob decrypts the ticket, gets the identity of Alice, the session key and extracts timestamp and can reply with some schema known to the parties so that the authentication will be performed. This are the **3 fases important**, login of Alice, request for the ticket granting ticket and request for, login is provided by TGT, then request for every service she needs, not needed any password, and then request for the service directly.

A common question is ok ticket granting tickets are used in order to prevent the need of Alice retyping many times the password, you don't want to store the master key of Alice, but **what about the session key?** This is a session key. It's looking like a master key, it's a key, what is the difference? The session key is used for the **session of work** of Alice, can be stored in RAM, not stored in physical storage, and after logout or after some failures of the system errors, power errors or whatever you can imagine, such information will be lost and you need to perform another login. So we can confirm there is no need to retype information at Alice side. This is a good solution orienting all the functions of kerberos. For some weird reasons there are all the copies of the slides.

In large networks you could have many printers. All printers are identical and have the same master key. If Alice is using the service of one printer the attacker can record the ticket and the authenticator and reuse immediately such information for using another printer. There is some **easy fix** to this issue. One thing is that in the content of the authenticator there is the identity of the particular instance of the printer. In the literature you read any time **KDC authentication server** and the ticket granting service, so the logical question is ok are they the same? I would say that different terms are used **only for historical reason**, you can implement the 2 types of services at the same side, you don't need to touch the 2 services. In large networks you need to have a multiple instances of KDC and also for incrementing reliability.

It is a standard considering the protocol and in this case you have **many KDC**, there is a master KDC and all updates are made on the master KDC, that is pushing new information to the other KDC that are **somewhat slave**. Keep in mind that the KDC is providing service like I get the message, the login of Alice, I check the identity of Alice, I extract the master key of Alice, then I generate session key, I generate some information, it means that information stored in a KDC is used more frequently in read only mode. That is **read only**, because you have to write when you are registering a new user or choosing to modify some policy, most of the time is used in read only mode, in case of failure this is somewhat making all user quieter because ok it's failing, the other copies of KDC are supporting all the service, in most cases answer is yes, changes are not so frequent, in most cases big organisations are having some internal protocol for implementing changes in the KDC.

It's not possible to implement in KDC **only in some particular time**, so that after the time interval the updates are pushed to the slaves. But this is not what we necessary need in every case. Maybe some organisation wants to manage other cases. We already commented here, no need to say more. **Message types**, there are several message types, some messages are implemented for managing special cases, I don't want to go in these details, too boring, we are just learning the general framework, the most important messages are request and reply we have considered, there is a message type that is **never used**.

No other details because I want to talk about tickets. Already talked about the content, a ticket is containing the name of the user, the address of the user, session key, lifetime, organised as units of 5 minutes, timestamp of the building of the ticket, the name of the server, in kerberos, timestamp organised in seconds, I will talk next time about the last topic about kerberos, in the case of topic is confederating several kerberos environments, I want to talk about the homework.

Homework number 3

HW3 title: Multiple parties Diffie-Hellman

Diffie-Helmann run with more than 2 users, again one week of time. **16 November 11:59 PM deadline.**

The description of the homework is very small, we are considering DH process for exchanging a session key, we know very well the way we do that for 2 parties. Now we want to define a good sequence of messages so that **3 parties can agree the same session key.** So in your homework you have to define how 3 parties can agree one unique session key, you have to draw the sequence of messages and consider what is going to happen if some messages are delayed, what happens if the message is delayed? All the generation, all the agreement is delayed, some part gets some delay, what happens if you have one party that is compromised, means **you can't trust that party**, but other parties don't know that. What happens? In the case of 3 parties, one of the 3 is compromised, they start exchanging messages according to your sequence of messages, what happens, what is the damage that the compromised part can provide to the construction? Of course there is a trivial thing not important, not worth it to be said, if there is a compromised party participating in the construction of session key, the compromised party is later sending the session key to other attackers so it makes easy some attack. Not important.

What is important is that ok we can focus like there are only such parties, no other attackers. What happens in the process of reconstruction? About compromising just it's useful this last case. This is the objective of the homework, it's very easy to be performed, I want to show again some rules, people forgot to send the contribution to this email address, cns@dis.uniroma1.it

There is a rule for subject, attachments, don't zip attachments, there is rule of email body, should contain the output of this hashing function. I still wonder why some people. **Have a nice weekend.**

More information about homework

There is **no precise structure** for the homework, I like to evaluate your creativity so I want to see how you are considering the topic so what ingredients you want to add. In particular for the current homework I would say that you can design in several ways the messages for the parties and depending on the sequence of messages you can setup system where the delay of one message is determining the **delay of the convergency** of the whole system. You can setup different schemes for messaging and you can afford some delay, and let other part of the process going forward. I would like to see a very **good design** for the messages in the case of 3 parties and then generalisation.

If you like that you can wait for the thesis. I am completely satisfied with the most natural choice for the session key. Still interesting to try to **minimise the number of messages**, because you can define different patterns for messages. Concatenating several parts in one message of course.

Even if some day the key exchange based on DH will be obsoleted never forget the **importance of their contribution** for public key cryptography. They were the inventors of the idea. On the same way Rivest, Shamir and others were the first implementors of the idea. What if your **grandmother** asks you hey son tell me what is this stuff, cybersecurity, how do you explain that to some people that are completely unaware about information technology? Think about that. We will be back to this topic, it's important to be able to talk to all possible kinds of people.

Kerberos realms

There are several cases where you can imagine different networks belonging to different enterprises that for some reasons **need to cooperate**. What does it mean to cooperate? It means that some users of one network may want to use a service from **another network**, for some good reasons, if you want to implement some integration between the networks you can do that and one way of doing that is implementing **realms**.

Realms is like a federation of kerberos networks, with some strong properties, you can't just take 2 kerberos network and say ok let's make it realms. Not so easy, you have to **redesign**, reconfigure some parameters about the networks. If you look in a documentation online you will find very good documentation from IBM, talking about the way of setting up in a quick way the integration of different realms. You can setup **different realms**, many, don't think you can setup some whatever number of realms, the cost of setting up k realms is **quadratic in k**. So you have to setup quadratic number of keys and this is somewhat annoying if you are having too many realms. Each realm has a different master KDC. We already know that in addition to a master KDC you can have some other KDC, **read only,** so you can have better performance because KDC in most cases are working in read only mode, you can replicate information on several servers to improve reliability and performance.

What is important is that if you have let's think the case of **2 realms**, your master KDC should be a principle of the other realm and vice versa. This is the requirement. This means there should be **some key agreed** between the different master KDC. You can setup the thing so you can have just **one master key** shared between all possible KDC, not completely realistic, it means you have started to plan the structure of the different realms from the beginning, normally things happen so that you have already your realm and you want to add some extra realm, what happens is that the master KDC of the new realm will be a **new principle** of yours, you have to setup a key shared between the 2 parties, the vice versa must hold, when you are setting up 2 different realms you want that **whatever user** in the 2 realms can use services from the other realm, you want a completely symmetric situation.

A part this detail realms are **autonomous** in the meaning that every master KDC is having a different list of users, that have the proper list for that realm. You can setup situation that can be complicated and add many realms. What is important to say is that ok in kerberos 5 this concept has been developed but what is important here is that a very critical part of using a service in kerberos is **being authenticated**, I am authenticated, I get the session key, I ask for the TGT, this is very important but **who can authenticate** a user? Only the KDC of the same realm. Because I told you every different KDC is storing information about the local users in that realm. You have to setup some trust between the different KD servers. A KD server will get a request coming from another realm after that user **has got the authentication** with its KD server.

So the authentication is made locally, after that the user is enabled to ask for a service in some **other realm**, the authentication is finalised in the local realm of the user. If you want to simplify the approach you can see every principal like a user or a server, because participants in kerberos are 2 types, users ask for service, servers they provide services. You need users being able to authenticate locally in the correct realm. So this is the **simple exchange** of messages we can setup. You suppose that Alice wants to ask for a service, provided by Dorothy, in another realm. Alice starts with a request to its local KDC, I want to add a detail, when registering a user in kerberos you register the name of the user, the instance of the user, at least for kerberos 4, and realm of the user. So in this case the realm of Alice is **wonderland**. She wants a service from Dorothy, so she needs a ticket for Dorothy, who is in charge to give tickets for Dorothy?

The ticket granting server of the realm of Dorothy. Dorothy is belonging to a **different realm** so it's important that Alice can ask the service for Dorothy so that the ticket for Dorothy by a request to the ticket granting server of Oz, in charge of giving tickets for that realm. How to ask a ticket to such object? I start in my realm and I ask a ticket for asking a service to this external server here. I can ask this service to my local KDC because this is the KDC of the other realm, registered in my realm. So I ask my KDC for a ticket and credentials for making my request to Oz KDC, then I send a request for Oz KDC asking for a ticket for Dorothy and he will be replying to me in the usual way, so that I will be able to **send my request.** Tickets work in the usual way, a ticket is encrypted with the master key of the receiver, you need that who is building the ticket must know the encryption key for the receiver of the ticket.

This is a simple schema, you should understand now that if you in some incremental way you will let your network grow in time, add a realm, for every realm you add you need to register the new KDC of the new realm in **all your KDC**, you will have a linear number of registrations and keys to be managed for every new realm, this is why you will have in total a **quadratic effort**, this is not so nice, considered like a **limit of kerberos** and this is the reason why while passing from kerberos 4 to 5 also a hierarchy of realms has been introduced, so that is not necessary to have a flat knowledge of all possible realms. We also should remember that at time of designing of kerberos 4 the encryptor employed was DES algorithm, **completely broken**, so what happens while passing from kerberos 4 to 5?

Here there is a small list of new additions, new ingredients, just nice to know in some general and summarised way. Of course all people involved in real configuration of some realm and using kerberos 5 will be important to add extra knowledge, it becomes into technical knowledge, other technical details. I expect when it becomes a **technical detail** you just open some document and say ok your tool is named key admin, ok run key admin, and so on. In version 5 there is some more explicit integrity of messages, in order to finalise the authentication also **nonces** are introduce, not only timestamps allowing to make a finer contrast to attacks, because there is a simple way you can **exploit**, you can organise your management of nonces and timestamps so you become completely robust again replay attacks. We will see in the next lesson about authentication based on public key.

Also you can encrypt information by using different algorithms, making encryption and in particular algorithms are not just constant name, you can follow the evolution of encryption algorithms and so on. There is some flexibility, so that you can add extra features without the need of redesign the protocol, without the need of issuing a new version of the protocol, one of the most nice things introduced is this concept of **delegation of rights**. Alice can allow Bob to access the resources of Alice, and Alice can define in a very precise way not only the amount of time Bob is allowed to use the resources, but also what **subset** of resources are granted to Alice, also the concept of **renew a ticket** is introduced, so it's easier to prolong the duration of a ticket.

In many enterprises you start a job running for 2-3 days, normally tickets are expiring in hours, at most one day, so it's important you can create the precondition so that a long job starts with a valid ticket and while the ticket expires the job is still running. Practical details are welcome of course but you understand the general philosophy of kerberos is not changing. Kerberos is considered **powerful**, very reliable but security analysts keep saying that kerberos is not so good, there is a single point of failure, you understand in particular the **compromising of the KDC** means the attacker can impersonate every principal of the enterprise, this is not really corresponding to the general methodology of handling security, so that if something fails the consequences are bounded, in some subsystem, subset of resources, this is a complete general failure, the most important argument used **against kerberos**, despite of that Kerberos is available for all platforms, still considered to be very good, it is providing the following service, instead of securing all your system you must secure in a complete physical and logical way one only system, the KDC. You can continue discussing about this topic, there is no final conclusion and will find many people that are very positive with respect to kerberos, other people, not the majority, they are **negative** with respect to kerberos. Any question?

With this I want to **close the topic kerberos** and I want to go back to the general path we are following, because after confidentiality, data integrity, we started authentication, we introduced a challenge response approach, based on symmetric keys between 2 parties, can we consider the introduction to trusted 3 parties with some possible approaches, the practical consequence of this is the protocol whose name is kerberos, now we still need to understand two other possible strategies to **make authentication**. One strategy is based on public key, the other is closer to the experience of user based on passwords and this will be the last topic about authentication, passwords are so important of course, we must say a few things and understand the meaning of weak and strong password.

Authentication with public keys

Authentication based on public keys. What does it mean **public key?** You know what a public key is. If we have a public key we have some other key, private key, associated to the public key of course. Since this approach has allowed the implementation of new algorithms and is allowing the construction of **digital signatures**, we need to better understand what is required so that the digital signatures can be considered completely secure, and what other schemes of authentication based on public key or **digital certificates** can be used, what are the last standards worth to mention. The basic problem in that we have already discussed, I have already introduced, I want to repeat, is so important.

If you want to setup authentication based on public key you can send a **signed message**, ok, you prepare a message, you sign with a digital signature your message and the message can contain some challenge, password, whatever you want, you can reuse some approaches we used, the weakest point of this approach is can you **trust the public key**? Because when somebody is receiving a digital signature of course it must be possible to verify that the digital signature is trustable, there is a verification algorithm very easy. You just get a message, you get the digital signature, the verification is based on **2 steps**, you just compute the hash, the second step is take the digital signature, decrypt it by using the public key and check if the result is the same you got from the hashing. They should be matching.

So, you are aware about the fact that in order to run the verification algorithm Bob needs to know the **public key of Alice**. What if the attacker sends a message signed by the attacker and says to Bob hey this is a message signed by Alice, this is Alice public key, instead is the public key of the attacker, if Bob is believing this statement he takes the public key, makes the check and finds that public key is associated to the private key for signing the message. With a small detail, it was the attacker to sign the message, **cheating** saying that is Alice's public key. If Bob believes to the statement, after the verification Bob will think the message is signed by Alice, this is the **weakest point**. We need to be able to **associate in a secure way** a public key with some identity, we need a strong association.

One tool for implementing such association is a **digital certificate**, we will talk about them, it's not the only tool, there is another distributed approach named web of trust, we will be mentioning it later. It's good to be completely comfortable about digital certificates because they are allowing people to be secure about the identity and if you remember when we talked about this problem few weeks ago I said that a digital certificate can be proving a **digital identity** or also some analogic identity. I mean real identity. I can offer a digital certificate that is showing my public key but I can also offer a digital certificate showing a public key associated with an internet domain. This happens when you are connecting to a website. You will get a digital certificate that is **associated with the subdomain** where the http server is running, it's not associated with a name. Can be also associated with the name of some organisation but we will talk about that. Any question?

Ok, as you could guess again Needham-Schroeder also considered the problem of authentication based on public key, this is a very obvious and **simple scheme** for implementing mutual authentication based on public key. So, this scheme is somewhat **simulating the presence of some authority** that is providing public keys of users. And this authority is trusted from all parties, so that the public key you got from such authority is the correct one and everybody can ask for the public key of everybody. By definition a public key is **public**, the more is public, the easier the usage of such key of course. Ok, so that in order to implement a **mutual authentication** between Alice and Bob, Alice could ask the authority, I'm Alice I want to talk to Bob, the reply is the identity of Bob, the public key of Bob, you see here there is no need to implement **any cryptography**, we are no concerned about confidentiality, we are concerned about authentication.

The authority is providing the digital signature of the message, made by the authority of the message containing the identity of B and the public key of B. I ask the authority, it replies with the information in order to let me believe such information the authority is **digital signing** the information providing to me. So I get this signature, I check the digital signature by using my verification algorithm for digital signature, I **generate a nonce** of course the usage of a public key for setting up authentication is strategic choice, then you need something else for finalising the authentication, you can generate nonces, challenges, timestamps, invent several variants, here it's just shown the generation of a nonce, in this way Alice is enabled to send to Bob a message encrypted by the public key of Bob containing the nonce and the identity of Alice.

You remember what we told about the usage of public key encryption for **confidentiality**. You remember, we are not really using that, expect what cases? Where the content of the message you are encrypting for confidentiality is small, if it becomes bigger message the computational effort will be very huge, if compared to effort of symmetric cryptography. Here the message being sent to Bob is a small message, contains a number and identity. Ok, so that Bob can decrypt the message by using the private key of Bob and **knows the nonce** and the identity of A. Now Bob wants to check Alice, so Bob just is doing the same and Bob is sending to the authority I'm Bob I want the public key of Alice and the authority is providing the public key, sends to Bob a message that is **identity of Alice**, public key of Alice and digital signature of this message, just the same, in the step before where Alice was getting the public key of Bob.

Then Bob can check the digital signature so he gets the public key of Alice, Bob can generate another nonce, N', and can send to Alice an encrypted message containing both the nonce generated by Alice to **prove it's me** and generating another nonce to be provided to Alice that is another challenge, and Alice can decrypt the message, checks wether the N is the correct nonce and can send a message back to Bob that is the encryption of the nonce invented by Bob.

Design an attack

We are not assumed the user can be cheated about the public key. We are assuming the trusted server is providing the correct public key. If the public key is **wrong** of course you will get wrong results from the verification. The scheme seems natural, what we can expect by using authentication based on digital signatures, can we **design some attack** to this simple scheme? Of course yes, we can design an attack. This is not the real final scheme but the difference between this and the real one is **very small**. Just a detail. Ok, how to **attack this scheme?** We assume the attacker is another user of the same network, so just like Alice and Bob the attacker can talk to the trusted server maybe because it's just a normal user of the same network, using the same way to make secure authentication of all users and again the attack is based on the **construction of 2 different sessions**, we have already seen that type of attack for allowing one session having success, the other sessions ok it will be not finalised.

So we have 2 sessions, R1 and R2. R1 is the authentication between Alice and Trudy, R2 is the authentication of Trudy **pretending to be Alice** and Bob. This is a way to implement **man in the middle**. As a precondition it's needed that Trudy is able to push Alice to start an authentication with Trudy. This is a precondition. It is hard to suppose that Trudy is able to push Alice to start a conversation? Not hard, if they are users in the same network they are **coworkers**. And there are many processes that are somewhat relating the work activity of both parties. Alice and Trudy.

By some simple **social engineering** Trudy can push Alice to start a conversation. In some cases Trudy is just asking Alice to delay some needed conversations because they have to start talking and Trudy is convincing Alice to start the conversation because Trudy wants to **setup the attack.** If you check the step 1, 2, 4, 5 they are simple steps to **get the public key**. 1, 2, Alice asks for the public key of Bob and gets it. 4, 5 Bob asks for the public key of Alice and gets the public key. The other steps, 3, 6, 7 they are useful for finalising authentication because the parties have obtained the public keys of the parties. Ok, so it’s not really important to see steps 1, 2, 4, 5, we understand it’s easy for both parties to get the public key of some other party.

So, it’s interesting to focus on steps for running the authentication, we are talking of both sessions, R1, R2. For what concerns session R1 remember session R1 is authentication **between Alice and the attacker**. Alice starts a conversation with the attacker. So in the previous step Alice has got the public key of attacker, in the step 3 Alice sends a message to the attacker containing the identity of Alice, the nonce encrypted by the public key of the attacker. About the **other session**, in the other session I want to remind you the attacker is pretending to be Alice and wants to talk with Bob, so wants to cheat Bob about the identity.

So, the attacker by pretending to be Alice while implementing step 3 of the second session, where the attacker pretends to be Alice and wants to talk to Bob, sends to Bob a message that is a nonce that is the containing the identity of Alice and this is encrypted by the public key of Bob. This is **easy** for the attacker. Notice that the name of Alice is known to the attacker, the nonce N here being sent from the attacker to Bob is the same nonce the attacker has just obtained from Alice in other session. So that Bob receives this message from the attacker and replies, he is thinking to send a message to Alice, he’s sending a message **to the attacker**, step 6 of the second session, in this step what is requested by the protocol is sending back a message to Alice, Bob thinks to talk with Alice, so that is providing the nonce obtained by Alice and the new nonce N’ generate by Bob. Bob is encrypting this message using the public key of Alice, he thinks he’s talking to Alice.

Of course in other steps he has obtained the public key of Alice by simple requests to the server. Now the attacker can continue in the first session of authentication between Trudy and Alice and since the last message Trudy obtained by Alice was the encryption of the nonce and identity of Alice now trudy can reply and send to Alice the nonce obtained by Alice, N, and n’ that is the nonce generate by Bob. Bob generates a nonce. Bob thinks to send the nonce the Alice, actually the message is sent to the attacker, the attacker is now sending the message to Alice in the other session. Can the attacker do that? Yes. So according to the protocol Alice is **replying to the attacker** sending as a response the encryption of N’ by using the public key of Trudy. This is what is requested, Trudy gets this information and now can act like Alice and send a message to Bob by implementing last step of the other session sending a message to Bob by using encryption public key of Bob of N’.

Notice please that how can **Trudy know N’** this is the point. When trudy has obtained N’ from Bob the message was encrypted by Bob using the public key of Alice. Trudy can’t decrypt the message, so is providing the same message to Alice so that Alice decrypts and finds out what is N’ and then sends N’ to trudy, now trudy can send N’ to Bob encrypted with the public key of bob. The difficult part for the attacker, that was decrypting a message meant to Alice has been done by Alice in the other session. Trudy is doing simple things, sometimes intercepts a message containing N’, Trudy needs N’ but can’t decrypt N’, so uses this message in the other session with Alice. Now Trudy can encrypt a message to Bob containing N’.

This is **the fix**. The protocol is just the same, but when we finalise authentication if you want to send a message, Bob to Alice, sending back the nonce generated by Alice, the new nonce generated by Bob and also identity of the sender. So the attack no longer works, this is just the only difference that we need for contrasting that type of attack, the previous attack now is failing, because when Alice if the attacker tries to ask Alice to decrypt a message to obtain N’ Alice gets the message but this time the message contains the identity of Bob, Alice is talking to Trudy, so Alice is able to say **this is not your identity.** This is the identity of Bob. The attack is failing for this reason. So, what we learned, we can implement a simple 2 ways authentication, letting the parties know the keys of the counter part. If you check you just add some identity in the messages the reflection attack based on the construction of 2 sessions for finalising one of them the reflection attack is **no longer useful.**

Digital certificates

What is a digital certificate? If we see at this message here being sent by the authority to Alice you can consider it as a **digital certificate**, it says that the public key of Bob is this key, signed by the authority, if you trust the authority and verify the signature you get the public key, the whole message is a digital certificate. Some standards for writing digital certificates has been developed, current standard is X-500 at version 3. Has been set up at the beginning of current millennium, **not so new**, but is a stable standard and it’s very commonly used and is the base of currently public key infrastructure. For the particular case of Italy in Italy there are some several rules for the construction of what is called the **code for digital administration**, a set of rules, according to that it’s the digital certificate one of the possible tools for qualifying the identity of people doing actions in the network.

The concept of qualifying is very important under a legal point of view. The design of this standard is belonging to a wider process of designing a larger standard, the standard X-500. I don’t want to go in all details of this big standard, I find it very **boring**. It’s containing a lot of conventions about the way entities and parties should be named. In such standard 3 examples of authentication protocols have been proposed:

- One way authentication
- Two ways authentication
- Three ways authentication

In this proposal there are several details, what is important to say is that they are **moderately modern** standard. After some time people doing standard understood is not so good to specify what algorithms should be used by the standard, because algorithms are evolving in time, but the main idea maybe is still valid, there is just need to **update the algorithm**. This is why you are not really reading the name of algorithms, you read the field that is expected to contain the name of the algorithm you are using.

This is the simplest case, the standard X-500 defined a **one way authentication**, a simple message, being sent by Alice to Bob, after this message Bob will authenticate Alice, he knows the identity of Alice, you see the content of the message. The digital certificate of Alice, this is like sending a secure public key, this is such a digital certificate, a statement provided by a trusted authority, what today we call **certification authority**, by definition a trusted authority, even if you check for the literature about attacks, several certification authorities have been compromised in time. Normally you can trust well known certification authorities and many customers, existing since many years.

This is the message, the digital certificate stating what is **the public key of Alice,** a message that is here denoted by this simple data coming from A, this is the content of message, a digital signature of the message. So, message, digital signature, digital certificate. Bob that is receiving such information can use the digital certificate for extracting the public key and check wether the digital signature is ok, if ok Bob can trust the message, containing a timestamp, the identity of Bob, and a proposal for **session key** to continue to talk. We normally use the session key, Alice is sending a session key to Bob that is generated at Alice's side and such session key has been encrypted by using the public key of Bob. Notice that in this scheme is assumed that Alice is already knowing the public key of Bob, otherwise you can ask where is the message where Bob is sending the public key to Alice?

This is not **symmetric** authentication, the role of the client and server are different, Bob is server, we can expect the public key of server is known for several possible reasons, we don't want to argue now on the reason why Alice is knowing that key. We here just use as hypothesis the fact that Alice is knowing the public key of Bob. Since the two sides are not symmetric we expect is easy to know the public key of

Bob, **not easy** to know the public key of Alice. In this way we implement what is called a **one way authentication.** You see here just the list of points we have already commented, because Bob will understand it was Alice generating the message and the message generated by Alice was meant for Bob. The attacker cannot just **record the information** and use it with another server, that information is containing the identity of Bob and encrypting information using the public key of Bob. The timestamp is somewhat preventing the reuse of the message multiple times. The attacker can record the message and send it to Bob pretending to be Alice that wants to start a new session. The presence of the timestamp is somewhat **preventing** that but I want to describe a technique that is a general technique showing how to use a combination of timestamps and nonces.

You can use both of them to contrast replay attacks, based on the fact attacker is **storing your messages** and using them again after some time. Assume that we want to use only a timestamp, so Alice is sending in a secure way some information to Bob, and in this information there is also a timestamp. Very precise, hour, minutes, seconds, ms, why not. You understand the more the finer is this timestamp, the harder is the management. You don't expect bob is receiving in **real time** these timestamp, even with completely synchronised clocks. How long takes this message to get to Bob? It can be few milliseconds, one second, so you should introduce some tolerance about that, you must agree on the fact that the message is **valid for some lifetime,** maybe 2 seconds, for instance. The discussion is valid even if you choose 500 ms. Whatever number you choose here, it means that if this is the axis of time, the timestamp you are generating, you need to define a small interval so that you can expect Bob is receiving the message within this interval, if so Bob will be considering the message valid, if out of the interval Bob will say **it's a reply attack**.

In this case a very powerful attacker maybe is able to see that this message is delivered to Bob at this time, very quickly, the attacker is able to immediately reply, still being in the same time interval, this can happen. So this contrast this measure is not completely secure because if the network is functioning in a fast way, if the attacker is **quick** and can act in a real time he can send a reply message here within the same interval, so the message is still valid. In order to prevent that you can add a nonce. The nonce is having a particular role, because when the server is obtaining such information the server can say ok I get the message, the timestamp is good, I'm still within the interval, I store the nonce. This is different from what we said. The nonces are **not to be stored**, because too many nonces, too many. It's not a good idea to store them. Let's see what we can do here. We store a nonce, if the server is storing a nonce the reply attack will be blocked because even if the attack is very quick he will send a message some time after the message of Alice, containing the same nonce.

So the timestamp is valid but the nonce is no more, and the server can see the attack. We ask server storing nonces. Yes, now we ask servers storing nonces but for how long time the server should store the nonce? Just for **this interval** to recognise an attack. If we store the nonce for limited time we can drop the nonce then. We are not storing nonces for all the time, just for 1 second, 2 seconds, then we drop it. It's not really overloading the servers. This is a **very good measure** to contrast reply attacks. This is general, not restricted to this approach, you can use this approach in the authentication, in whatever message you want to protect from replay, this is a general measure.

Two session keys

In the naming of this standard you will read the one way authentication, two ways, meaning the number of messages the one way authentication is implementing a one message based authentication. There will be a message from Alice to Bob and from Bob to Alice. The messages are naturally enough, you can see some messages, there is a signature of the message, a digital certificate provided so it's possible to check to **verify the digital signatures**. Notice that the other party is also sending a similar message, digital certificate of Bob, message coming from Bob and digital signature made from Bob on the message. The message is changed at the following one, when Alice is sending a message to Bob there is a **timestamp,** nonce, identity of Bob and the proposal of **session key** encrypted by the public key of Bob.

The answer provided by Bob is somewhat similar because you see here there is another timestamp for time-stamping the moment, a new nonce N', the identity of Alice, the nonce originated by Alice and the proposal of another session key, k'. Encrypted by the public key of Alice. Why **two session keys?** In some cases it makes sense to encrypt messages from Alice to bob by a session key, encrypt from bob to Alice with another session key. Both parties know both session keys, they are originating session keys and communicating to the other party. With 2 session keys it means there is a **precise role**. From one side to the other side with a session keys, messages in the opposite direction with another session key, this can prevent attacks based on **reflection**, for finalising the first section.

One of the measure used for contrasting reflection. Here there are **2 comments** in my opinion that are worth to be made, one comment is the following, we see here that Alice is communicating a digital certificate to Bob, so that Bob can check and verify the signature of Alice, after that Bob knows the public key of Alice and can use it to encrypt the proposal k', ok? The second message is somewhat made possible by the fact in the previous message Bob has obtained the public key of Alice, in the first message Alice is communicating the public key, but Alice is proposing a session key where the session key must be **encrypted using the public key of Bob**, so Alice is using the public key of Bob **before** getting the digital certificate, this is funny, not realistic.

It becomes **realistic** if we again accept the fact that the 2 parties are not completely symmetric. If we accept that here Alice is able to send a message to Bob using the public key of bob because in some cases the scenario is not symmetric, there are some good reasons why Alice is knowing the public key of Bob, here it's the same, this is valid as long as there are good reasons why Alice is knowing the public key of Bob, if there is no good reason if the 2 parties are symmetric they are ordinary clients, you can see the approach where this proposal of session key encrypted by the public key of bob is **missing**. If we omit this component in the message Alice is just providing some information and Bob is still able to reply and send the **proposal of a public key**, so in this case where Alice doesn't know the public key of bob in advance we can use **one session key** proposed by Bob in the second message.

Also there is another point here, another comment we can do on this scheme, in the message being sent by Alice you see the identity of Bob, there is no identity of Alice, so this is somewhat looking like a message that we be discussing about the protocol for authentication based on public keys where the **lack of identity** could be used for allowing an attack based on multiple sessions, it can be worth it to add extra information here to make it clear who is the sender, the initiator and the party providing the answer, both identities could be inserted here in the message. You understand it's easy to propose some **variation on this general schema**, old schema proposed when developing the X-500 standard. There is also a third proposal that is based on 3 messages implementing mutual authentication, meant to be robust against reply attack.

The 3 messages are like this, Alice to Bob, Bob to Alice and again Alice to Bob. In the first message Alice is providing a message, digital certificate and signature of the message. The message is containing several components, has the proposal of a session key, identity of Bob, a nonce and a timestamp. When the timestamp is described with this **symbol O it means it's optional**, the protocol is admitting the possibility that is not provided. The reply from Bob is similar, a message, the digital certificate, signature of the message providing an optional timestamp, the nonce obtained by Alice, a second nonce by Bob, the identity of Alice and a proposal of public key that **can be the same**, so this is not really a proposal but it's a confirmation, it can also be a different session key if they want to implement symmetric encryption based on **different symmetric keys.**

The new message being added to the protocol is a message from Alice to Bob that is the two nonces, there is a unique message containing Bob nonces, maybe the replay of single message **very hard for the attacker**, there is a signature on the 2 nonces. Anyway as a final remark about these 3 schemes authentication I would say they are somewhat "**obsolete**", somewhat old, they have been proposed several years ago and developers and security analysts are proposing different variants for implementing the authentication. The general idea anyway provided is still good, and I'm making reference to the proposal for authentication based on public key. Also we have seen authentication based on symmetric keys. Now a simple nice exercise.

This is what happened during a workshop where several researchers were discussing about the goodness of some way to **make authentication**. This is a possible protocol where Bob sends a nonce to Alice and Alice replies by sending the digital certificate, another nonce she's confirming the nonce taken by Bob, identity of Bob and you see Alice is **digital signing** the 2 nonces and the identity of Bob, this is the message, this is the signature of the message. And now Bob can send a message to Alice that is somewhat similar, digital certificate of Bob, nonce by Bob, confirmation of the nonce obtained by Alice, identity of Alice, and the digital signature of the message.

Ok, in the workshop the Canadian team after some found a way to **run an attack**, you can find some documentation about this fact the reference is here, this is again based on **multiple session**. So you should consider it like just an exercise at this point, we understood how multiple session can be dangerous and how to prevent the usage in this way. Attacker to Alice pretending to be Bob this is my nonce, Alice replies to the attacker, thinks she's getting a message from Bob, certificate, nonce, identity, signature, now the attacker **pretending to be Alice** sends a message to Bob this is my nonce, and sends to Bob the nonce Na, the nonce just obtained by Alice. Bob replies with the standard message to the attacker, Bob thinks he's talking to Alice and sends the digital certificate nonce Nb, Na, the identity of A and the signature.

Now the attacker trudy sends a message to Alice **pretending to be Bob**, a third message of the previous session of authentication, adding certificate of Bob all the nonces, Nb Na, the identity and the signature made by Bob of such information so just Trudy is using the signature obtained by Bob to confirm some information sent to Alice. Ok, this is another **successful attack**, just as an exercise today. We have been studying reflection and multi session attacks. You should remember 15 years ago, not so obvious, people where not used to reflection attack, this is somewhat **recent type of attack**, this was a field of research. Ok, just an exercise. Of course several teams working in this project finalise the proposal of a mutual authentication this is just was **fixed later**. Now we can start introducing the public key infrastructure.

Public key infrastructure

Public key infrastructure is what we need to have an efficient. Official PKI is based on the presence of a trusted certification authority, you can have **2 types of certification authority**, one is the legal one, an organisation registered in some public register and approved as a certification authority, its certificates are valid wherever is accepted such rules, such laws. For every country not every but for many countries there are **official certification authority**, trustable by definition, they have been approved after a procedure for the management of digital certificate and so on. But what happens if a legal certification authority is used for certifying the public key of some user and such a certificate is sent abroad in another country where that certification authority is not considered to be legal?

This happens very often, I guess even to you while having a normal usage of the browser or other primitives have met the issue of some certificate unknown. How to manage it? This is a big problem, because if the issue is unknown, there is no path of trust between such unknown issue and some known and trustable issuer you have to decide about your trust, that issuer of a certificate. So should you believe and trust or not? This is **up to you**, there is no general rule and all suggestion is depending on the level of security you want. You can consider particular settings when you can imagine you are developing some application, the application is working in **your network**, protected by outside, what happens is happening internally. What prevents you from creating a local certification authority issuing digital certificates to be used only locally and since all the host in the local network are knowing such authority they are accepting the digital certificates of such authority. This is not requesting the need of registering the identity of some certification authority.

It means you can have 2 types of certification authority, the **official one,** registered in some office, and the **unofficial one**, the one you are just creating in some protected environment, because all the participants in the environment are knowing the public key of this certification authority and maybe you can hard code the public key of the local certification authority in the main host of your network. In general digital certificates issued by an unknown issuer are considered **useless** if you don't know if you can trust, closed world assumption means can you trust? No secure you can trust? You **don't trust**. A consequence of the closed world assumption.

Other important details are associated to the keys that are used for public and private keys because you know that after time is passing you need longer keys, so a digital certificate **should expire**, you can't think you will use forever digital certificates because if today the length of the key used for a digital certificate is **good enough** to consider secure the digital signature from attacks maybe after 20 years the new computational power from the attackers will be making such keys weak, so it's not good to design to issue digital certificates valid forever. This model will ask more money for a **longer during** digital certification but we can't expect digital certificates during more than a few years.

What is a **digital certificate?** The standard X-509 is specifying how a digital certificate should be prepared, and this is what you see here is just is a general scheme describing the structure of digital certificate. Anyway there are already some nice information to know. The digital certificate is containing some **standard fields** and the role of any field is well defined, it's true that the version 3 of the standard is allowing new fields, flexible with the possibility to make extensions, you can have digital certificates with some extensions, providing further information in digital certificate, you may ask why, all what I want to know is a public key, a name to be associated with a public key, then I need to see the digital signature to the statement, why I would need **some more information** present to digital certificates?

There are several details to practical management, for instance every certificate is having a **serial number**, a unique number, if you say the number this is unique within the digital certificate of the same certification authority, there are other information we will be discussing in the next slides, but we can just have a look to the simpler information, **distinguish name** is the name of the certification authority, **validity** is defining the interval of time when the digital certificate can be considered valid, this is providing a nice question, what happens if I make a digital signature today? My digital certificate is valid, next year will be expiring. If somebody checks my signature next **after date of expire** what should be the result of verification? Good or not good?

When verifying the signature you should know the interval of time when the certificate of valid but also when the signature has been made, so that you have to check that the **date of the signature** was internal in the interval of validity of the certificate. In many cases this is requesting you store with a digital document maybe a contract, you have a digital signature, you have to store a digital certificate, after 10 years the certification authority is no longer existing, you cannot ask to a certification authority give me the public key of that guy 10 years ago, you cannot ask that. It's important to see the **need of storing** for many years digital certificates.

In Italy there is an office that is requested to **renew digital signatures** on the document. The document is permanently stored somewhere and after every 5 years take the whole and signs **with its signature** the whole and then after other 5 years signs again with the current standard, so that many signatures for the same document for permanent storage of official document you store. There are lots of problems on **long storage of documents**. Pdf for an official document? How do you know pdf will be used after some years? Maybe will be obsoleted, no longer viewers will be existing. In next generation I don't know what it's going to happen. This is why they invented a special version of pdf, named **PDF/A**, a subset of pdf, its self describing format, it means that at any time you can write a program reading a format that can learn how to interpret the content of the document.

Name of the user, another field, public key of the user because this is the focus of digital certificate, public key and information about the algorithm, if you use a public key for RSA that is a particular type of public key, has a corresponding private key, but in RSA the relationship between the private and public key is different so you need to **explain what is the algorithm** for making the signature that should be employed for that public key. Issuer unique identifier, a code associating with issuer, subject unique identifier, extensions, some of them will be discussed in a short time, all such information is signed by the certification authority. This is very interesting, why? The digital certificate is signed by the certification authority. We already knew that, you want to check the digital signature of certification authority, maybe you are under attack, the attacker has provided to you a fake certificate, you need to check. **How to check?**

You need the public key of the certification authority. Otherwise you cannot check. Who is giving to you in practise in real life the public key of the certification authority? **Another digital certificate?** Ok there are many certification authorities. So, for checking the signature of a certification authority we need another digital certificate, signed by another certification authority, but I want to check that sign. You understand is generating an important problem, how to be sure the process is **converging** and you are not going forever for a certification? Very interesting problem we will be facing in the next slides, maybe discussing it on Friday. Let's just see with some more details some of the fields, I don't want to talk in details about all the fields. Ok, there are **3 versions of the standard** and version is the first field of the digital certificate and current version is version 3.

Serial number is an integer number, concatenation of the name of certification authority and serial number is unique. The field signature I think that I don't remember but I think they are in the same order, that is the third one, the field signature is saying what is the algorithm used for computing the signature on this certificate. The certificate is signed so it's important to specify the algorithm used for signing this certificate. **Issuer**, the name of the certification authority creating the certificate, not just a name, a standard name described by the X-500 protocol that is also stating how to write names of digital entities. This is very complicated and I've already mentioned I don't like too much these details, I want to say the important part for digital certificates. X-500 looks like something like that, there is a country in this particular example country is US, there is organisation and organisation is meant to be the company name or entity name or whatever it is, OU is the organisation unit, particular organisation that is associated to a digital certificate, this is the common name, like a nickname used for describing the subject in a short name.

There are **many rules** about what types of names are allowed, what type of names can be sub-name, OU is allowed under the name present in O, and so on. All practical details mentioned in the standard I recommend you if you need such information just go and **check these details**. What is nice to say is that it has **never invented an official way** to describe an X-500 name. So different browsers, different clients for email, different OS will show the same certificate, the name of X-500 names in different ways. You cannot expect the same X-500 name is shown in the same way by different products of different brands. I don't know why they never agreed about official format.

Validity the validity field is actually containing 2 subfields, one subfield is not valid before, this date. The other is not valid after, validity is defining an interval defined by 2 dates. Then there is the X-500 name of the subject, the next field is the public key info about the subject, 2 subfields, one is just the public key and the other is an identifier for the algorithm meant to be using such public key. Ok, there are some identifier for certificate because they are registered in public registers, having a **unique identifier**, showing certificates. This can be present also for what concerns the subject. This is redundant, what is interesting is the presence of extensions, because a very important problem that is related to the **man in the middle attacking** people, is related to the **revocation** of the digital certificates, because the certificate can be revoked for several reasons, the certificate is valid until some date. Why the subject was having the certificate?

Maybe he's an officer and among his duties there are the **need of signing some official documents.** What if this person decides to **change work?** He's having a digital certificate but the organisation should prevent this person to sign further documents. Just because he decides to change his job. Digital certificates should be revoked. When you are checking your digital certificate not only you have to check the certificate but you should also check wether the certificate is revoked or not. Valid and not revoked, this is what you need to know. At some weak ways for checking for revocation are shown in the part. If such information is not shown there is some official way for checking for revocation but it's longer, more computation and more time, introducing **delays** people are not liking.

We have discussed the content of a certificate, now we can talk about how Alice and Bob can check the validity of a digital certificate. What if the 2 parties are using different certification authorities? In general certification authorities form an authority, like that. This is Alice, this is the CA used by Alice, this is the one used by Bob, this is Bob. In this digital certificate the **issuer** is this one, Alice CA, subject, the owner of the public key is Alice, a field saying ok Alice is not a certification authority, the signature of certification authority of Alice, this is what such certification authority is issuing for Alice. Now what it may happen is that the certification authority trusted by Bob is issuing a digital certificate for what subject, this other certification authority, in this case this certification authority is a **client of this other certification authority**, the digital certificate is saying the issuer is Bob certification authority, the subject is Alice CA.

The public key is the public key of such certification authority, the field about the type of the subject says this is a certification authority, all **signed by the certification authority of Bob**. The digital certificate of Alice is possible being to travel along the hierarchy of digital certificates when you have a relationship like that. In general this is not always happening, so in practise it is true that you have a hierarchy of certification authority. This notation here means that the certificate of Bob, has been issued by this certification authority A, you can establish a **certification path**, this is user B, and certificate has been issued by certification authority Z. Z is offering a digital certificate issued by Y, another certification authority, Y is offering a digital certificate, there is a **chain of certification**, in this chain is linking in some way W with B and making possible to accept digital certificates issued by unknown authorities, is sufficient there is a path of certification, am I sure there is **always a path of certification?** In any case a path?

If answer is yes I'm not convinced, the path should be infinite, when I stop? At some time the path should stop, there are not infinite certification authorities, even if all the authorities are building a trust so you can see a unique path, this is not the reality I'm just simplifying, you suppose this is a certification authority, this is another, and so on. **You cannot go infinite**, you have to stop at some time. What about the last one? You have 2 type of certification for the last one, **self signed certificates**, self signed, the authority is not only signing, also signing a statement saying what is its public key, you have a case of self sign certificates or **mutual trust**, the case where certification authority A trusts certification authority B so it's issuing a digital certificate and vice versa, they are providing double certification, using these 2 tools you can face on the practical case you need in practise, this is making very difficult because if you want to check a digital signature it's possible you need to traverse all the paths until reaching the certification authority you trust, then you can stop. And how to **check the whole path?** You need to ask for further digital certificate.

There are 3 cases of **certification revocation**. The private key is compromised. If you think your private key is compromised your certificate should be revoked. What about a digital certificate? All the signature made before the date of compromising are **still valid**. All signatures made after that date are no longer valid, revocation can be having the current day but I can also find a report saying ok I understood now that one week ago my private key has been compromised, in that case revocation is having as date revocation **one week ago**, so all activities the usage of digital certificate is no legal. Another possibility is the one described, some **people changing the job**, the digital certificate should be just revoked. The other case is when the **certification authority is compromised**. In particular the private key of the certification authority, all the digital certificates issues by the certification authority should be revoked, it's easy for the attacker to have the private key to sign digital certificates, so constructing fake digital certificates.

Digital certificates (end)

Good morning everybody. We can continue our discussion about **digital certificates.** You remember last time we talked about certificate revocation. We insisted about the need of revoking certificates for several reasons. The first is that certification authorities normally do that as a **business** so they want to be paid for issuing digital certificates, so in the business model is important that digital certificate are expired. There is also a good practise because under a security point of view maintaining for a long time the **same keys** is somewhat helping attackers.

Changing the digital certificate normally means changing the pair of keys and while time passes we can get longer keys. Other reasons are the private key has been **attacked**, and this is very important to in this case is important to be effective in revoking the digital certificate starting from the date where the private key has been compromised. Another reason is somebody that is having the certificate because he is acting in some job, doing some function, for some reason he's changing the function or **changing the job**, it's important the digital certificate is revoked by the certification authority. There are a sort of **certificate revocation list**. Such list you can think of a list of certificate revocation like something looking like a certificate, maybe the format is not exactly the same as the format of digital standard certificate but is organised in **fields** and is important to say that a certificate revocation list is containing a list of serial numbers of certificates that have been revoked.

This is very important, certification authorities are publishing a revocation list and if you want to check wether some certificate is still valid you need to **examine some original source** of information, so one possibility, the original one, that was meant when designing system using such revocation list. The document that is published by the certification authority is having a **format well defined**, specified by the same standard specifying certificates with public keys. There are some fields, many of them are having very intuitive meaning, what is important is this part here, there are several entries and there is for every entry you will see serial number of some certificate, the revocation date and some **extension**, extension are standard introduced in version 3 and they are allowed so if you need to publish further information associated to such certificate you can use such extensions, for instance you can use some tag, some keyword, you can use it for inserting URLs, and other sources of information for the analysis of that particular case.

Is important to say that the last field is the **signature**, made by the certification authority on all the content of the digital certificate. Here we have a list of fields about such fields are the same we have examined so I don't want to talk again about that. There are 2 interesting fields, somewhat new, there is a **date** informing when such the current list has been issued, so when the certification authority is publishing a new list this field will be the current day and there is also another possible field that is **optional**, informing about the new date for the next publishing of another up to date revocation list. You have some many cases some well defined information about the issuing of revocation list. Keep in mind that revocation list is a **big document**, not just like a standard X-509 certificate because contains many entries. This is considered to be like a drawback of this way of informing the users about revocation of certificates, managing such revocation list is not so effective, is somewhat **demanding** in terms of **resources**, delay while making connections.

The **usual entries**. What I want to say is to just mention a **more modern approach** for checking the validity of a digital certificate. Of course you can use revocation list, maybe when your browser is connecting to such website using the protocol **https**, the website is identifying itself with a digital certificate, your browser will want to check wether such a certificate is valid, modern browsers they have built in by default the ability of checking, I remember when that was not the default, you had to configure the browser so that digital certificates should be checked every time. What is important is that a new protocol has been proposed this is what I mention if you want you can easily find information for such protocol, this is somewhat alternative because when the digital certificate you are obtaining from the website, I'm talking about websites but in general is for **every type of authentication** based on digital certificates.

Is possible to use information from digital certificates, version 3 is allowing some extension and one of these fields is the URL where the browser can connect for **using some services** provided by the certification authority for making a query, the browser is not looking for the entire revocation list, just making a query to the certification authority, done according to some online certificates status protocol, most browsers are implementing this protocol. Ok, this is making **much more effective** the process of verifying digital certificates, the browser will make a connection to the URL provided by the certificate and will get a response about the validity. I want to say 2 things, one thing is if you think of the whole process, the certification authority, think of a big certification authority, issuing **many important certificates** for important websites, having thousands of visitors every minute, you know there are many websites like that. It means that the certification authority is getting many connections from many browsers, asking for digital certificates.

The certification authority can collect such data and can profile people, is able to associate information coming from the connection, at least the IP number but not only that. So, the certification authority is able to check and **profile all users** connecting to the websites using certificates issued by the certification authority, this means a concern from the beginning and researchers have addressed this concern and proposed some changing protocol, updated protocol and this is known as OCSP stapling, keep in mind that in the web digital certificates are the main tool for making authentication of one party, because when you connect with https protocol your browser will **authenticate the server** because the server is offering information about itself based on a digital certificate. Your browser will check such information, the server is implementing a sequence of steps, we will see such steps, we will study soon the TLS protocol, and by using these steps based on a digital certificate the browser will be able to authenticate the server.

Normally you can think that you could do also the **opposite process**, I mean identifying your browser by using another digital certificate. Is true, you can install in your computer a personal certificate so that you are **authenticating yourself** when connecting to the website. In many cases you have very fast authentication based on the fact that both parties are using a digital certificate, in this case both parties will be checking the validity of the other digital certificate. Also I want to mention another problem, not present in the slides but is very important, when you are connecting to facebook.com maybe you are young, you don't know until few years ago the connection **was not based on https**, it was a connection plaintext, that is the past. Today you connect to such website what happens when you connect to the site? Your browser is showing you information about the security of the connection. In some part, the left, you will see a lock using some colour having a meaning, blue colour, green colour, in some cases you just see a lock, in other cases a lock and a name. What is the difference? That is **important difference**.

When you offer your service, you want to setup a secure server, you ask for a digital certificate, when you ask the certification authority for your digital certificate, what is the identity you want to be **associated** to the digital certificate? Your name? The name of your company or just the name of the domain associated with the server? **The less expensive** digital certificate is the one associating subdomains to public key. The association is between the subdomain and the public key, you can check such a certificate, if check is ok the result is valid, the browser will show you a lock, connection secure. Who is the guy running the service? You have some tools for making official queries in the web. From the digital certificate you just obtain information about the association between the **subdomain and the public key**.

In other cases the digital certificate will be offering more information, containing also an official legal representative for the service, you will read the name of facebook corporation, google inc or whatever, such certificates are more expensive of course, for several reasons. One is just because you know you can offer a less expensive service and more expensive. Another point very important, when you ask the certification authority for a **simple certificate** the whole process can be run by some remote protocol, I mean you don't need to go into the office and show your ID card and so on. So they are just checking that you are really originated. If you want a name, you have **to go into the office** or use some strong authentication so they can give deeper information about your identity. So in most cases you are going to the office and use stronger authentication protocol, not simple to be used.

The second type of certificate is called **extended validation**. EV. So you will prefer to use for your server in the web an EV certificate, also because you know in the web there are so many organisations collecting information about servers, organisations, who is offering EV certificates will have a **higher reputation**, this doesn't mean you shouldn't trust normal certificate, but the organisation behind the certificate is not so deeply caring about transparencies, about evidence and charging some organisations about responsibilities that can be associated to facts, incidents that can happen to whatever web application. So, you should prefer **more expensive certificates**, browsers give evidence of such particular type of digital certificate. We should keep in mind that a digital certificate is something you can use also for identifying yourself, you can install your digital certificate in your browsers, your operating systems and so on, this is very important.

I had in my mind another point to discuss, I apologise I forgot it, I want to mention about a different, ok I got it, sorry. When you are asking for a digital certificate and you can also explain what it will be the usage of the certificate, you can ask for I mean you can generate a pair of keys, you can obtain the certificate for the public key and then **what is the use of such keys?** Web server? Digital signature? Both? Authentication of a browser? Other protocols that allow authentication based on digital certificate? Normally you get a certificate **for one usage**, so if you want to be able to make digital signatures and to set up your web server you will need 2 different digital certificates. Such information about the usage is written in some fields of the certificates, version 3 is allowing extension fields, in some field there will be written. If you want to have multiple use of a pair of keys there are 2 points to say. The authority will ask you **more money**. Second, this is less secure, is a good practise to associate your pair of keys to one type of domain. For digital signatures, for web server and so on.

What happens if one of the application supporting multiple use is **compromised?** You will be compromising all the usage of the keys. This is not a good idea. You need different certificates for different usages. This is very important. As last slide I want to discuss the **PGP**, actually is something allowing to implement **secure email**, used at application level, the study of PGP is done in web security and privacy course. I want to mention a very interesting point about PGP. The idea of the creator was **I don't like certification authorities**, they can be attacked, we all need so I prefer a distributed approach. He proposed the so called **web of trust**. Web of trust is something that is proving, let me say, the correctness of the association between a public key and an identity, not by a digital certificate but by using the **trust** obtained from other users.

You can think now I'm speaking in some theoretical way, I decide to create a pair of keys for myself, there are many software allowing that. Then I show on my website my public key. And I ask for visitors please this is my public key, import your public key key in your keyring and sign my public key with your digital signatures. **Key-rings of PGP** they are allowing to import public keys of people, normally when you do that you import a public key associated with an address. In some cases also a name, but the **strong association** is between email address and public key. You can think on what you can do, you import some trusted public key into your keyring, you can decide that the most trusted you trust so much some of them, you decide to sign by using your digital signature such keys. ok, other people don't know about your action, you sign some keys, what happens is that there are PGP key servers, public PGP key servers, you can **upload your keyring**, what you have signed in particular, to some PGP key server. You don't need to upload on all of them, you don't even know, there are many well known, the best known is the one **from MIT.**

You can upload your keyring so that when you need the public key of somebody because you want to send him an email message and you want **confidentiality**, you need his public key, how to get it? You can try **several paths** but you don't know if you just heck for the web you may find some website offering a public key. Different approach is connecting to a PGP key server, you make the search, you get the result. You also see the list of identities that have signed such public keys, so you get a public key if you see no sign at all or maybe 2-3 signs you are not really sure about that. If you see a very high number of digital signatures you can think is **trustable**. You will trust such public key if among the signers there are people that you trust already, so you understand the **problem is starting**, after the process of signing the public keys has been started in a good way you can check and decide what public keys you can trust, you find that people you are already trusting have decided to sign such public keys. This is building what is called the web of trust, offering people the possibility of generating locally generating public keys uploading the keys to the PGP server and start a process of digital signing the **trusting one.** Was proposing the key sign parties, if we decide all together let's go to PGP.

When you are looking at somebody participating the web of trust and you have to decide if you want to trust such person you have to take 2 decisions, do I **trust this public key?** First decision. Second, do I trust him as **a good signer** of public keys? Is not the same problem. A trusted person can be very naive and decide to sign many keys. You can trust someone but not what is signed by him. It's also possible to generate keys for revoking the PGP keys and so on. They are inter operating and they are **good solutions**, 2 years ago the guru of security wrote an article, he told the PGP is dying. We don't need anymore PGP. I'm not completely agreeing about that, I just reported the. Myself I'm using PGP in my email and in many case I'm attaching my public key to every email I'm sending, it depends on the client I'm using. I invite you all to see PGP and decide if you want to implement PGP for your email, that will be mandatory for taking the exam. Any question?

Homework number 4

Before starting the new pack of slides I want to talk about the new homework. The rest on the recording of 11/17 because I'm lazy and I'm not doing homeworks.

Authentication using passwords

Let's talk about passwords, there are so many jokes about passwords. I'm tired to be sitting all the time. Let's just collect some money so we can buy a wireless microphone. Before going into the formal details of the content of our course about passwords let me start by saying some **simple facts**. I assume all of you are very familiar with passwords. Did you ever try the google search most used passwords? You get many results. How such results are provided? Because after some incidents, some web server being compromised password in the web server are exposed, you can collect them and make your statistics about the passwords.

I want to ask you if you know the service **pwned**, there is a service you can connect to a web server and you can make some queries, the server is based on the fact the creator of the server is collecting all informations about all incidents that are known and determined some data bridge about passwords and you know in many cases the result of such data bridge are published in the **dark web**, people can just download such information using tor browser and extracting big list of passwords, associations email - password from the dark web. You knew already, I guess you are familiar with that. So let me tell something about your web server, you make your web application, you need to authenticate your users, the most popular system is **password based**, even if passwords are **insecure**. Maybe you have some database storing a list of usernames and storing passwords?

You understand if you store passwords for your users you are storing sensitive information and in the case of data bridge or your server is compromised the attacker will get **all sensitive information**, in most cases usernames are just emails and passwords. This is very bad for several reasons, first if I attack a web site I get the association email password, I will try the same association for all the services in the web I can imagine, I can login to twitter, facebook, in whatever service you can imagine, today so many services are **based on email**. Don't forget that people that are not very confident and not familiar with the IT technology if they register some web site for the email when registering for the service of facebook they will write the email and when they read they have to type a password they **will use the same password** of the email system.

Ok, this is stupid, but not so stupid for people not understanding the difference of who is holding the password and what is the usage. This is also strongly related of the issue of **reusing the same password**. People keep reusing the same password. I do. I know that is dangerous but in some cases I reuse the same password. Also you know that, ok let me continue the previous topic. It's not a good idea to store a password database. You should perform authentication, you can store **encrypted passwords**, this is better of course, so when Alice is providing the credentials Alice will send the password by using a secure connection, you get the password, you access the database, you decrypt and match the password. Is this secure?

Not really, in case of compromising your web server is compromised so also the key for decrypting passwords is compromised, still no good. Ok, you can store some information derived from the password, maybe the hash of the password, so you get the password by Alice using a secure connection, you take the password, you **compute the hash** and check wether you are storing the hash. This is better, not so good, why? Because attackers they run what is called **dictionary attack**, what is? They take long list of words, coming from official list of words you can find in the web for implementing dictionaries, for instance one of the most used is the one coming from the project open office and libre office, they offer a list of words for whatever language. You **import text files** and can compute the hash of all possible words, many different hashes using I don't know several versions of SHA, still today applications where server side they are hashing password based on md5?

MD5 is broken, so you can run a dictionary attack and if you want to make it shorter you can use services like crackstation.net, they offer you the possibility of writing a hash and since they have made a very huge **pre computation** they will be able to provide you the counter image of the hash, the original word. If the word is invented that will be impossible. This is suggesting that passwords should **never match a word** of some language. In modern attacks the attacker is not only taking words from dictionaries but also computing combination of 2 words, up to 3 words coming from dictionaries by concatenating such words, so they will be checking some simple phrase, short phrases. Another point very interesting is ok I can anyway try to **brute force passwords**. What does it mean? You have a list of hash of whatever encrypted passwords and you can try all possible passwords. You cannot do that online of course because all services offering authentication after a small number of attempts they will **stop accepting further requests** of authentication, but you can run the attack offline.

In order to make the attack harder you can ask the user to choose passwords having many strange symbols, if they are alphanumeric for every character of the password you will have 26 small literals, 26 capitals, 10 digits, this is not much. If you add **extra symbols** like parenthesis or whatever you can enlarge the space of possible password making brute force generation **much harder**. What is nice to say is that security experts today they are saying something that is the **opposite**, we could read in the last 10 years. We read ok choose your password with capital letter, small letter, digits, special symbols, this is mandatory. They check your choice and they ask you to make a very complicated choice. Today modern experts are saying **no good practise**, why? Because it has been proved that choosing a password containing at least one capital, one symbol, is pushing people to start **insecure behaviours**. Insecure behaviours for managing such tough passwords, they keep them in the pocket, written or they keep it on a post it on the scree, so many insecure behaviour. Today there exists protocols to have good protection starting from simple passwords. I will be offering some recent articles, I will publish the URL of such articles in the log of our lessons.

Anyway going back to the **most used password** even if you make many searches you get many list of most used password they are different but somewhat consistent, the most used are 123456, 00000, qwerty, let me enter, password, you know. This is very interesting, under a cultural point of view the usage of some female names. Anyway, after that I will be saying what is in my opinion the **best approach** to implement a password based authentication. Even if you become the strongest security expert, so you will start saying don't use password, let's user some stronger authentication, the person who will be asking you to work who is paying you will say stop, that is **bullshit**, you should use passwords, my customers want passwords. If you have such specification use passwords, how can you do that? There are several security ways, this is a **field of research**, people among you can contact me for a master thesis, after the exam. Master thesis with me after the exam. If you want to talk about authentication we should remember there is a huge difference between **humans and machines.**

Humans will use short password and will use words coming from a dictionary. The attacker knows very well you can replace the o letter by a 0, you can replace the I by symbol 1 and so on. So one using a dictionary for every word the attacker will try some 100 attempts, 100 variation of the same word. There are several drawbacks for human beings, but also **trojan horse** and the experience with users is somewhat boring, also I want to mention that research done in the USA about 15 years ago. The same experiment was done in **2 different enterprises**, having similar numbers of host computers and users. The first enterprise they decided to push a password change every month. Mandatory. The administrator was pushing the password change, so users had to change the password every month. The experiment took 1 year and a half, in the other company the administrator made the same choice but also offered users a **button** that could be pushed and if the user pushed the button he can decide the mandatory password change. So, mandatory password change for both companies, but in one company the users had the possibility to disable such oppressing behaviour. What happened? **Very very few people** chose to disable the password change, first.

Second, in the other company no disabling possibility a lot of **insecure behaviour** started, like keeping a post it in the screen, on the monitor, people writing the passwords. in the company where people had the possibility to disable the password change more proper behaviours where used by employers. So, there is a **psychological point of view** very important, when deciding your policy about how users should some secure service never forget to consider the human psychological implications that maybe an engineer is not able to follow completely. Ask some expert about psychological issues. Password must **not be sent in the clear**, of course there is another important point, even if you are sending a password encrypted the attacker can be just storing the encrypted password, so he can try to authenticate the attacker itself by using the same password, just sending the same encrypted information. This means that what is sent for making authentication should be **different every time**, otherwise the replay attack will be easy for the eavesdropper, this is not so simple, in many cases I know several web applications where the same password is requested.

The same password is not having the same encryption during the TLS section, it means they are basing the security against replay attack not on some proper approach but on another tool that is the **TLS protocol**, if not used or successfully attacked then the password will be sent in the clear. Then he will get some encrypted information, 2 levels. Theoretically speaking you can ask for confidentiality, data integrity, for every need you have, you are enforcing 2 levels for all the needs you will implement something that is **not really performant**. You should find the good trade off, not so easy. Since in many cases what is really important is confidentiality you use 2 levels of encryption is very good, one level is provided by TLS, another is managed by the application.

Let's talk about the way **UNIX is storing passwords**, at least in the original approach. I don't know if you are aware about the fact that at the beginning UNIX systems were considering just **the first 8 characters** of the password, what is coming after the 8th char is useless. The password chosen by the user is converted to some secret key, at that time the encryption algorithm **was DES.** This is why they choose to use the first 8 characters, keep in mind that the design of UNIX was made in the USA and the encoding of characters was the ASCII encoding. How many bits in the original? 7 bits. ASCII is a 7 bits standard. There are **many extensions**, to 8 bits, but in that case extensions are not in many cases resistant. So, you can use 7 bits of information for every character, 7 times 8 is a number that is 56 that is the same number of bits for DES key, just adding some bits for implementing the normal DES key and the idea is ok, the user is choosing a password from a password I derive a DES key and then I compute DES of a sequence of 0, and again DES, and again, **25 times.** The result is what is stored at server side. In many cases this is just not the official DES but a very simple variant so that the hardware implementation of DES becomes useless. So you understand what is the process of authenticating, this is the one for UNIX, not even considering the problem of sending the password **over the net**, just using authentication on the workstation, you should think a UNIX workstation is having many users, it happens that at least in the first implementation of UNIX every user could access a file, /etc/passwd, I guess all of you know such file.

This is a text file, in this file there is a line for every user, in the line there are several information, the username, result of such encryption, path to home directory, what type of login should be used for such user and so on. In the passwd file is readable from every user, that can work on a local copy for running **several possible attacks.** In the more modern implementation linux systems for instance they decided to **remove** from that file the string that is the **encryption** of the password, other information still present. Encryption password is in another file that is not public readable, so they are preventing attacks. This is the approach for storing a password, since the beginning they decided to use cryptography for storing locally information about passwords. What **attacks** can be done against this file? Every user able to read this file can run dictionary attack, very easy, you get the DES key and try to encrypt the sequence of 0, then you can check for the password, in particular you may think the attacker may try to do that in a smarter way. If I'm the attacker I have a copy of the file containing all encrypted password, I take one word from the dictionary, I generate the key, I encrypt the 0 many times then what I do, I **check the whole list**, every word every string in the list of password for checking if there exists one user using that password, you understand that?

You can no need to check for every single user, you extract one word from the dictionary, compute the key and then encrypt the sequence of 0s, you check if the string is present in some line of the password file. If you find a match you find the username and the **job is done**. In order to prevent the efficiency of such type of attack **password sorting** has been introduced. What is that? When the user is storing the password because the user has just chosen a password the system is generating a random string, concatenated to the password. What is used is an approach where the encrypted data are encrypted in some process like this one, but also other using encryption in some way, encrypting the password. What is the consequence? Before analysing the consequence we should say that the salt is saved as **clear text**, so is known also for the attacker. What happens? The attacker can no longer choose a word from the dictionary and wether there exists some user that wants to use that password, because of the salt.

So the attacker should for every user takes a word from the dictionary, adds the salt and checks the encryption. This is **reducing the efficiency** of the attack, but it's not a complete fix of the problem, is just offering some constant factor to the time running of the attack. Still if the attacker is able to guess a user password the job is done, I mean guessable passwords are very easy to be attacked and can also try the **whole line attack**. So, we need some secure system, in particular in the net where not only we should be concerned about the user of the workstation that can take the file of the password but only attacks coming from the net. In case of authentication on net Alice wants to **authenticate herself** to Bob, can't send the password in clear, cannot try a DH exchange for establishing a session key because of man in the middle attack, maybe can try a challenge response handshake like the one we saw at the beginning of authentication, the eavesdropper can carry out a dictionary attack again. Alice can choose a word that is not coming from a dictionary but is very difficult and few people can do that, otherwise they can design a **strong password protocol.**

What is that? We are going to see it now. So, for designing a strong password protocol you need to use cryptography and you should keep into your mind all possible types of attacks the attacker could run, brute forcing, dictionary attacks, simple eavesdropping, replay and reflection attacks. Let me talk of **this approach**, at least we introduce the approach next week, this is a very nice and elegant approach, strong authentication based on password. The name of this approach is **Lamport hash.** Lamport is a researcher well known in the world of the security in the world of algorithms and also it is **the guy who designed latex**. Latex is having this name because La are 2 letters the beginning of Lamport, I don't know if I told you the story of latex name. One day a very important and famous researcher, Donald, decide to write a big encyclopaedia about computer science. He decided to write some number of volumes, I don't remember the number and designed all the table. When starting to write the first volume he said, what I should use for writing? After some analysis he said I want to design my tool. Whose name is **tex**, x is just a symbol the a greek letter that is the k greek letter. The tex system is a standard system for writing documents where you can write the formatting commands within the text, so you are writing a program. this is similar to latex, what you use to write your homework. The difference is that writing in tex is like writing **an assembler program**, latex is equal to using a high level programming language.

So one day lamport said of tex is **beautiful but too hard**. Let me introduce some macro offering users high level commands, each macro will be translated in the corresponding sequence of tex commands. This was a **good design** because all the community started to use latex, developed in a very beautiful way and had a big success in all the community.

This is the first reason I knew lamport, that I discovered he was a **great cryptographer**, he invented the one time password. How does it work? Alice is choosing a password, Bob is the server that needs to store the password, instead of storing the password Bob stores the name of Alice, a number N, and the hash of the password **computed N times**, he means hash the password, hash the hash, hash the hash of the hash of the hash N times. When doing authentication Alice sends the username and says I'm Alice, Bob replies by saying ok number N. Alice writes the password and on the workstation used by Alice it's computed the hash of the hash of the hash of the password **N - 1 times.** Not exactly N. This string is sent to Bob. Bob gets the strings, computes again one level of hashing and checks wether the result is the same as what is stored in the database. If yes, authentication is successful, if yes the line in the database associated is updated, N is decremented by 1 and the hash N times **is replaced** by the hash N - 1 times. So Alice is sending the new value to be stored in the database, replacing the old value, this happens only if the authentication is accepted. Every time Alice is sending a different hash. This is contrasting the replay attack, every time the password is different.

And of course this is based on the **security of the hashing function**, it's difficult to go to the counter image of some hash, you need a cryptographic hashing function. Even if the attacker is able to compromise Bob server what happens? The attacker gets this hash here, it's difficult to go back to the previous version, **no dictionary attack** is possible, you have to start from the beginning, this is offering a harder time for the attacker. But it's still possible to do some attack to the server. Indeed we will be including this approach, so the strength of this approach is sending one password one time only, so replay attacks are useless and the limit is Bob is not authenticated to Alice of course. I don't know if you ever seen what banks was doing with customers before token generator. For **internet banking** you have a token generator today. Before that the bank was giving the user a piece of paper containing password, every time using a password the user should delete it, next time use next password and so on. **One time password.** The many passwords used for the approach of one time password are the different hashing of some initial password. Just I want to mention that you can use sorting for this approach, you can store the password hashed many times concatenated with a salt, when N reaches 0 just change the salt, generate a new salt. Ok. Questions? I want to say more but no time, professor Lenzerini will complain. *Lenzerini*?

Common mistakes on homework

I just opened some contributions for a quick look and I found a common mistake. If 2 parties they exchange messages so suppose that Alice needs to send information to Bob when Alice is sending information to Bob if you want Alice sending also a nonce to Bob so Alice is sending information and nonce and maybe Alice is sending information concatenated with nonce using hash to enforce data integrity, this is not really powerful, this is a **weak approach** because you are not preventing Alice from generating in advance some days before many possible nonces, so that he manages to find collisions.

If you want to use a nonce for preventing collisions the nonce should be provided by the **other party**, now you use the nonce for computing your hash, so you have not the time for making computations for finding collisions, if you are originating information and nonce nobody is preventing you from doing pre computation so you can test several many nonces in advance so that you can find the one that are providing a collision. Just use the nonce the other party is providing to you, is different at Alice side if Alice is computing messages concatenated with nonce, this is Alice side, if the nonce is coming from Bob Alice is not having the time for finding a collision, if the nonce is coming from Alice Alice can do pre computation, this is **not secure**, just a little point but it's important, I hope have provided to you some insight so you can analyse in a better way your protocol. Secure against collision.

Otherwise Alice can cheat in the game. If the nonce is originated by Alice this is not good, Alice can do pre computation on nonces. If this is the message this is Alice, this is a nonce originated by Alice, can do pre computation, she can have lot of time for generating and testing several nonces, she can choose the nonce, if this is coming from previous message from Bob Alice is not having any time to do pre computation and finding collisions. This is just the possibility, I saw people using this approach, if the nonce is originated by Alice this is not good. You should consider the nonce as a **challenge**, you can use the nonce provided by Bob so that you can provide information describing.

Lamport hashing

Ok, we will talk again about this discussion and choices. Last topic we started was Lamport hashing. Providing a framework for making authentication in a more secure way, because the framework is implementing what is looking like a one time password, in this framework Alice is providing every time **different information** for making the authentication.

I want to remind the approach, this approach at Bob side there is a line stored for Alice containing a number N and the hash of the hash of the hash N time of the password of Alice so when Alice is doing authentication she is communicating the username, she gets the number N as a reply so Alice computes the hash of the hash N - 1 times and transmits this information. Now Bob can compute **one further hash** and check wether the result is the same as the one stored in the line for Alice, if yes authentication is considered successful and the line is replaced by another line, where **N is decremented** and instead of the value present the value of hashing communicated by Alice is going to be used. So for next authentication Bob is storing the previous hash. You see here a somewhat metaphoric question, why this sequence of hash reverse it? Why Alice is transmitting a hash and next time she will be transmitting not the hash of the hash but **one hash less?**

Just because the hypothesis we are using Alice is using cryptographical hash, easy to be computed, hard to be reverted. If the next password is obtained by hashing the adversary in case of compromission of one password can compute next passwords. Here in case of compromission the adversary **cannot compute the previous** value of the hashing, so it's hard to know the next value of the password. This is the meaning of safe against eavesdropping, adversary is not able to predict next password, even in the case of database reading the adversary will know pairs like this one, number, hash of the hash many times, so if we want to run a dictionary attack you can try it but you have this factor making your effort much more complicated, you have to compute many times and you have to do it for all the dictionary. In particular case that Alice is not choosing a word belonging to a dictionary the **dictionary attack is useless.**

Actually it's good to introduce **salting**, useful because the salt is allowing some advantages, one is when doing authentication you understand the first time Bob is communicating to Alice N, then is communicating N - 1, when N reaches 1 you **cannot longer authenticate** so you can implement the solution of regenerating a salt, recompute the hash for many times so that Alice should not need to change the password, you can imagine the process done in several ways. Also is possible that Alice is using the same password with **different instances** of the server, not using different salts, and again salting is somewhat making harder the dictionary attack because in case of no salt the adversary can take a word from the dictionary, can compute all the hash and check wether there is some user matching, in the case of salt the adversary should do that for every single user because he has to concatenate the word chosen with the salt. So it's introducing an **extra factor of effort for the adversary**, extra factor equal to the number of users registered in the database. This is **Lamport hashing**.

Even in the case of UNIX password you provide a password, some processing and provide the result, in that case you can have a collision. You can always have a collision but you have a low probability. There is one **attack to the Lamport hashing**, the n attack. You must imagine the case where the attacker is making man in the middle, so that Alice is trying to do authentication, the attacker is forwarding the message to Bob, Bob replies by some number n, the attacker forward the reply of Bob to Alice by changing the n and making it smaller, so the attacker is changing n and providing to Alice a smaller n, and the attacker knows that value, Alice will reply hashing a **limited number of times**, now the attacker knows what is needed to make authentication, just computes extra level of hashing and providing the result to the server. The n attack, **very bad**, because if you want to prevent such attack you have to adopt some strong measures, one possibility is that Alice is knowing the value of n, but this is a very hard requirement for Alice, she should remember this extra information changing every time, another possibility is using one client only. Because in that case the client used by Alice that will be able to store the value of n and check wether the next n is the good one. Can you push Alice using **one client only?**

Of course this is not good normally for web application, maybe some of you is using the internet banking application from several banks and those people know that you can install the internet banking app on **one smartphone**, it's not possible to install it on 2 smartphones, one only because the smartphone will be storing extra information for making more secure next authentication. If you have the power of pushing Alice using one client only, you can manage the problem of n attack, the client will be able to store locally extra information. If you cannot do that you have to **prevent man in the middle**, in order to contrast this n attack, for preventing man in the middle you need authentication. I'm doing authentication and I need authentication first, not easy. We will see next steps to fix this issue. Anyway this approach allowed the construction of the so called **s key**, token generators and they are providing the approach of one time password for secure authentication, at the beginning the token generator was not really advise, it was a piece of paper where the user had written many passwords, every time the user is making authentication **draws a line over** the used password, in some other approaches the password are just hidden by a layer of something you can remove by using a coin, and you can see the next password, but passwords should be used in the correct sequence, otherwise it will not work. This became into a standard offered in this request for comments.

The problem of **weak password** is a problem very important, so you can start approaching a methodology for improving the scheme for the authentication so that you add extra ingredients, try to contrast the measure I mean the issue associated to a weak password, I repeat this fact modern security researchers since short time, 1 year, no more, they are saying that it's time to **stop all the hard requirements** about your password choices, choose a new password, long, containing a lower case, capital case, digit, special symbol and should be different from some pattern etc. lot of oppressive rules that are making users adopting insecure behaviours, just improve the protocols for running the authentication process. Many organisations they are getting the old message of the need of generating strong passwords, a message that is 15 years old, they are getting **today this message** so they are starting today to push users to choose strong passwords, when this approach is considered too weak because of the problem of too many passwords to be managed.

A next step is considering a stronger approach that is the EKE, **encrypted key exchange.** In this approach you have a requirement, user and server are sharing a secret, this secret it can be also a weak secret, because next actions will be providing some more security, even in the case of weak secret, you can use the EKE approach for being robust against the dictionary attack, mutual authentication and generating a session key to be used in the messages. Let's see **how EKE is working**. This is the schema, there is a coloured part meaning that doing authentication, the non coloured is doing key exchange, let's see step by step. Alice is having some password, and there is some rule allowing to derive some information from the password, you see it by this notation here, W is a function of the password, you can imagine some hashing on the password. This is a shared secret between the 2 parties, to be honest Bob is not knowing the password, just the value. Alice does not need to know W, she just remember the password and every time she just types the password, that is hashed to the value.

The first message is sent by Alice that starts the conversation, is I'm Alice, encrypted by the weak secret, so you see by this notation we are in the fact the weak secret is used as a **symmetric key** for encrypting a message. The message is $g^a \mod p$. I guess you recognise what is $g^a \mod p$, where g and p they are constant values and a is a value invented by Alice. This is just **one step of DH**

authentication phase. Alice sends this encrypted message to Bob, now Bob can choose a random number b, also chooses a challenge c1, actually a nonce, and the reply from Bob is the encryption of a message, what is the message? $g^b \mod p$, the classical reply, the challenge, the whole encrypted by W, the shared secret. Both parties know they can build the session key computing $g^{ab} \mod p$. In order to construct the session key the two parties must **share a secret**, otherwise is not possible to construct the session key here. Now the two parties need the session key in order to confirm the authentication, Alice is sending to Bob the challenge c1 and another challenge c2, originated by Alice, the message is encrypted by the session key, and Alice expects to receive back a reply from Bob that is the encryption of Alice's nonce, made by the session key.

What happens here is that we are revisiting DH, introducing some extra ingredient so that we make **more robust** the approach against the man in the middle attack so that we can complete the authentication by running these messages. The main difference with DH is this approach is **not so weak** with respect to man in the middle, because the attacker is not expected to know the weak secret. This means if the attacker knows the password of Alice he can impersonate Alice, if the attacker is seeing the exchange of messages even if he's knowing the password he's not knowing the random number guessed by Alice, not even the random number of Bob, he will not be able to know the session key. EKE is somewhat strong as a protocol for making authentication. Not the strongest but we have done some improvement. Dictionary attack is ineffective. We rely our security on the robustness of approach and on secrecy of passwords and keys.

You see here, in many cases it's interesting to know information to be exchanged without making the two parties aware about the attack. This is just eavesdropping, carried out for a long time, they will never know about the attack maybe. This can be interesting in some **practical settings**. In that case even if the attacker is knowing the password he will be not able to guess the random numbers a and b generated by the 2 parties. Notice that the phase coloured here I called authentication, you could also say but do we need such phase of authentication here? Maybe the previous steps are already making the authentication between the two parties, this is stronger, because this authentication is based on a **strong key**, not on a weak secret. So we have actually 2 authentications, one based on a weak key and the other on a strong key. It's useful to have this last phase of authentication. The approach provided by EKE has generated a lot of follow ups.

There are many practical implementations, many of them looking very interesting. I just want to mention **2 interesting variants** of EKE, one is SPEKE, simple password EKE, the other is PDM. According to SPEKE, the idea is using W in place of g. What is the difference? You expect that g is known, because this is a DH exchange, we define what is the private and public key, the private key with DH exchange is just the secret number generated, other information. In this way you can use W in place of g so you are going to transmit W^a Alice to Bob, and then Bob to Alice W^b so that the session key will be W^{ab}. This is a simplified approach, providing experimental good results, but proving under a theoretical point of view a strong difference between the original EKE approach and SPEKE is hard, actually we don't expect a **real important changing** on security.

Another simplification is PDM, where what is interesting is that p is no longer a fixed number but it's a number that is depending on the password. So you can just fix your number g to be equal to 2 because number p that is the number we use for our mod operation is a function of the password, some function of the password. This is the password derived. So the message here is ok, EKE is a new approach to strong authentication, is providing some **interesting properties**, and there are several forms starting from EKE, like SPEKE and PDM, you can check for documentation about other possible forms. Keep in mind some important points, what happens if **the attacker is knowing the W?** The attacker is knowing W and could impersonate Alice, right? Even if the attacker is not knowing Alice's password, the attacker knows W and if knows W you see the messages, if he knows W the attacker can impersonate Alice. If the password file is stole, theoretically speaking is possible to run a dictionary attack.

What is the **password file?** Information stored at Bob side. For Alice is W. Indeed all researchers are talking about the augmented EKE, augmented. This **augmentation** is very interesting, just a nice idea. The idea is that Bob is storing some information that is derived from the password of Alice, such information is used for verifying the password but at Alice's side it's necessary to know the password. Otherwise you cannot use the protocol, so what is the difference with the previous case? If the attacker is knowing W he can impersonate Alice if we add augmentation we add some extra property, you need to introduce extra ingredients so even if the attacker is knowing W this information is **not sufficient** to impersonate Alice. No longer sufficient, the attacker must know the password of Alice. This is just an example for augmenting PDM. Remind that PDM is the EKE version where g is equal to 2 and the module p is depending on the password. What is **storing** the server for Alice?

The name of Alice, number p and 2^W mod p. W is again some hash of the password of course, in this case Bob is not storing the hash of the password, some information **derived from the hash** of the password. Also is storing p, but what we said the number p is derived from the password. P will be different from every user and the server is not knowing the mechanism for repeating the process of generating number p, it's not knowing the password. With these pre conditions Alice can choose the random number a and from the password can compute the weak secret W, the hash of the password, and number p, is the module, using some algorithm that is again based on some hashing, in this case the hash value is modified so that the final number will be some extra requirements, the best is when **p is a prime number.** There are many possible ways for managing such a problem, even if the number is not prime DH can work in a good way.

From the password Alice can compute the hash of the password and derive information p and now Alice can send 2^a mod p, and Bob can reply by 2^b mod p, this is for implementing the DH, where g is equal to 2. In order to make the approach stronger Bob is also sending this information, a **hash of the session key** and here you see two arguments. This is not fixed of the approach of the augmentation. Anyway Bob is providing this information, notice this is the hash of the session key and of a value here, what is this? 2^b times W, this is a value computed by Bob, but Alice is not able to compute such a value, she's not knowing b. B is known at Bob side. So, the reply of Alice is **another hash**, the other hash is you see here it means it's not necessary the same hashing function, even if can be very similar and the information here should be 2^{ab} mod p so it's the session key has been fixed and information 2^{bW}.

A possible **variant** is instead of using DH with the operation well known, you can be inspired by what is done in RSA, in particular when RSA makes the verification of a signature, you can implement it by using this solution, more complicated protocol, but we are very close to the end. Look at the drawing here. What is Bob knowing about Alice? He's storing the name of Alice, W that is the hash of the password, the public key of Alice, so we are introducing a usage of something inspired to public key cryptography, and he's also storing an information that is indicated here with Y, this is the **encryption of the private key of Alice,** the server is storing the private key of Alice, encrypted using W'. What is that? W' is another value derived from the password, you derive W and W'. It's good **they are different**, not to reuse the same W for the two different operations, because in case of compromission of W you can still save I mean you can still consider safe the private key of Alice.

Look at the **operation**, Alice starts by choosing a, compute W from the password and sends this message to the server, the name of Alice and it's an encrypted message, the message is $g^a \mod p$ encrypted by the weak value. Bob is able to decrypt this information. Now Bob is choosing a random number b, a challenge c and sends this complicated message to Alice, what is this message? Is somewhat encrypted by using W again. What is this information? $g^b \mod p$. We are converging to the solution. Another information is the encryption of Y you remember what Y is, Y is the encryption of Alice private key. Y is **encrypted again**, by using this value here, this value is the session key. The 2 parties are converging to this session key, $g^{ab} \mod p$. To be used for encrypting y. And then also a challenge c is being sent from Bob.

Bob is sending **3 components**, the encryption of this information that is allowing after Alice to build the session key and the private key of Alice encrypted twice, one time using W' so not even the server is knowing the private key, the second time encrypted by the session key. Now Alice is able to decrypt $g^b \mod p$, can build the session key and by using it can find what is the original value y, after finding y Alice can generate from the password the other weak secret W' and use W' for decrypting the private key so that Alice now is able to know the private key. Alice is not storing locally the private key, she **gets it from the server** and the server is encrypting the private key. And what the message Alice is getting is a double encrypted. Now Alice is able to send a message to Bob, signed by Alice by using the private key of Alice, and this is the hash of the session key and challenge. This is I would say the more complicated augmentation, so that the adversary cannot be able to attack the scheme not even if the adversary is knowing the W. If the adversary is knowing the original password he can impersonate Alice. This is a fail of the approach, if you approach authentication based on password you should understand that if the adversary manages to find the password he will be able to impersonate the user, this is why today in many cases we are considering **multi factor authentication**, what is that? Authentication based on 2 different steps. I think that many of you are using it with email or facebook, I hope you do.

Focus on protocols

Now I want to enter a new part of the course, actually this is a new part not completely new, we started our corse by saying we use **cryptography** for information security, by listing what main items the main requirements of information security and how to use cryptography for aiming at those goals. After a while we started seeing more complex applications of our cryptographical approach, we saw some protocols, not only authentication, even the last homework is just an example showing to you how by using simple messages you can setup something that is useful for one application, it's meaningful at application level.

The first **complex protocol** we have already seen is kerberos. A complex protocol because in kerberos the parties are exchanging complex messages and they are using session keys, they are using hashing and other interesting ingredients, ok. So we are getting a protocol, what is a **protocol**? Just the implementation of some approach, some idea, but it's more than just cryptography, it's also providing details about how to decide a key, in what format you should exchange information, when to encrypt or not, how to represent standard information and what encoding and so on. The protocol is providing a tool allowing the people to set up a design aiming at implementations of the protocol, still it's true a protocol is not providing all possible details, meaning that you can have **different implementations** of the same protocol, they could be not compatible, but when the protocol is well specified actually the implementations of the protocol should be inter operating, this is what we are aiming at.

So, a protocol is much more complex, when defining a protocol we are somewhat **closer to the goals** of information security, not so close to single operations of encryption and decryption, of using some particular ingredients of cryptography. Protocols are what we are using for employing in the best possible way cryptography and other measures, we can reach our security goals by using lot of stuff, much more complicated. If you want to study a protocol in a **deeper way,** maybe when it's complex like this case we will need I don't know I guess 40 hours of lesson. So big protocol, so many details, and the documentation is very huge. Now we start seeing some security protocols, this is the first one, we will see also TLS, but the idea is that we don't want to be able as a result of CNS class to develop an implementation of the protocol, too many details, we want to prepare all of you in **understanding what is the protocol**, why and when we want to use it, as designers you will be asked to make choices, so the committer will ask you ok I want you designing a secure application, a web app, make it secure. It's up to you to **make it secure**, you understand what secure is. You have to choose if you want to use IPSEC or another protocol, TLS.

What are the main ideas leading us to **some choices?** The main ideas so this is very important, is the beginning of the design. In this case we will be just users of the protocol, being a good user is complex anyway, you have to choose the protocol, configure it, the protocol can work in many possible ways, you have to choose the right configuration, this is stuff network administrators are asked to be that, if you want to be that you need **all the information** studied in this course, but also more detailed information about how to configure details of single systems. We want to be aware of the design principles of using a protocol. In addition, in order to use a protocol we want to be aware about what are the services provided by the protocol, what is providing IPSEC to me? I should understand the useful mess of this protocol.

One approach is easy, this is a security protocol, is it providing confidentiality? Ok I like it. Is it providing data integrity? Authentication? You understand this is a way to ask question on some high level and in many cases this is sufficient, for some general analysis you don't need to know what encryption algorithm is used for making IPSEC working, it's sufficient to know it's encrypting information, providing confidentiality and other services. On the other point of view you are computer engineer so you are expected to understand more **technical details** about protocols, if it's useful and if you need to develop some software implementing the protocol or some variation you need to enrich the protocol, open the source code and to **make your choices**, you need to have some starting point for going deeper into the analysis of the protocol or single algorithms, we need to know the architecture of the protocol, the main ideas allowing the protocol to work. In some cases you will need just the knowledge of the services provided by the protocol and how to interface your application with the protocol, you don't care about how the protocol is implemented, this is what engineer call a **black box analysis**.

A black box analysis is just some consideration about the protocol by observing what are the requests you make to the protocol and what are the answers to your requests, you don't know how the box is organised inside. It's **important** to do such analysis, sometimes I ask that on exams and I still see today that many people when I ask a black box analysis they open the box and tell me ok this works by deciding the key and so on, ok this is not black box analysis. Engineers should be able to understand when you need a black box and when you need an **open box analysis.** One of the keywords I like to use is being able to do use cases, you know what use cases are, you studied use cases in software design. Again you need use cases for understanding when you need a protocol. This is different, we are not designing an application, just thinking about the following problem, is this protocol what I need? Then I can use this **use cases** for the protocols, I hope that my usage of word use cases not resulting ambiguous to you, this is not the classic use case you do when you have to design an application, you have to choose a protocol, the user of the protocol it will be you, designing the application, any question?

We will see that protocols are **very complex**, the parties start in some cases talking, one party starts saying ok I'm able to encrypt by DES, AES, and you? Ok I know bla bla bla, how can we talk? If **no intersection**, no talk of course. They have to agree about some common ingredient for running the operations we need for our security. We will see that step by step. Now we start considering security by considering the **classical protocol stack**, and we have what is called the application level, the transport level, the network level, then we have lower levels. We are not considering lower levels, we are just considering from IP to higher levels. This is a classical architecture of protocols, actually in that when we are **not introducing any security approach** we have IP, and over IP we have TCP. This is the case where we are using TCP, with UDP there is some difference. The application level is using TCP, some application level protocol, making some request becoming TCP messages, becoming datagrams, sent over the network. The classical framework. You can introduce security here between IP and TCP or also you can introduce security here between application and transport level. 2 different approaches, introducing security here and doing it here, generally speaking you **don't want security in both levels,** there are some cases where you will need both. Now since I know all of you are very expert in networking, just by considering the question what is the consequence of introducing security here or here, what are the consequences? Your expect when you insert into the payload of a datagram they get some processing so they are **encrypted** maybe and the payload is encrypted. This is making the service we get. In this case what is the consequence? The traffic is encrypted, easy. We need to be able to make more accurate questions. We are also interested in understanding the possible attacks an attacker can try.

We have never talked about the **traffic analysis attack**. Maybe I'm not able to decrypt your transmission, not able to attack your data integrity, but maybe I'm able to understand you are using a browser, or maybe you are using your email client, or another possible type of application. This information, if you can classify the type of traffic originated from some target you are already collecting a lot of information and you should understand that, what is a **port number** for TCP? How would you define? It's a number of course, how would you describe this number? The ID of a service. Yes, this is somewhat the address, it depends if you are a server or a client. If I can see all port numbers I can guess what type of application you are using. This is allowing traffic analysis. Is the attacker allowed to see port numbers? If yes he's able. Even if I'm encrypted, so my information is confidential. You can encrypt all information and the attacker can run traffic analysis. Providing a lot of information. When the attacker starts what is called **deny of service** he will try to attack your network, he needs to know the type of packets going to the network, if he has run some traffic analysis he will know the type of packets and can address some sub targets. If 90% of traffic is of type A I prefer to attack type A traffic, if I want to run my deny of service.

The IPSEC approach is providing **security at datagram level**. Not considering application or transport level, you know how it works. When you have some communications all the messages are composed in many datagrams, every that passes through routers until it reaches the destination. Going router to router the datagram will be incapsulated in some other framework, ethernet or whatever. The type of lower level transmission can be different, from datagram above all networks will be using the same standards, this is the **revolution of Internet**. Implementing security on datagram level means ok we can operate at datagram level. When we are ensuring security for datagrams we are talking about one datagram. Whatever it means secure, datagram that is encrypted, having some primitives for enforcing data integrity, whatever. You know a datagram is composed by a header and a payload. If you want to encrypt information what you encrypt? The payload? The whole datagram? **You can't encrypt the header**, it contains the destination address, or other information like time to live. So you may decide to encrypt the payload, this is fine. You need to fix details about what is the approach of ensuring security over datagrams. What can we do? What are the **limits** of this approach? This is what we are going to study now.

Security at IP levels. If you implement **encryption** for datagrams you have extra effort for routers, they are receiving datagrams and retransmitting datagrams to next routers, if you use encryption or other operations you ask routers to **work more**, this is very important to keep in mind, also implementing security on the level of IP is not the only way of implementing security, when you are using your browser and using **httpS security** you are not using security at IP level, but at some higher level. What level? Ok, we will see that later, it's related to TLS protocol, over TCP. Also you can enforce security by using some approach based on the application level, I mean what if we want to have a digital signature on your emails? It is good, maybe you have digital signatures on email, I mentioned PGP, a tool to have digital signatures on email. If you want to **digital sign every email** you are sending this is a solution working at application level, it's the main client taking your message and computing all the stuff for making the digital signature and sending information over the net. You can assure security also by operating at application level.

We have **3 levels**, IP, transport and application level, and we can operate security at any of them, you have to be able to choose the best level for approaching the security you want. I would say that if you implement security at IP level you have some security tool that can be used by all applications, this is so important, this is the first design rule, you want to choose some security, if you want to provide **security to all applications** using the net you can use IP security, all applications using the net will be sending packets over the net and they will be IP packets. So it's **very different** if you want to make secure one application or all the applications. So you want to make your black Friday, you want to buy because it's black Friday, you go to amazon, you need some secure transaction, an application, make secure it. This is why we don't need IP security, that is providing security to all applications, since when we are doing that we are operating at IP level, it means that your operating system that is processing information being sent over the net when composing the datagram will try to compose the datagram using some security **cryptographically secure operation**, this is transferred to application, you can insert IP security, remove it, application will not know.

Is this good? This is good because you can activate or deactivate IP security, you don't need to reconfigure application, it will just continue working with or without security of course. Any question? This is **philosophy**, not technical details.

What is IPSEC for? You see here the **3 important security services** provided by IPSEC, authentication, confidentiality, key management. By **authentication** I mean we are able to understand who the 2 parties are, when transmitting over IP the datagrams, there is authentication between 2 parties, be aware about that, what are the 2 parties exchanging datagrams? Routers, but IP the sender of the receiver they are not changing, if I send a datagram from here to Australia my datagram will have my IP as sender, the IP of receiver as destination and from router to router the IP of the sender will be the same, the hope is implemented **incapsulating the datagram** within some framework implementing a different local address. This is very important, this means we are authenticating the 2 parties showing the IP number, source and destination. Authentication is also implying data integrity, being able to understand what are the 2 parties.

Also IPSEC is providing **confidentiality**, and we perfectly understand what confidentiality means. Since in order to provide this service we need to use encryption of several types we need to use keys, so IPSEC is providing mechanisms for k**ey management**, what is that? It's a standard for generating session keys for communicating, permanent keys, if we want to use them and so on. What I also want to say is that these 3 mechanisms are provided by IPSEC but doesn't mean I don't want all of them by IPSEC. In some cases I don't want confidentiality, just authentication, this is fine, I can have it. To have authentication I need again key management, because authentication it's using the key. Maybe I need confidentiality and not authentication. This is a set of **possible services** you can get from IP.

There are several documents describing the IPSEC standards, very complicated, in the next slides we will see something, is important that few years ago the internet engineering task force decided not to consider IPSEC mandatory for IPv6. The standard was also including as a standard IPSEC, so if you are implementing, using IPv6 you are also using IPSEC that is like a subset. After that time they decided that IPSEC is optional, so you can add IPSEC security to IPv6 in the same way you can add IPSEC to IPv4. To make it not complicated I will be **ignoring version 6**, because not to frequently used, just a generalisation of some concepts and the answer to some basic needs. We will be focusing on additional security for the IP level considering IPv4. Remember it's **no longer mandatory** for IPv6. This drawing is showing some first use cases for IPSEC. I see here 2 important examples, this network here is part of a local network, just a LAN connected to the internet.

Think of the case where you are managing an important organisation, having many buildings that are far away each other. Also very distant. Ok, you may like the idea of having a logical LAN, that is including all the buildings, how to implement them? A possible implementation is using IPSEC in this way, this is your local area network, also this is another local area network, you may think you don't need security because you have a lot of protection for the perimeter of the network, inside you are **already protected**, you don't need encryption within your organisation, we can assume in some cases, this is not a rule, in some cases we can assume we just use simple IP protocol and **no encryption within our LAN**, we are protected from outside. You can implement IPSEC in order to construct a link, between the exit point of one LAN end the entry point of the other LAN, they are gateways so they work as exit and entry point, both of them. This means when packets are going with LAN they are standard, when is sent **over the network** of the internet is protected by IPSEC.

Here starts the **protection of IPSEC** over the datagram, going in a protected way and then will be reaching the gateway of the other LAN, the gateway will remove the protection, reconstruct the original datagram, sending it here in the other LAN, this is a very classical framework, a classical way to solve the problem of constructing a secure logical LAN that is unique for a big organisation. Here you see already some details about the way about ensuring such an extra security, it's too early to talk about that in the next slides. So one use case is build a link between two physical LANs for having one only logical LAN, and this is called as a **tunnel** between the 2 LANs. They are linked by a tunnel. Indeed the concept of tunnel is a concept well defined within IPSEC. I guess that you as users you may know what a VPN is.

Virtual private network, is anyone using VPN for doing P2P? Basically is any of you that doesn't know what is a VPN? Should I explain what is a VPN? Ok, it's a service, I want to just describe what is the **user perception** of the VPN service. The user while using a VPN is sending all the traffic being originated by the computer to some remote device. The remove device can be wherever, it's just sufficient you can connect, there is a multi hope path from your computer to the destination, you will be building a connection that is protected under many point of views, encryption, data integrity and much more, all your traffic exiting from your network adapter is going to that server. If you try to analyse that traffic you just see encrypted datagrams, they are going to the remote server, what service is providing to you the VPN? You can imagine there is a tunnel, very strong tunnel that is containing all your traffic flow from your computer to the VPN server, while it is receiving the original datagrams that **cannot be seen** by the eavesdropper, when the VPN is receiving the datagrams he will do the actions for delivering the original datagram from the server itself to the real destination, but it will act as a proxy, will be changing the IP sender and the outgoing packets to destination will result into datagrams having as source address the address of the VPN server, if you are going to see some webpage the web server will see as IP the address of the VPN server, not yours.

A VPN server is providing **secure service** for confidentiality and data integrity and other matters of security and also acting as a proxy by representing you from your request over the network. Why am I talking about VPN? The other use case in these slides is showing what is a VPN, there is a case where user is at home, and he wants to connect to the network of the enterprise, if he wants to enter the network you need to be **protected**, enterprises are issuing several policies, all communications need to be protected, so you will need that all your packets going through the internet and reaching the enterprise should be completely protected so what you need is a VPN, from your computer to the gateway of the network. This is important use case, you want to make secure the connection, this is very important because this use case is just a part of many use cases. They are somewhat more complex, for now these 2 examples are enough. IPSEC is a very popular way to implement VPN. The most popular implementation is the **CISCO based VPN**. CISCO is a trademark.

Some benefits of IPSEC, you can setup your gateway in order to provide strong security for the traffic that is going from outside to inside, from inside to outside, you don't need to protect the traffic, you can just implement a security point on the gateway, this is ok as long you can ensure good security, not that obvious that your LAN is secure, if you remember in the top 10 attacks since many years you always see **trusted insider**. That is some colleague able to use the same network, it's an internal attack, not so obvious you can afford a completely free from security measures local network, it's good to say you can just implement such strong security at the level of a gateway, you don't need extra effort. You **cannot bypass** the gateway running the IPSEC because you have no possibility by using some other measures. The IPSEC is transparent to application, you can activate, deactivate IPSEC, the application will continue to work in the same way, they won't see any details. Also interesting because it can be transparent to end users, this is important, in many cases you have end users not to smart, so skilled in computer science, in IT, so this is very important and is a standard way to implement VPN.

You can setup your IPSEC so that you can select different types of security for different types of users in the network, much harder, requiring extra configuration, you can define some type of security. You can build a list of requirements. I have already talked about **practical application of IPSEC**, I don't want to say more about that. Also this is the typical slide practical application of IPSEC, enhancing electronic commerce security, this is a **fake.** I have never seen a practical implementation of e-commerce based on IPSEC, you have other solutions. The solution is based on TLS guaranteeing the security for one application, not all applications. Of course you can do using a VPN and a secure connection, but this is another you have other needs, for instance you can be in some countries where some public service is forbidden, you know that **facebook is forbidden** in many countries, you setup a VPN and you can make a secure connection to facebook using https, also using a VPN otherwise you cannot connect to facebook servers. Why this is fake? Because all educators keep saying that so I'm not enough to stop this.

Modern OS from normal computers are prompting for authentication, you cannot prevent such connection, also if you are able to hack your OS, there are lots of xml files that you can modify so that you change the behaviour of your OS but such actions cannot be prevented, there's no way to configure the system for not doing that. **Nothing new**. From the beginning of the history you have been tracked for whatever you are doing, not mentioning the tracking through the webcam. Last thing, the introduction to IPSEC. When you configure your IPSEC you can decide between some basic services. Confidentiality, you can ask confidentiality on or off, authentication, you can ask if you want authentication on or off, of course you can ask for both, and remember authentication means being able to authenticate the IP number connecting and also to ensure the data integrity. This is done by the **combination** of 2 protocols, inside the bigger container, IPSEC is like a big protocol containing lot of small protocols implementing many different services and for instance the protocol providing the service of authentication is named authentication header. AH.

IPSEC protocols

IPSEC is a protocol containing **several protocols**, authentication header is one of the protocols used. The other one is encapsulating security payload, **ESP**, used for encryption. You can use one of them, the other one, or both. The more secure, the more demanding in terms of performance, so it's not so easy you just decide to setup all possible security features and go forward just because you will observe a **decrease of performance**, because of so much encryption and decryption at many different levels. So, I don't want to repeat some general features of the protocol, we are now going to see some of them in detail, other features will be just mentioned because the topic is too big, indeed if you want to study all details of IPSEC maybe because some day you will be involved in some development of some new version, new **philosophical approach**, you see here some documents.

You have a document presenting an overview of the architecture, another document discussing the AH protocol, another one discussing ESP protocol, another one about key management. This is the classical drawing presented just to understand how the different topics are each one related, so **general architecture,** the two important protocols, the tools used for making such protocols work, they are protocols anyway complex, they will need some key management, some encryption/decryption and so on, then there is another document where some conventions are explained also lot of details useful for key management, I want to say that in our brief discussion about IPSEC we will be **not covering key management**, that will be some deepening you will need if you want to go into further details in the protocol, key management is a set of solution we have somewhat considered, new features presented but we have not enough time for discussing it.

This other slide is just information, you can see here other details, how you can easily guess the **protocol is dynamic**, is evolving in time, this means after presenting general framework the protocol from time to time there are new updates of the protocol, here you have a better list of official document but I know there are some newer documents you can check for further details. This means you know where to go if you need some deepening. Let's talk again about the **security features** we can get from IPSEC. I want you remind the basic philosophy of the protocol is existing just over IP. This means whenever you have some application using TCP protocol why TCP is implemented in order to **make request to IP?** Such request will be directed to IPSEC and then the whole incapsulated in a IP datagram. This means the whole process of protecting the IP layer is **completely invisible** to applications and to users, very important to be reminded, if your computer is using such a layer this will be intersecting all the network traffic that is entering and exiting from computer.

We will see it's possible to **reconfigure the protocol** so that some particular traffic type can be just set up so that it can bypass IPSEC if you want, while configuring you can decide wether you want to intersect the traffic or bypass IPSEC in some particular cases. Generally speaking this is the set of features we can see so it's good we can discuss all of them, access control, connectionless integrity, data origin authentication, rejection of replayed packets, confidentiality, traffic flow confidentiality. **Access control** means the capability of preventing some non authorised usage of the resource. It means that every datagram that is going to some destination and such destination is offering services of whatever type this can be analysed so that is possible to understand if the originator of the datagram is authorised to use the service. Another feature is the **connectionless integrity**, you should remind that when you are just considering the traditional TCP/IP framework you have datagrams sent over the net and the path of datagrams is not so perfectly defined, it may happen that a datagram is going faster than another datagram, the order they are receiving is **not necessarily the correct**, also it's possible the datagram is not arriving to destination, every datagram has autonomous life, there is no issue at IP level, this problem is considered by higher level protocol, I hope you remind all these details, at TCP level you can manage the resending of lost datagrams and of course you can have other protocols like UDP that is not concerned to have data integrity, some datagrams getting lost is not really an issue, but what is important is that for having such capability of having the data integrity the protocol **TCP is needing a connection**, it means there is an handshaking and while working the protocol is having a memory about the current state of the protocol, the protocol is defining several different states for the different moments of the protocol, of course the most relevant state is established but you have to setup some process for opening and closing the connection.

In general we have some **data integrity** using a connection, here they are providing other types of integrity, connectionless, it means no memory of other packets, so this is not completely true actually, we will see some details, but it's handled at some local level than TCP, so that we can be sure about some particular characteristics of the information being transmitted over the net by a single datagram. **Data origin authentication**, ok the authenticator, the originator of the datagram is recognised, there is a real authentication so that you know what is the identity of the other party, the protocol is robust against replay attack, so an attacker recording the traffic **cannot submit the same traffic** to some destination, there are some features preventing the same string of bytes being transmitted and understood by the other party. There is the encryption that can be setup, you may want encryption, maybe you are not interested in it, this is an optional feature of the protocol, and some confidentiality about the traffic flow, depending on how we make the two protocol AH and ESP working we can have more or less confidentiality about the **traffic flow**, what is analysed by the attacker that just wants to classify the traffic, not to decrypt it. Traffic for web server, web radio, whatever. The traffic analysis is one of the main activity that is used by the attacker while starting to focus on the target of attack.

In this table we have several lines, every line associated with a feature. Here we have 3 columns, we understand we can have the protocol like ESP with encryption only, ESP with encryption and authentication, be careful, **encryption and authentication** is providing some services that are **overlapping** to the authentication provided by AH. AH is providing also authentication, so now we start learning about the possibility of having some overlapping, we are not interested in overlapping the features, we can choose among such 3 possibilities but we can combine the features, we can use AH and ESP, of course is not so interesting using AH and ESP with authentication, we are repeating the protection in the same approach, not a double layer of protection, we will see such details soon. You also see that the list of features here is not associated with every protocol, so if you want to have some **traffic flow confidentiality** you need to use ESP, AH is not providing such a feature, if you want confidentiality you should use encryption, AH is not providing confidentiality, all of them are strong with respect the replaying of packets, data origin authentication is associated with authentication, AH is authenticating the other party, ESP with authentication also, just ESP with encryption is not making authentication, other features like data integrity is **associated** with authentication, we have check here.

I want to say you are not expected to just learn such a table and remind it, you should **remind the characteristics** of the protocol, from them you will be able to redraw the table, this is very important. Any question now? Let's see more ingredients about IPSEC. A very important concept used by IPSEC is the concept of **security association.** This is a critical concept in the protocol, it is a relationship between a sender and a receiver, a sender is sending a datagram, the receiver is receiving it. A security association provides information about this type of transmission, sender sending datagrams to the receiver and in order to understand what type of security should be associated with such datagram, we can use features in different, security association is a **one way relationship.** It is defining the features of sending datagrams from Alice to Bob, details about sending datagrams from Bob to Alice are defined on a **different security association**, this is very important. A one way relationship. All security associations are stored in a database in what network administrator call a database, not a real database, I call it just a table but ok let's stick.

There is such **official database** and this is well defined in official documents, if you are enabling IPSEC over a network interface you will have a database for inbound the traffic and outbound the traffic, you want to keep separate in different ways. A security association is identified by **3 important types of information**, what is called a security parameter index, a pointer you can imagine such an index like a pointer pointing at information technical information about the security we are going to use for this sending the datagrams. The IP address of the other party and what type of protocol should be used for such sending, AH or ESP. You should understand from this that if you want to **combine security features** by adopting both of them, AH and ESP, you will need two security associations, because the idea is you want to combine the protocols, let's just combine different security associations, we will see the combination later, for now we just focus one single security association.

Some technical details, if you want to have **bidirectional traffic** you need 2 different security associations, the choice in the case you want to encrypt information and you want to authenticate information you can easily imagine you can have a list of possible algorithms for the encryption and authentication and such choice is left to the administrator, the officer that is taking care of the configuration of IPSEC, I remind you what I told you last time, configuring in a good way IPSEC for your organisation is not so easy, it takes some time and needs an administrator that is having some skills about that, the first time you try that you will be making **several errors**, typically you will learn by your errors, the best way is setup it in your house, maybe you don't have special requirements, just standard ones, you can make your settings and understand what are your errors, using services and applications. The protection for outgoing packets is described by the **security parameter index**.

It looks like a port number, and you read here that is somewhat enabling the receiving system to understand what is the security association under which the packet should be processed. For such particular type of packet IPSEC should be skipped, this is understood by the security parameter index. We are having some **similar possibility** for incoming packets, remember we have 2 similar database for outgoing and incoming packets. Let's skip this for now. Parameters, we see in details the parameters defining the characteristics, we have a counter, that is like a counter used in TCP protocol, there is a field in the protocols AH and ESP that is storing such a number, sequence number like just TCP, there is a flag that is indicating wether of course what happens when the counter is reaching the **maximum value?** Restart from scratch? No.

Theoretically speaking if restarts from 0 maybe is still existing some other packets having that number, when reaches the maximum you have to regenerate some details of the security association, you decide wether the reaching of the maximum submit to the network administrator somewhat looking like an alarm. There are information about the **anti replay windows**, a mechanism used for avoiding the possibility of having attacks. Other parameters are information about the protocol AH or ESP, of course if we are using AH we want to know about the authentication algorithm, what key is employed and other parameters. Keep in mind authentication algorithms practically speaking mean hashing or HMAC or other approaches.

What is the algorithm, if there is the choice again we have some keys, lifetime of keys and other similar details. What you need for making a good encryption. You can have somewhat like AH with CBC. Possible features. We need some detail about the key and the lifetime of the key because there is a security policy regarding the usage of keys, belonging to the key features of the protocol. Last parameters describing security association they are the lifetime of the security association because to be secure we want that a SA is not working forever, somewhat expiring.

Some details about the way you want to implement the protocol, **tunnel or transport,** we are not really ready to understand the difference, for such details wait a few slides, is important to know about the MTU, the size of the maximum packet, if you are having big packets then IP is fragmenting packets, since there exist several versions of **fragmentation attack**, maybe you don't want to allow packets to be fragmented, you know what is a fragmentation attack? This is a nice attack, fragmentation attack is that attack of typically fragmentation attack is aiming at implementing **deny of service**. I send information about a big datagram fragmented, to implement that the IP protocol is just offering some tools for handling simple fragments, so there are flags in the header of the IP informing that this is a fragment, last fragment, there are some offset showed in the header of IP in order the receiver is able to reconstruct the original datagram.

Fragmentation attack is based on the fact that information provided for describing the current fragment are failing, with wrong information, wrong offset, so that the algorithm collecting all fragments will crash. Crashing will mean crashing of the IP layer of the destination. So fragmentation attacks they are dangerous and typical type of attacks that can be run you understand is an attack run typically **within a LAN,** not just over the internet, technical details. There is another database, not only the database storing the security association, there is a very important database called the **security policy database**, this is a very important database, it is containing the first information about the fact that for some datagram you want IPSEC protection or to bypass it. So the database is containing entries for making differences in the packets defining the current traffic, each database is associated with a set of IP numbers and a set of upper layer protocols, that maybe using from above such protocol, they are called **selectors.**

Each entry in the database is pointing to a SA for that type of traffic, and don't expect you have a 1-to-1 association between entry in this database and SA, you can **reuse some SA** for different needs. Such selectors they are used for implementing what is called **filter**, so you can analyse the pattern, the traffic against you can match against the filter and you can understand what is the interest of SA or more than one. In order to realise this type of filter the process is like this one, compare the values you are reading in the several fields in the packet against the pattern you are storing in the line of security policy database. When you find a match it will tell you what is the correct SA or maybe not SA meaning bypassing of IPSEC. Once you have understood what is the SA for the current packet you have to understand the security associated parameter index, obtained from the SA and then you have execute what is requested by the policy for that current datagram.

The implementation of IPSEC is **heavy on the system**, you are analysing every single datagram against some database containing policies and you understand for each datagram what you want to do. Why IPSEC is considered heavy. Doesn't mean we don't like it, we will be using it just in case we have a good motivation for using it.

Is defining policies that makes you able to understand the treatment for every single datagram, the datagram in order to understand the processing the datagram will be compared to the selector, so that is possible to understand if there is a **matching**, not so different from what is happening in a firewall. The policy database is informing you about the process for every single packet, after that you have to process a packet, what details? You find such details in the SA. In some cases keep in mind the concept of user ID is not always defined, if it is is coming from some upper layer. In some cases such information is added to the several layers of protocol layers in order to enforce particular policies for particular users, there are other **technical details** will be able to understand better in the final part of the course that are associated with access control, indeed maybe in the last week of the course, access control is somewhat if you want to understand what is of course you know the **system of permissions**, for the owner, group and others.

You can generalise such approach and associate with every single resource of the system, even cpu, network adapter and so on, you can define who is having the privilege to use it, with what mode for using it, access mode, read only, write or grant permissions to others and so on. All this stuff has been developed in some **military framework**. In the selector for choosing the security association. Other simple information, transport layer protocol, what protocol for the network, what ports to be used. You can use each of them, AH or ESP, in a transport mode or tunnel. We must understand the difference. Let's imagine that we are analysing our security requirements and we understand we need confidentiality, now there are 2 possible ways of using ESP. they belong to 2 different philosophies, I want to say there is a **more secure way** and a less secure. Why not always using the most secure? Is **demanding** in terms of processing, the tunnel mode is good to mention that tunnel mode is the classical mode that is used for implementing a **VPN**. VPN is typically requesting the tunnel mode, most secure way, let's start by just saying some information about transport mode.

In the **transport mode** you can imagine you have some original datagram, you are wanting to send some datagrams, this is the original one, having a header and a payload. In the transport mode this is the original IP header, this is the payload and you introduce an extra header associated with IPSEC. It's important to perfectly understand the approach, if you do that imagine you are using an extra header, while you are sending the packet over the internet, the many routers they are processing the packet what are they seeing? A **datagram**, they don't know about it's a datagram using IPSEC. The routers will see an IP header and a payload here.

They will process the datagram **like a normal one**. The extra header is not processed at level of routing, when the packet is reaching destination the IP packet is delivered to the current protocol using IP and you should remember in the standard IP packet there is a number that is informing what is the particular protocol used at IP level, when you are delivering the packet to the correct layer of protocol using IPSEC the software will be able to process this IPSEC header. What is important to say, while you insert this **extra header** you are meant to protect this information here, the original payload of the original datagram. This is not completely true because there are some exceptions, when inserting such extra header actually you will be able to protect a small part of the IP header, basically you are protecting the original payload. This is a **classical solution** that is chosen when employing IPSEC from end to end. We should ok we will see again the **different use cases**, you remember them, we considered just 2 important use cases, we will be considering more, what is important to remember is that who is the one using IPSEC? A final computer used by some final user or a gateway that is the entry exit point for a network, implementing a logical link between this network and another LAN physically far away? Happening in the case of big organisations having wide area network and they need to see the networks like logical LANs.

The approach to **tunnel mode** is ok take the original IP packet, this is the original packet, you use it intact and your original datagram will be the payload of a new datagram. You see here this is the original datagram, containing the original IP header, the original payload, this is becoming the payload of a new datagram, having a new IP header, of course having an extra header for implementing the details of the protocol AH or ESP and it is somewhat protecting the original datagram, you see the difference between these approaches, here you incapsulate **every datagram in a new one**, is not really needed you have the same IP header in the new, think when you are at home and want to access your computer in your office, but the LAN of your office is protected so you will need some IPSEC allowing your computer to be connected to the gateway of your LAN, you will be able to directly connect to your computer, so maybe this original IP header is containing the IP address of your computer in the office, actually to be correctly routed you need to specify here the **IP address of the gateway**, allowing you to enter the network.

So you understand in this way by using this type of encapsulation you are **hiding** who is the destination of the packet, and hiding, especially in the case you use encryption, you hide the **port number**, that is here within the original payload in the original datagram, you hide such an information and you are contrasting the traffic flow analysis. You understand this is an approach that is heavier for 2 reasons, first you encapsulate every single packet in another one, this is requesting some extra processing, also you can have an issue using this approach because the original

size is increasing, is not so obvious you can afford the increasing of the size, datagrams can be fragmented but if you increase the size maybe before it no fragmentation was needed, after the increase maybe you will need to fragment the datagrams, and this means your policy should be accepting fragmented datagrams, it means you will be somewhat vulnerable to fragmentation attacks. This is just a basic flavour about the two possible mode to **activate the protocols**, transport and tunnel mode.

In transport you need an extra header, in tunnel you encapsulate the whole datagram. The transport mode is in many cases preferred for a host to host communication. To give some security and the communication is between 2 hosts you are connecting the 2 hosts for some particular needs and implementing IPSEC just for communication between **the 2 hosts**, a good solution is using the transport mode. With tunnel mode you will need an extra level of processing, normally **not handled** in a good way by the host, you need a gateway offering some extra capabilities in terms of computational power and efficiency. Basically you protect the payload, your payload will be encrypted or authenticated, of course there is some issue in the case you are using the **network address translation** because today many cases organisations are using a NAT, so that all the host belonging to the network when sending packets outside will show the same public IP, actually they have different private IP, the translation is used with a NAT, using different port numbers.

Anyway there is some mechanism for using NAT table, such mechanism is defined in some official document. Keep in mind that when you are protecting information by using authentication header AH or also ESP with authentication typically you are **hashing information** so if you are hashing you cannot afford any small change in the original information, otherwise the hash will fail and the packet will be dropped, this is why we need a special mechanism in the case of NAT. In case of tunnel mode ok we have some **extra effor**t for implementing the encapsulation of the original datagram within a new datagram, this is offering **more security** because if you are encrypting your transmission the original datagram is completely encrypted, included the header of the original IP, typically this is used for implementing VPNs, both in the case links between the 2 LANs, this is router to router connection, using such a tunnel, or typical case where the user is at home and using some VPN for connecting. It's a **host to router connection**. Here we are just showing the classical suggestion, you should use a transport mode, unless you have some strong requirement, suggesting you to use the tunnel mode, the stronger requirement is associated with a VPN. You can if you make a bad design choice so you decide to have the transport mode where you should have chosen the tunnel mode your framework will be working, the security level will be not the same, you will not hide all information, the attacker could made some model but it will be working.

Here there is a **summary,** this is the original datagram, in transport mode you take the original datagram and just add an extra header, in tunnel mode this is the datagram, becoming the payload of a new datagram, containing a new IP address and a new IP header. In many cases the destination indicated here is not the same of the one indicated here. You see here the 2 protocols, what happens in the transport and tunnel mode? We have defined transport and tunnel mode in a **general way**. Not in a special way for AH or for ESP. Next slides I would say we will see next time, we will study the impact of the choice, transport of tunnel for the 2 types of protocol, for the authentication header protocol what happens is that you are in transport mode, the extra header is providing some information allowing to make authentication on payload and some portion of the IP header, a datagram that is having some header when is being routed over the internet is getting some **changes in the header**, what is changing? For instance the time to live, changing at any hope, other information can be changing.

If you know some information is under changes over routing you cannot make any authentication about this information, in the AH with transport mode you will be authenticating the payload, not meant to be changed, and some portion of IP header that are meant to be never changing. In the case of the tunnel mode for AH you are making authentication over the whole original datagram, you are encapsulating the original datagram as the payload of a **bigger datagram,** when you add the extra header it allows for authentication the payload, the whole datagram and some portion of the new header, you are authenticating for sure the whole datagram. In the case of **ESP without authentication** you are encrypting the payload of original IP, in the case you use the tunnel mode you encrypt not only the payload but also the original header of the original IP, becoming the new payload of the new datagram, you should be able to understand the difference.

In the case of **ESP with authentication** you are encrypting of course the IP payload and you authenticate the IP payload but not the header. There is a small difference between ESP with authentication and AH that is for authentication. The last is authenticating also a small part of the header, ESP only the payload. Small. In the ESP authentication in tunnel mode you are able to authenticate all the original packet because is the header of a packet. So you see while reading such tables, tunnel mode is offering more security, you have always such more security with the performance that will be important in terms of, I mean in some cases the tunnel mode is becoming the bottle neck in a connection, when you expect the bottle neck in some cases becomes the tunnel. Any question?

New homework. Säs.

Other IPSEC features

Let's go back to IPSEC and let's see again some features of the protocol that are currently used to make IPSEC working. We have **2 main protocols**, the AH and ESP protocol and there are 2 possible ways of employing them. AH is providing support for data integrity and authentication of IP packets. This is implemented by using an HMAC function, according to the original definition of the protocol the hashing function used was somewhat **old hashing function** like you see here, MD5 or SHA1. In the updates of the protocol some hashing functions are obsoleted and new hashing function have been introduced.

These are technical details, you should be used at the idea that you will be reading some official document, I mean RFC document describing the standard, after that if you check the header of the document you will see pointers to newer documents introducing some changes, some updates and it is important you carefully read **all the updates** in order to be completely aware of the full features of some protocol, this is true for all important protocols, not only IPSEC. The idea is that AH is able to give security to what concerns data integrity and authentication of IP packets and it's preventing attack by IP spoofing just because the protocol is employing a system of sequence number, similar to TCP protocol, and there is a synchronisation of information so that it's not easy to break this synchronisation so that the attacker can send a spoofed packet and use the correct key and so on. The protocol is introducing a **new header**, now we see some details but I don't want you get the information you should learn every small details of the protocol, I want you having the **overview** of the AH protocol so you can discuss its features.

Anyway, this is the schema and you see here the classical fields, next header, this is identifying the higher level protocol, there is another information that is the payload length, then there is the **security parameter index**, a string of bits, about the security parameter index I want to add a further consideration, if you remember we defined the security parameter index like something that was describing some technical details about the keys and other techniques we are using in order to implement the cryptography to ensure security. The way such bits are interpreted is not officially defined, you can setup your schema of bits so you can say that a part of this string is an index to some table, a part of these bits are pointing to some other information that is added and managed in some explicit way in the local operating system and so on. **No official interpretation,** just like a string of bits, you can use for identifying some technical details, on the other hand this is a string of 32 bits, so it's not practical to think we have a table of 2^{32} entries and this is a pointer. This is typically in some structure.

Sequence numbers and information used for performing authentication, for instance the value of the hash function and this is generically called authentication data. The characteristic of this protocol is that it's **protecting the payload**, using the HMAC function to the whole payload but also to some part of the IP header. The non mutable fields, you know there are mutable fields in the IP header so that fields are like time to live or checksum or the field for implementing the fragmentation and so on, they can change during the routing of the datagram, so AH is considering in the authentication in the data integrity service only the part of the original IP header that are not mutable. AH is operating **on top of IP** using this protocol number, remember that when the packet is going away distributed over the internet, if there is some extra header this is understood after seeing the interpretation, if you are a generic router you receive some datagram like this, you see the header and the payload, if the protocol that is denoted by the header is 51 you pass the packet to some piece of software elaborating the payload, extracting the AH header and then can process the real payload of the original datagram. There is one level added in the processing packet.

Not much to say about HMAC, it's the traditional HMAC, again I want to point out that if you check the current documents his hashing function are no longer listed in the last documents, they are **obsoleted**. The idea is using the HMAC, so you see here the key, the outer padding, the inner padding, the traditional HMAC. When you are using AH in **transport mode** I hope you remind you have 2 modes to implement the details of such protocols, in transport mode this is the packet, if you consider it after the processing of IPSEC you will see the original IP header and the payload. If you are interpreting the payload under the AH rules you will consider the next header, the authentication header, that is organised like we have seen, then the TCP header and the data transported. This is true in transport mode, remember transport mode is just adding an **extra field**, you have the header, the payload and you add in the middle an extra field.

In the **tunnel mode** instead what happens? You have the original datagram, that is becoming the payload of a new datagram, so we have a new IP header and we have added header for making protocol AH working. The difference is that first we are protecting the **whole packet**, because the original IP header is here so even the mutable fields are protected by the protocol, and also the non mutable fields of the new IP header. This is not obvious that the IP destination here is the same of the destination here, they could be different destinations, for instance if you are making a connection to some gateway, you deliver the packet to it, this is containing the IP of the gateway, then the gateway extracts the content and the original datagram and delivering it to the internal network. Only in that moment the original datagram and IP header is made explicit. Of course this is more interesting in the case where you are also using **encryption** because in that case you are hiding information.

The other protocol **ESP** is the one providing encryption and it's supporting all modern encrypting algorithms like AES, triple DES and often is based on CBC, using the feature, encryption plus authentication, it is somewhat overlapping the service from AH, if you ask for encryption and authentication again you have the HMAC, the same usage of AH. The main difference is that when you are using the authentication feature the authentication is made only on the payload, so the non mutable fields of the header are not included, so it means that AH is somewhat **more inclusive** while doing the protection, data integrity and authentication. You get the confidentiality and you can have in some cases traffic flow confidentiality, in some cases, not every case, because if you have such confidentiality about the traffic flow, **traffic flow** is what some attacker is interested to understand, he wants to check the type of traffic exiting from your computer, the target, so a preliminary analysis will be checking the type of traffic exiting from the computer, I will send the packet for email, browsing, some authentication for some special services, I will be contacting several ports of several servers.

If you can **hide** such information the attacker is prevented to run the traffic analysis, so you can prevent the **traffic analysis** when you can imagine you are encapsulating the original datagram in tunnel mode and the payload is encrypted so the attacker cannot see any information about the original datagram, neither the port toward the datagram is directed to. In this way the attacker will be able to understand you are sending packets to some infrastructure, then what happens after the packet enters the gateway is a **secret** for the attacker, he can't make further analysis. What you see here is the logical scheme of the header that is introduced by ESP, the type of packet, again you can see here the security parameter index, you see the sequence number, then you have some data, payload data you read here, whose size can variate. To be able to manage **different sizes** there is the possibility of padding such information so that at the end of the padding you will have the pad length, software processing the packet to understand the exact size of payload, and next header, meaning what is the protocol processing the packet at higher level.

Then there is further information, **authentication data**, notice that if you see this schema what you read is that you have a header here, you have the payload and you have other fields, they are called trailer, header trailer. The trailer will contain some information and some further possible information, in the case you are asking for the authentication service, you have an extra part called authentication data. If you are authenticating information all this part will be covered by authentication service and if you independently from your request of using or not authentication you will have some confidentiality about the payload, so this is what happens in the case you are using ESP in the transport mode. This is the original IP header, this is the original IP payload so we are adding an **extra header and trailer** here, according to what we have seen previously, and there is the possibility of having a further part used when we are asking for authentication. This is ESP, for the case of both encryption and authentication, in case of no authentication this part here is missing. This is not meaning the trailer is missing, a part of the trailer will be in any case in the packet.

This is the case of transport mode, in **tunnel mode** we have here the original IP header, the original IP payload, this is tunnel mode, such original datagram becomes the payload of a new datagram according to tunnel mode rule and this is the new header, the header for implementing ESP, the trailer, this is the part maybe used if authentication is used. Again using in that case you are encrypting the payload, but the payload is the **original datagram**, so you are encrypting all information about the original datagram, you are also authenticating the original datagram, the header of the original datagram is part of the payload, of the new datagram. This is interesting, I mean using tunnel mode is requesting extra effort, the tunnel mode is **affecting the performance** of the system, if you have some good reason for requesting the tunnel mode you already get some useful services by just using ESP with encryption and authentication, to maximise protection you need to combine more security associations.

The comparison between transport and tunnel mode has been already discussed, all these features been already pointed out, I want to add that if you read the original standard documents there are some **suggestions**, is ok use the transport mode in the case you have a **host to host** security association, this is the suggestion. Host to host, whatever it means, don't think you if I say host to host what do you think? What is in the middle? **Whatever**, not just the internet, maybe host to host means environment can be very complex, this is the LAN of the enterprise here. This is a LAN in New York. You can build a VPN unifying two LANs, this is implemented by building a connection between two gateways, to give security to the other part of the path, here you have the internet. What if you need some extra security because the packet that is originated here is going through the local network, then in the tunnel, then again in the local network, here it is **unprotected**, why? You can assume the local network is strongly protected, is this sufficient to say ok I don't need extra protection? No. Because for some cases there can be confidential documents originated by here and my colleagues should not be able to see some confidential documents.

I may have some special needs, this protection here is **not sufficient**, I need an extra protection, I build a security association between these hosts providing what I want, authentication, encryption, the fact that here there is IPSEC is independent, I mean that after I build the security association here what is exiting from this host is a datagram, containing some information, authentication header and so on. When this is going to the network interface of the gateway it will process it according **not to the security association** established between these hosts, for this router here the datagram will be just a standard datagram, elaborated by using the security association going to be built here, for this part of the path, you understand why it's important to be able to examining different ways the contribution of our ingredients for the security, because the scenario can be complex.

We will see some further examples about that, so if we have a security association host to host built in transport mode this can be just a part of the game. The real game can be much more complex, depending on the settings of the features of security we need for our packets. What is important to say is that if the security association between these hosts is based on ESP packets here are encrypted, so when the datagram is going to this gateway even if the gateway is attacked the gateway is under the **control of the attacker**, that can open the content and will see an **encrypted packet,** information for decrypting such a packet the required keys for making the decryption are not known to this gateway here, the security association has been built between these host here and this host here. You can **segment** the usage of IPSEC, this is a segmentation from here to here, we will see further examples of fragmentation. If you need a VPN typically you need the tunnel mode.

These are two typical drawings, a first here is showing the typical case suggested in the official document where you have a **transport mode between 2 hosts**, this host is belonging to some internal network and then is going through the external network and then going somewhere here, in this case is suggested considering the transport level security, it's just a security between host and host. When you have security between a gateway and another player is recommended to use the tunnel mode, you see here the corporate network organised in 4 subnets and you need a **logical LAN**, you can implement this logical network by the VPNs. In this case is useful to use the tunnel mode. No rules, there are no specific rules, some recommendations and then you have you need a skilled designer, very sensitive to different needs for the different types of security you have, strongly depending on the actual setting.

Let's open small parenthesis about the **anti replay services**, this is just small details just telling something about the implementation of this measure for contrasting the replay attacks, I want to remind a replay attack is an attack where the attacker has recorded some previous packets, and he’s sending again the same packets, maybe the attacker is not understanding the content of this packet but he’s **sending them again.** The anti replay service is a service that is proposed in the original documents but further researchers are carried out in the subsequent years, today if you check the literature you will find many different proposals about the way of implementing this service. This proposal is ok is fine but it’s not optimised for performances, so you can invent something better. This is still today under research, the topic is very demanding in terms of analysis. Anyway, the standard service for anti replay is based on a **sequence number**, you have a counter so when you setup a new security association the counter is set to 0, every time you send a packet the sender is incrementing the counter, and this value is stored in the sequence number field of the packet.

Technically speaking the **first value is 1**, since we are having 32 bits you have a maximum value for this sequence number, if the connection is used for a long time maybe the limit is reached and what we need is to terminate the current security association and start a new one and starting again from 1. So that even if you have 2 packets with the same sequence number the packets are using different keys. The standard implementation of the service is based on the fact that the receiver of the packet is using a window having a fixed size, here denoted by w, and this size is also a default proposal of 64 bytes. Better to see the picture here, this is the window, fixed size of the window, this is the stream of information being sent from the sender to the receiver, for every packet you can see the sequence number and understand wether this packet is following within the current window or not.

If the packet just received is **falling within the window** and it's new, I mean you see I am using some markers here for reminding what is the list of packets I have already received, when I get a good packet I mark a slot here, just a bit, for saying ok I received this packet and I remember that, when I receive a packet I first check if the position is marked or not. If not the packet is **processed**, maybe there is the MAC service to be checked, the packet is checked wether is good or not, if the check is ok then I will be marking the corresponding slot, if the packet is falling on the left of the window the packet is old, just dropped. The packet is falling outside the window, in the right part, I will just **move the window** so if I get this packet I move the whole window of 2 positions here, while maintaining the size. You understand this system is not ensuring you will receive all the packets. There is another layer of protocol that is **fixing this problem**, that is TCP. The current research is trying to find a solution that is more performing under the point of view of performance and is able to guarantee the correct sequence of all packets.

Combination of security associations

Let’s talk now about the combination of security associations. Because every security association is implementing either AH or ESP. If you want to implement **both**, you need to combine two security associations, the result is called **security bundle.** We have 2 types of security bundles, the ones built using the transport adjacency and the one obtained using the iterated tunnelling, when we use two security associations over the same packet you have to process the packet and then the result according to the second security association, transport transport, transport tunnel, iterated tunnelling, you have **several choices**, and also it’s interesting to ask ok I want to use AH and ESP, what I use first? I want to immediately say to you that both solutions can be used, first AH or ESP. Why we want to combine security associations? Maybe we want authentication and confidentiality, if we want authentication and confidentiality you can combine encryption and authentication and you can obtain what you want using several approaches, ESP with authentication option, use two security associations, AH and ESP and use them in the mode of transport adjacency or transport bundle, let’s try to have a look to the **consequences** of such choices.

The simplest case is the one where we are using **ESP with authentication**, one only security association. It’s easy, if you are using ESP with authentication you can setup a transport or tunnel mode. In the transport mode you have this features here, this is the original IP header, this is the payload, you have the extra header and the trailer, and you have the encryption covering the original payload, the authentication covering all except the original IP header. If you use ESP in tunnel mode the original IP header is here, you will encrypt the original IP header and the authentication will cover everything except the new IP header. The **authentication field is not covered**, is the result of the hashing function. For both cases the authentication applies to the ciphertext, you authenticate the ciphertext, we can say it by just remembering authentication after encryption, encryption first. This is not only choice of course.

The other possible choice is to use the so called **transport adjacency**, here we assume we still want authentication after encryption, this is just because we have considered the previous case, standard ESP with authentication, for this standard we have authentication after encryption. Now by using 2 security associations, we decide to bundle them in transport mode, so the inner security association is ESP. ESP is providing encryption, in this case we **don’t ask for authentication,** we get authentication from the other security association, it’s useless to ask ESP with authentication. Then we protect the result of this encryption by using the AH protocol, providing authentication on the ciphertext. This means that the encryption is applied to the IP payload and the resulting packet is the IP header and the ESP packet. We could check the next slide where you see here the original packet, now we are considering the encryption obtained by introducing the ESP protocol, so you see here the header and the trailer and this is **encrypting all the payload** of these datagrams, and now we apply the second security association, providing an extra header here, and it's providing authentication over the whole packet here, you see the notation remembering that we are authenticating the whole packet except the mutable fields, they are not authenticated. The result is this, you have encrypted this payload and you have authenticated the whole packet except the mutable fields. This is built on the **transport adjacency**.

This is the result we should shrink this arrow here, because we need the header for getting the security parameter index. We have talked about encryption first, what about **authentication first?** What we want? Is not so different actually if you want to do authentication first, but authentication before encryption is interesting for two reasons, one reason can be this, authentication data if you authenticate first the authentication data are protected by encryption, you cannot attack the authentication data, you are making **harder the attack.** The other interesting point is that if you want to verify the authentication of a message like plaintext, if the authentication is made on encrypted data if you want to verify the data later you need to re-encrypt the data for checking the authentication. This may be considered not well come. In these 2 cases you can have some reason for preferring authentication first.

When you combine the 2 security association you just have to **decide the first** and the second. **Transport tunnel bundle**, in this case we can just see the picture, this is the original datagram we introduce the authentication first and then we can use this approach using the transport mode we can add the extra header here, AH, then we can encapsulate this datagram like the payload of a new datagram here and we can encrypt all the information here that is describing the original datagram and in this way you have both the services, of course you can switch and decide to have encryption first and authentication then. The difference is again what we have mentioned before here, in this slide. You can design the way you want to obtain the services, I want to ask you a simple question. In your opinion, does it make any sense to build bundles based on transport mode containing **three security associations?** Not making sense, because you have 2 protocols, you should not make the error of thinking here for packet being transmitted you are having many security associations, we are just considering the security associations defined between the 2 parties.

Now the original proposal of the IPSEC protocol is considering **4 possible scenarios** that should be supported by the protocol, they are this ones you can see here. Now next slide will consider any of them in order to better understand the meaning of this multiple queue you see here showing the concurrent use of many security associations. Remember, if you want to deliver some software compliant with IPSEC it should be able to support these 4 different configurations, at least. What we see here, we see a **host to host security association,** starting from a host belonging to some intranet, the intranet is going outside by using some gateway, to the internet, then you may enter some other intranet with another gateway reaching another host, you are not interested here in the VPN between the local intranets, you are interested here in the security associations between these 2 hosts, you want to have the possibility of defining at least AH in transport mode, ESP in transport mode, ESP followed by AH, you can **combine them** in a more complex way as you want providing support for authentication, encryption, authentication before encryption, authentication after encryption, so you need to have the **possibility of choosing,** the administrator of the IPSEC protocol should be configuring in details about this type of protection.

The standard is open, they cannot model every possible scenario of interest, they just want to provide some official port to some theoretical case. This is another case when we are considering in an explicit way the VPN between a gateway of an intranet and a gateway of another intranet. This is a case where the intranets are **connected by a VPN.** Typically you want this implemented by tunnelling and of course you can offer all the security you need. The tunnel supports AH, ESP or ESP with authentication. Typically you don't want here iterated tunnel. You need to link in a secure way different intranets. Other two cases, here you have seen the case where you want some security association between one host and the other, the 2 hosts are **belonging to different intranets**, you have the combination of multiple security associations, and inside this tunnel you will be delivering secured packets, secured by other security associations, this is the case I have drawed in the blackboard.

The last case is when you are at home and want to **access your computer at the office**, your host at home is open to the internet and opens the security gateway of the enterprise and then packets are enabled to go through the local internet. You want multiple security association, you want a VPN protecting you from your computer to the gateway, then you want to deliver packets within this tunnel, these packets can be unprotected or maybe you want to protect these packets because you need **extra confidentiality.** I remember I don't know if I'm able to find it, I gave in some exam.

Key management in IPSEC

IPSEC is including a big part for **key management**. The key management is very important because is requiring the possibility of setting up a key, the two parts should agree on a session key to do all the jobs, all is based on that session key. Two different protocols are employed, just mentioned. **Oakley** is a variant of DH, implementing a DH key exchange and preventing some typical attacks, theoretically possible against DH, in particular man in the middle, in practise they provide solutions to implement DH with authentication, this authentication is preventing the man in the middle, this is obtained considering several new ingredients, like cookies for this connection and some global parameters, of course in every case several random numbers. You can use also some fields providing **further security**, we have not studied elliptic curves, if you remind I have already made some remarks about that, you are encapsulating your approach within a polynomial offering some particular characteristics allowing you to have smaller keys and obtaining the same security you have with standard encryption methods, requiring longer keys. They use shorter keys.

This is Oakley. This is the protocol for setting up security associations, not just invented, the security association should be set up using a protocol, the name is **ISAKMP**, the security association and key management protocol, providing a framework not for setting up the keys, for that we use Oakey, once you set up the keys how do you share them? Agree, modify, delete and so on. What is missing here is very important name, basically the set of these protocols, Oakley and ISAKMP, the set of the two is known like **IKE**. This is the standard, union of ISAKMP and Oakley. Internet key exchange protocol. Just mentioning that, not considering any detail of that, the standard is too big. I want to mention that is possible to setup parameters and security associations for groups, not just for pairs, two parties, multiple parties, this is specified in one particular document that is providing the specification of all technical details, in this case we are not giving any support on the topic, this is more complex and is interesting only in some specific settings. That is for IPSEC, we have considered a general framework, some details about the type of security, we considered the possibility of combining security associations for reaching more structured levels of security, most is related to the skill of security administrator, not sufficient to just know the standard, to address the details of having a good configuration of your IPSEC.

TLS protocol

We are going to consider the contribution of TLS. First we start by saying the original protocol was **SSL**, during the time the protocol started from version 1, 2, 3, after going to SSL version 3 the new version was TLS version 1. Now we are at TLS version 1.2. Anyway today the protocol is TLS, but many people continue using the old term SSL and also myself I forget sometimes the need of saying TLS and I continue saying SSL. The best known software for TLS is having the name **open SSL**, so is increasing the confusion in my opinion. What does it mean to make security at this level?

It's very different from making security at level of IPSEC. The differences are, we are introducing a layer **between TCP and application**, what happens when you have no security level? Application is asking what type of service to TCP? TCP is providing what service? A very particular service, different from the service you are getting from IP, IP is providing the delivery of datagrams, every transmission is organised into datagrams, so whatever is your need your transmission is fragmented into datagrams and every single datagram is treated in the same way. In case of TCP, TCP is one of the protocol **asking service to the IP level** of course, but TCP is providing some further service, what is the service? As for any protocol you have the concept of sender and receiver in TCP. Who is the sender and receiver in TCP? What is the concept of address in TCP? You can say it better. **Port**. The address is the port number, you know a port is a number associated to some application. Low port number are associated to services, high number ports with clients. Standard TCP.

If I am introducing a layer between TCP and application what is the purpose of this layer? A security layer, so you can imagine whatever you are expecting, encryption, data integrity, authentication, anti replay, whatever you can imagine, it's ok. With some technology you will get all. What type of connection we are **securing** while using this level here of security? We are securing the communication from **port to port**. IPSEC is securing the communication from host to host, port to port. So I want you to completely understand the difference and you should be aware of such a difference, being able to secure the communication from port to port you are securing what? An application. Because it's an application requiring the connection from port to port, this application is of course using some host and the host are defined by their IP numbers, indeed when you generically speaking you talk about **sockets** you have the IP and the port number of one side and the other side, and these set of information are describing the communication between the 2 applications, port to port from some host to some host.

To **make secure an application** connecting Alice and Bob typically you don't need to secure all communications between them, you need to make secure the application between Alice and Bob, this means that you will be preferring in most cases TSL against IPSEC. Because IPSEC is a way to make secure **all communications** between 2 hosts, all communications means many communications, of different types. So I want you to be aware about the fact that while introducing a security level here you are securing an application, providing security features for a communication from port to port. Other details, IPSEC is **invisible for applications**, you can switch IPSEC on or off, applications are unaware about that, they just continue working, applications will be asking the service for TCP, they don't see IPSEC on or off. Now if you introduce security level here, is this invisible to applications or not? This is **not invisible**, you change the interface, applications are being interfaced to TCP, if you introduce an extra layer and says to the application you have to be interfaced to this other layer, it's not invisible. This is a **cost**, because you need to do some action on the application to inform it that should make request not to TCP directly but to **another layer** that is working over TCP. Any question? This is philosophical but very important. You are the designer, one of your decision will be do I want IPSEC or TLS or both? In some cases I will need both, not many. Using both is very heavy. Questions?

Another way of saying, of describing the security we obtain by TLS is **security end to end**, you know there is no very precise definition of what end means. End to end security means generically describing the identity that we use, of the two parts that are communicating in a secure way. **Two applications**. WhatsApp is offering end to end encryption. The server is not able to decrypt, because keys are agreed between the two parties. For people interested in that there is a master thesis examining the main differences of WhatsApp end to end encryption and the protocol used by WhatsApp. There is an **original implementation**, the security offered by another application, signal, this is very interesting. This is available for a master thesis.

Before going into technical details is important to say there is a very **nice story** about this protocol, the game started in the 90s when netscape was fighting with internet explorer for dominating the war of the browser. IE on one side, netscape on the other side. Netscape was the originator of some new ideas very smart, especially thanks to elgamal, we know already this name, that is one of the first designer of the SSL protocol. Today when you see **httpS connection** you understand this connection using the protocol http but http is being transmitted over TLS. You see https and you see the lock in the browser, you see ok it's good. Is this secure? **Who knows**. We can be really disappointed about this type of security. This depends on who server side offers support for TLS. Why?

This is important, because imagine you are asked by some organisations to take care of the setting up of a new web portal. You care many details, nice, interesting, you want to setup TLS. You want your users being secured by a good protocol. Now imagine that there is some user using some old browser and the browser is using some old versions of SSL. Imagine you know that **windows XP** has been obsoleted, no longer security support for XP, windows XP was offering a browser, IE version 6, that version of the browser was not implementing the last version of TLS. Your decision as technician taking care of the new website, do you want users with **old browser** being able to connect to your site? You are security minded, you say it's not good. Your boss will say you have to **maximise the number of possible user**, you have to offer connection to every browser.

It's not good you just are using the last version of TLS, this is one of the features of TLS, the two parties start a conversation and they say what are you doing now, what are your capabilities, they can agree to talk using some downgrade of the protocol, the server can **downgrade the protocol** until reaching the level of the client, you have your super secure server, a client arrives and says I'm SSL version 3. Conversation based on SSL version 3. **Many attacks**, where is your super secure server? There is a fight between 2 different needs, maximise the number of users, make secure your connection. Today when considering your connection to a secure web server you may be want to check the security of that server to see what type of encryptor is allowed, what level of TLS allowed. How to check that? You don't know how.

You need to understand, open SSL is a very good tool, there are many standard services to make a **quick analysis of web servers**, you will get many informations. Last time I checked the email server of sapienza, I checked the TLS connection. I won't say. Anyway, of course you see documents like this one, this is a RFC **prohibiting SSL version 2** because of the bugs in the old version of the protocol, this is somewhat asking to us and to the whole community not only ensure the strongest security with latest level of TLS but ask the users to **update browsers.** Browsers must be updated every time. This is just a small list and today we are using version 1.2 of TLS. TLS 1.3 as draft, not working yet, even if the draft is full of details and addressing new features very interesting, improving on TLS v1.2 there are several issues and there are **compatibility issues** with frameworks and libraries.

There was an experiment by google that decided to enforce TLS v1.3 for chrome. It was a disaster, users couldn't use in a correct way standard services typically used. Today you should stick this version of the protocol and prevent all users from using older versions. TLS is offering some **contrast to the usual attack**, spoofing, attack to data integrity, these slides were designed when the first part of the course was not yet defined, you will see some details obvious to you, we have already discussed.

Features offered by TLS

Let's now see a brief introduction to what TLS is offering, typically you use TLS in a normal interaction to a web server by your web browser. Actually you can use TLS in other application, not only for browser, even is the best known application. TLS is providing authentication. When connecting to a website that is TLS enabled your browser is authenticating the other part, the website, your web browser will be **sure about the identity** of the web server. This is one feature of TLS, this is a **one way authentication.** What is typically not known but by most users is that you can use TLS for implementing mutual authentication, typically the server will be offering a digital certificate to the browser for proving its identity because it's using public key cryptography, you need to certify a public key.

What you could do is doing the same at browser side, installing a digital certificate on your browser proving your identity this means that you can do **mutual authentication** during the connection between the 2 parties. This is saving time for users, just connect and both parties are immediately authenticated. They examine the digital certificates, checking they are not expired, not revoked and so on. You can do that very quickly and obtaining the goal of implementing a mutual authentication in a **very fast way**. If you are not basing the authentication on a digital certificate you can use other approaches, possible within the TLS framework. One possibility is using the **pre shared keys**. You can hard code some keys into the parties and they are just manually inserted keys, you can use them to run several algorithms for deriving from such shared information session keys, we have seen several approaches.

There are other protocols, I would say the most important protocols are **remote secure password protocols**, one of the strongest protocols for authentication based on password, this means that you can relay not only on the possibility of having digital certificates, you can also rely on strong password protocols for doing **quick authentication** between the 2 parties. Most cases what is preferred is using digital certificates, allowing to use public key cryptography to implement an exchange of keys. When making a TLS connection you have 3 phases, the peer starts a conversation to understand what algorithm they can both support. They agree in this way on some algorithms to be used, encryption algorithm or hash functions and so on. Then they do **key exchange authentication**, after that they can start to run the normal communication offering encryption based on this symmetric ciphers and messages are all authenticated.

So typically the encrypted communication is based on some **symmetric ciphering**, the result of the first part where the parties agree on the session keys and on the algorithm to be used. If you want to be a little bit more precise, at the beginning client and server started talking about what suite of ciphers they know. So the client is saying ok I know these algorithms, the server is having a list of choices and from the known algorithms the server chooses the best one matching in the ordered list. As I already mentioned you can base the authentication and after authentication you can **derive session keys,** if is not based on public keys you can base it on some shared secret, pre shared keys. Message authentication is implemented typically by using again HMAC. In order to make the key exchange in most cases we have two prominent solutions, RSA, or Diffie-Hellman, or some variation of that.

Instead of using RSA or DH you can base key exchange on **pre shared keys**, they are shared so you can derive key material for running the subsequent steps or you can base the approach on a protocol I just mentioned, secure remote password protocol. For having authentication you an use **digital signatures based on RSA**, DSA. For encryption you can use several algorithms implementing encryption, typically you want to use AES. HMAC was based on md5, again in the last recommendation for TLS md5 has been considered obsolete and so on. In order to work with public keys we need digital certificates, that are presented in this form, standard fo digital certificates, we have discussed this standard, I hope you are familiar with that. This is just the same, I don't want to repeat it.

Here of course is showing the need when a party is obtaining a digital certificate is important to **check for the validity** of a digital certificate, you should check is not expired and not revoked, using the online state protocol. this is the basic standard way to establish a session key using standard RSA encryption. You encrypt the key with the public key of the other party and so on. All these long steps are the steps we are using to establish the connection. The **handshake** of TLS protocol is the most complex part. That may fail, in case of failure the connection is not established. After the handshaking you will have all the parameters set and start the **secure conversation** between the 2 parties and use them for some time, for the time you need. According to the original design of SSL, this was maintained in the subsequent versions, the protocol was based actually on **2 different layers of protocol**, you can check this figure here.

You see here IP and TCP and now you see here the SSL material. Meaning that if you want to analyse the details of TLS you have to consider **4 protocols**, the handshaking, very important, a small protocol for changing from the handshaking to the current session, a protocol for implementing a system of alert for sending messages in case of failures and the standard secure protocol whose name is SSL record protocol, providing security offered by the SSL because after the handshaking at the beginning all the details are good, no failure raising, at the beginning you will use the handshaking, when it's ok you use a protocol for just saying that ok the handshaking is ended, let's start the real game. The real game is set by the usage of this protocol here.

You see from this figure that **http is actually using SSL record**. SSL record to be established needs to be activated by an handshaking that is a demanding interaction between the 2 parties. The **handshaking is long**, establishing of a secure connection requires many steps, many messages exchanged between the 2 parties. We will see some examples, to understand why is so long. In the last implementation of Firefox the message in the bottom part is showing some extra information about the TLS handshaking phase, to make users aware of the fact that some delays are associated with the handshaking, not with the website. And the handshake is including the steps for **validating the digital certificates**.

Now **2 important concepts**, we have the concept of SSL session, and of connection. This is a difference between session and connection. Session is a general framework, the result of a handshaking. The two parties establish several parameters and such parameters define a session, now within the session you can establish **several connections**, using the same parameters. Why that? Because the handshaking is demanding and in some cases you don't like to start a new handshaking for extra connection. For multiple TCP connections you may be using one session only and one connection based on the same session. The session is defining a logical association between client and server and how I have already mentioned is the result of the handshaking protocol. The session is defining a set of parameters of cryptographical meaning.

The session **may be shared** by multiple connections, the session is **stateful**. A connection is a communication link that is between 2 parties, peer to peer, defined within a session and is stateful. To better understand the difference between session and connection, we now have a list of data meaningful within the framework of session and connection. Let's see the parameters for session state, we have a full description of what a session is. The state is defined by several parameters, we have a session identifier, just a string of bits, you have a certificate, this is actually a description of the x509 v3 certificate. This element **can be null** in the case the security is not based on the public key infrastructure but you generate session keys by other possible approaches, like DH. What compression method you are using, before encrypting? The specification of the **ciphering approach** you are using, for instance what data encryption algorithm we are going to use, no encryption or old algorithms like DES, broken. What hash algorithms they are used for the HMAC, for data integrity. Another parameter is the so called **master secret**, a shared secret between client and server. This shared secret is established as a result of the handshaking of course. Then there is a flag indicating wether the current session can be used for handshaking a new connection, you can have multiple connections within the same session, this is described by a **flag**, you can create a session not allowing multiple connections, but generally you will approve that.

These are basically the main informations describing a session, so let's see again, identifier, certificate, compression, description of cryptography, master secret and a flag. If you want to describe what is defining the **state of a connection** even the connection is a stateful so what is the information using for implementing your connection, for each connection client and server are choosing random numbers, there is a **nonce** decided by the client and nonce originated by the server, then you have secret keys used for write operations originated by the server and write operations originated by the client. A **write operation** is sending a message, a read operations is getting a message. Using a secret key. The two parties are able to use keys, one for encrypting, the other for decrypting. If server is encrypting by some key the client will need that key for decrypting, that is a key used by the server for encryption only and by the client for decryption only, and vice versa. The server will be able to use the same key for decryption operation.

Other information needed is, since in most cases we are using CBC for encryption, you need the **seed**, also we need sequence numbers because again what we have seen in case of IPSEC we want to maintain sequence numbers for being able to contrast replay attacks, for being able to reconstruct the original sequence of packets. So you see that when you are considering a connection you have some practical information, they are keys that has been obtained by the master keys information, defined the session slave. The session state is defining some master secret and you can derive new secrets for any connection existing within that session. In particular the **most important protocol** offering the service of guaranteeing the encryption, confidentiality and authentication integrity of the messages is **record protocol**, if you remember the figure here the record protocol is the one that is seen by http and is using TCP as transport mechanism. In order to get confidentiality there are many encryption algorithms that are listed here and in this case you see the original list belonging to SSL version 1 proposal, during time you get the introduction to new encryption algorithms and see the obsolete. And also you get the integrity by using an HMAC, in order to make different from the standard there is a different padding.

These are the **basic features** of the record protocol, now I want to just show the slide because next part should be analysed next time. What you see here is providing a logical scheme, you can imagine your application level protocol should send some information from one party to the other, application data is **fragmented**, they are compressed, HMAC is added, the result is encrypted and you get some data to which you add a header. And then you can proceed we will see what is the structure of the header in the next meeting and if we will be able, any question? If we will be able to respect this original idea next time we will conclude TLS i will try to show you some examples of use of open SSL for doing encryption digital signatures of processing digital certificates and so on. The library is very big, if you like you can decide to bring your laptop and follow the same steps I do in front of you. I hope we can afford that and we have no accidents from doing that. Thank you.

General framework of TLS

Let's start reminding the general framework of TLS architecture and you should remind that the TLS protocol is a layer that is inserted over TCP and it's structured in a sense that we have what is called a **record protocol**, used for transporting information and other components of the protocol, handshaking, change specification, alert and some other details, all applicative level.

So, you can easily understand we have a layer here and several other layers vertically segmented. We started talking about this record protocol and if you remind we define the concepts of session and connection, after that we have started talking about the record protocol by saying it's supporting confidentiality and data integrity, actually these services are optional, if you see the architecture even if we have not yet started talking about handshaking you understand what handshaking is, at the beginning we start with this, so the first messages between the two parties that should communicate, can these messages be **encrypted**? I start with an encrypted message. What is the key for encryption? I need some details, to agree about some details, so according to this picture you see even the first message of the handshaking belongs to the of course to this level here but in order to be transmitted to the other party the record protocol will be used, you can easily understand the record protocol should be able to encrypt and guarantee data integrity and whatever, the very beginning it **cannot**.

The **typical services** like confidentiality and message integrity are optional of course and when actually you are using TLS connection you will have both confidentiality and message integrity. We have also seen this picture where you can understand the application level data they are fragmented into parts and every part is first compressed and a MAC is added, we have then encryption and we build an **encapsulation system** for transmitting these records. We have just an extra header here. In order to guarantee the data integrity an HMAC is employed in the record protocol, very similar to the one discussed, you can see in the definition that a concatenation is used instead of xor. About the name of these fields here we will be able to understand better after a few slides.

Again we can have encryption and it is done by using several possible algorithms for implementing that encryption, again like in the previous slides here we can read the original proposal for the original SSL, after time you can easily understand some cipher are obsoleted and new ciphers are introduced so the **encryption is optional**, if the compression is carried out the message authentication code is computed after the compression, and here you see the scheme of the information that is transmitted, you can see that we have a content here transported by the protocol, we have some extra fields at the beginning and at the end, the meaning of these fields is explained in the next slides, but it's very intuitive I guess because the only one not intuitive is major and minor version, the meaning of the first field, **content type**, of course it's defining what is the protocol asking the service to develop protocol so it's identifying what piece of software should process the data and it's using the services of the record protocol. It can be directly an application level protocol like http but it can be also some part of the handshaking or other technical parts of the TLS protocol.

Then there are 2 fields, **major and minor version**, the meaning is the one you see here, in time you know that we started by SSL version 1, version 2, then became into TLS 1.0, .1, .2. you can see here that all these versions they are having as major version number number 3, minor version number and this number starting from 0 until reaching 3, so you understand SSL version 2 is having number 2 as major version number. I remind you that SSL version 3 is broken and there are several possible **attacks against TLS 1.0**, so if you want to make your system secure you should use last version and be always ready to allow your server to downgrade the version of TLS, in the case the client is not able to use the last version of TLS, to downgrade one level, **not more than one level**, 1.1, this is not what happens in the reality, we have already discussed this point, several web servers are allowing **deep downgrading**, then we have the length of this portion of information that is transported by the protocol and we have here optional parts that are used for the case when the protocol is supporting encryption and message authentication, if no encryption, the last part is not really present, if we are supporting encryption the last part is used for handling things like padding, for performing symmetric block encryption, other information they are the values of the HMAC function for message authentication, so this part is optional. If you remind we defined the concept of **connection and the parameters** that are somewhat characterising the state of the connection and such parameters are name of algorithms, keys and other useful information for doing cryptography, so depending on the state of the connection we can see here this extra part or not.

About the **payload**, what is the payload? It's the information transported by the record protocol, we can have a normal payload like information that is having some meaning at http level or we can have other payload in the case where record protocol is providing service to the handshaking the payload is organised in small portion, type of the message, length and content. We will see some further details about these possible messages that we can have for implementing handshaking. This is another protocol, **change cipher specification**, just one byte, it's like a signal saying ok we have done with our handshaking, now let's start the game by encrypting and authenticating messages, this is the signal agreed between the two parties when the handshaking is successfully complete.

There is a protocol carrying information about possible problems, the **alert protocol**, we will say something about that later. We just defined the change cipher specification protocol, just reminding some typical words they are employed while describing the TLS protocol, the concept of pending state, current state, they could seem somewhat weird but actually the pending state is what we are constructing while defining the details of cryptography we want to use, during the handshaking more information are added to define the **pending state** and at the end of the handshaking we have a pending state completely defined, we are ready to make this pending state the current real state and in the terminology of TLS they are making 2 different types of state, **write and read state**, they mean of course sending and receiving.

For what concerns the alert protocol some bad facts may happen during the handshaking or during the normal operation of the protocol, we have 2 types of alert, **warning or fatal.** In the fatal case we are just closing the connection, in case of warning the connection continues but the higher level has to decide how to deal with the warning, typical warnings they are related to **bad certificates**, no certificate, unsupported type, revocation and other types of issues with the certificates. In the case the other party is asking for unexpected closing the connection they are making the connection broken, unexpected message, cannot decompress, parameters provided for defining the cryptographical material, not in the good format, not used.

The most interesting part of the protocol is the **handshaking protocol**. Complex and somewhat long, indeed when people normal users are connecting to a TLS enabled website they can send a real small delay, associated with the handshaking, allowing the 2 parties to make authentication of their identities, what type of encryption and what algorithms they should be using, what keys, and both parties according to the protocol may offer **several possibilities**, just imagine all the possibilities, the way to setup a key, a session key, the way to encryption and other cryptographical information, they are ordered according to some order that is an order describing the preference for the security purposes, then the 2 parties have the power to agree some way to define such practical details so that subsequents steps to such details of such information and then the real communication, protected communication will be starting.

Typically the handshaking may be seen organised into **4 different phases**, now we are going to see in more details the 4 phases, we start seeing a general picture that is a drawing showing messages between 2 parties, you see these big lines separating the 4 phases, and you see also the name of the phases and the errors showing the messages, the shaded errors are **optional**, they are not mandatory, so such messages can exist or not, depending from the state, and protection and some other details. The **first phase** is the hello phase, the second one is the phase where the 2 parties start communicating exchanging precious information like sending certificates and starting sub sessions for defining materials. In phase 2 the server is offering such information, in phase 3 is the client. The last phase is the one where both parties recognise that all steps have been carried out in a good way and they agree on the fact they can start doing encryption.

Keep in mind that while handshaking has been carrying out the 2 parties have been only handshaking, you should remind that we defined the concept of session, connection, multiple connections within one session and so on. The 2 parties should open more channels to communicate in a secure way, in every case every channel is opened in this way, starting with handshaking and no information application level is sent until the handshaking has been completed. If you remind this schema here, handshaking here, you see this type of organisation of the message being sent according to the standard, this is the message defined at the handshaking level and you should remind such message becomes into the content of a record protocol message, the **payload** of the record protocol message.

For defining the type of the message we have a **message type**, we have here the name of such message types, the message type is defined by the first byte of the payload of the record protocol, and you see here the names, names for reminding what is actually done by every single message, every request, client hello, server hello, certificate, of course these are just starting messages, hello messages and certificate is a type of message sending digital certificates, then sever key exchange message is a type of message allowing to setup key material for the write operations of the server. Remind what we said last time, we have **different key materials** for read and for write, every part is encrypting with some key and will be decrypting messages incoming with another key, there is a write key and read key, both are shared between the 2 parties. The names of the messages are very clear in some cases. Information for the setup of cryptographical material.

In the very beginning the client is starting by sending a client hello message to the server, in this message you can recognise some pieces of information used for continuing in some successfully way the rest of the conversation, version the client is showing the **highest version** of the protocol that can be with, timestamp and a nonce, this means the capability to know the time, this means capability to have pseudorandom number generator, cryptographical secure. A **session ID** that is a number describing a session, number 0 means just ok, we are starting now, a number different from 0 means ok I tried to resume some old connection or to open a connection within an existing session. **Cipher suite** is the list that contains the combination of the algorithms supported by client side, in decreasing order preference.

Every item is describing not a single algorithm, but a combination of algorithms like I know how to do AES encryption with a key length 128 and CBC so every item is a long list of details and this list of details are pre defined, belonging to a very wide universe of items. We will see some examples later, I want to show you a nice and smart way to check a TLS connection. Something similar for the compression algorithm, another sequence of acceptable algorithms for doing compression. The reply from the server is the message named **server hello** and this is containing same type of information like the client hello. Just in the case where the session ID offered by the client is number 0 then the server will generate a new session ID.

When providing the cipher suite the list of simple items and every item is defining several details about what we can do is important to know some general possible choices, it's possible to setup a session key by using RSA in the way we described it. It's possible to define a session key by using DH, there are **3 possible versions** and other possible way that are used like **smart cards** to be employed and other physical devices we are not going into details about that. A cipher specification is a crypto algorithm that can be symmetric block algorithm or even stream algorithm, a MAC algorithm, the type of hashing and the initial vector for CBC or other operation modes, and information used for generating the session keys of course such key material depends on the choice, we want to use DH or RSA.

So let's see the difference between the **3 types of DH**. I hope you clearly remind how DH works, I hope you remind the weakness of DH, the man in the middle, if the parties are not authenticated. What happens? in order to do the fixed DH the 2 parties are offering Digital certificates and in the extra fields allowed in the standard x509 there are some useful information like the public key for implementing DH, you remind the public key for the DH case is a **set of information** like the prime number, the generator for the group in multiplicative group in the case we are making the right choice for DH, this is the case. Actually in most cases it happens the server is **offering a digital certificate**, otherwise you can't setup a TLS connection but the client can't offer in many cases a digital certificate, it happens that the public parameters server side they are offered by a digital certificate always present, for what concerns the client side if the client side is having a certificate is ok, there will be the parameters, otherwise it's requested to do a client authentication or changing just the way we are doing the key exchange.

This is the **fixed DH**, keep in mind using the fixed DH is providing some useful security, some information are stated and they can be derived from the digital certificates, you cannot choose at any time different set of parameters changing the prime number. So it's true that 2 parties are still generating a random number that is the private key and this random number will be used for computing the power and the result of this computation will be the last part of the public key, but the possible results are belonging to a smaller set of possible choices, because public parameters they are fixed. In practise the fixed DH is good and really used when 2 parties are both having a digital certificate. Keep in mind in some cases the parameters offered are not compatible and if not the DH cannot happen.

The **ephemeral case**, the case where information for doing DH is exchanged by signing messages with RSA or DSS case, they sign the messages to do authentication of every single message. It's possible to authenticate both parties because I mean we have public keys offered for doing the signature of messages. According to the ephemeral case we have to consider that in theory you can consider this possibility more secure because all parameters can be changed at any time and you can establish session keys belonging to a larger set, in the case of the fixed DH actually certification is showing a public key. In this case we have the messages exchanged by the 2 parties that are **signed**, and the signature is made in some official way by using a certificate allowing the signature, this is not all certificates are meant to allow signatures, in some case you have a signature for a pair of keys and in such a case you can just sign, you cannot use them for generating a secret because in that case of course you need to use **RSA for confidentiality** but if you have only digital certificates for signature this possibility is not allowed, we will talk about this.

The other case is the **anonymous DH**, this is the weak case, not allowed by the last version of TLS, in the preliminary version of SSL anonymous DH was allowed. Is the original one that is weak against the main the middle. In the step 2 of the handshaking we have the server authentication the key exchange, the server sends information to the client offering a digital certificate maybe offering a whole chain of digital certificates and information for the key exchange in order to construct a shared key, to do that the server sends some information depending on the type of construction that is chosen in the handshaking so the type of construction is in some cases some of the DH we have just discussed or the other case is RSA, we will see we have a slide, the details of RSA the next slide.

Other possible messages sent from the server to the client is the **request to get a certificate** and the information that phase is completed. The message offering the certificate is the first message of the second phase and I mean this is what we really need because in all practical implementation of TLS today we want to be able to authenticate the server side, the client wants to see the digital certificate and of course this certificate is used for the ways we are setting up the key for the session and the only case this is not used is the anonymous DH, **never used** today because completely obsolete. In the case where DH fixed type is going to be used the digital certificate is containing the public parameters of the DH key exchange, now there are **2 possible cases**, key exchange not needed or needed.

Not needed in 2 possible cases, the case where we are going to use fixed DH we don't need to realise the subsequent steps for exchanging keys because this certificate already contains all information needed for constructing the key, or in the case where you want to implement RSA key exchange, in this case there is just a message being sent encrypting the session key. This is the other case, we need a key exchange, in the case of anonymous DH of course, this is obsolete, in the case of ephemeral DH the message is including the numbers defining the public key of DH and a signature of such numbers. In the case of **RSA key exchange** it happens that the server is using RSA but the public key for RSA is a public key for **signature only**. So what happens, server creates a temporary pair RSA public private key in order to do encryption to send an encrypted information, in order to do that it's used a temporary pair of keys and of course what happens is that the digital certificate containing the RSA public key for signature only is used in practise for sign information.

Is it clear why the server needs to create a **temporary pair?** The digital certificate is supporting RSA only for signatures. If I want to sign I need to encrypt so I need to be able to generate a temporary pair. In **phase 4** it's the one where the 2 parties have completed the handshaking and the clients sends a message to server with the information he's ready to make current state and there is a similar message coming to the server, to start the encryption. After that encryption starts. This is the **end of the handshaking**. There are several differences between the original family.

If you see here this old slide there is a detail written here, actually there are **more than 3 possible protocols** that are using the record protocol, it is worth to mention this protocol here, everybody talked about that a few years ago, the protocol is the so called **hearth beating protocol**, just implementing a simple exchange of messages, hey you, still there? And the other one replies ok I'm still here. The implementation of this protocol in the open SSL software was **bugged**. You can imagine the message, was implemented like this one, if you are really there send me this 4 letter word "blah", actually no need to say how many letters, the other part is getting the string. This bug was happened that the name of the attack was the heart bleed, the attack was implementing this way, if you are there send me this 400004 *big number* letter word "blah", in the original implementation the other party was replying by sending a high number of letters, this is a **buffer overflow** attack.

In many cases information managed in the memory they are cryptographical keys. This was allowing to **expose cryptographical keys**. The bug was introduced in 2012 and was fixed 2 years later, many organisations were employing this knowledge for attacking SSL connections. There is another very well known attack that is more complicated, we are not going into details but in order to make this attack the client requests to the server to downgrade the conversation to SSL version 3 and after that the attacker can send some number of messages for getting **partial information** of subsequent encrypted messages. This is a way to break the encryption for the TCP connection. The way to contrast this attack is to forbid this version of the protocol, so the attack is prevented. Now i want to show something to you.

Next homework. Ciaone.

Secure shell (ssh)

Our discussion about secure shell will be very short, we in particular are not going into details, how the security is enforced, we just mention some algorithms, just some **basic information**, not going into details, people that are interested in details should deepen by themselves. This is a protocol running **over TCP**, we are at a higher level and it's providing **security features**, it's very used in some basic important practical applications, it's one of the favourite protocol for network administrators, this is why we need to give some information about the protocol, this is one of the protocol defined by many official documents, so for all details you have to read 6 or 7 request for comments, so much to read, but the practical aspect of usage are very interesting, we will be providing some basic information you can start deepening if you are interested. How many of you already using secure shell? Any of you using secure shell for anything different from a terminal? Maybe some of you by using secure shell you get a **terminal** and you can type your commands, you have a command prompt so you type whatever you like, you can interact with a shell of some remote host, but you can also have different type of services from secure shell, **not just a remote terminal**, that is interesting, I agree, maybe is the best known service from secure shell but you can get other important services, ok we will just try to see something. I would recommend using secure shell because it's offering a fairly good security, even if the strongest agency of security from some countries are able to break it, so keep in mind this. You cannot protect so easily, but we will see some further detail.

Since secure shell is a protocol you should think of secure shell like a **set of smaller protocols** and each smaller protocol is using some basic features like compression algorithms, encryption algorithms, HMAC for doing integrity and some other good services like authentication services. You can think of the protocol like very useful tool able to setup a channel of communication with some other party, one point to another point, **multi point communication**, we are not seeing such details. And the channel is offering some basic security because is making available encryption, data integrity and the possibility of having one or two way of authentication. The most common usage of secure shell is that usage where a client connects to a server, the server offers authentication based on digital certificate and the client is allowing the user to **authenticate himself** by using the credentials like username and password but also more secure ways to make authentication on client side are supported. It's offering a **family of services** for doing authentication.

The practical uses of the protocol are you see now here a list, ok you can get a shell for interacting in some remote host, I don't know if you have some familiarity with **telnet**, do you? Long time ago telnet was the only protocol allowing people to interact with some remote host, today telnet is not really used just because it's **completely working in plaintext** without any primitive for ensuring data integrity or authentication, allows you to authenticate yourself but brings username and password as plaintext, today is considered **high insecure.** When I was doing my PhD, more or less 100 years ago, I was cooperating with important researchers, one of them was working in Zurich, I remember I was writing email to this guy, one email of one page, typically one day for writing it in a correct way, send email, ok. Now? Question. Is he reading my email? When he's replying?

Finger protocol, when I visited him in the university I had the opportunity to see the hostname of the workstation he was using. I could be sending some query to the workstation, finger, username, fully qualified domain address. Option -l, long output. The protocol was replying saying if the guy is logged or not logged, if is logged what are the processes are running, how many console using, how long time is idle, when he did the last email reading, and other useful information. That was like a paradise, you could make **query about whatever people.** There was also the possibility to query a workstation by asking information about all users logged in the workstation, of course for obvious reason of privacy **the protocol is no longer working,** everybody is cutting the protocol. In addition finger protocol is associated with a well known attack, based on buffer overflow attack, providing on the finger command line a very long parameter and causing the server to run into an overflow and exposing information in the system stack. Anyway, why I'm telling that? I don't know.

Today we can connect to remote host for interacting with some remote host by using secure shell. One time people were using telnet or remote login, very similar to telnet. If you want to run one single command on one remote host the initial possibility offered by UNIX system was the remote shell command, you can send one single command, this is insecure also because works in plaintext, no security features. To execute one single command without having some remote host you can use again secure shell. Making **secure your ftp**, you can use it for having a secured ftp so that you can upload and download files in a secure way, interacting with some remote host of course. You can use such features for uploading and downloading files in a more complete and integrated way so that you can implement functions like mirroring, backing up information on remote host in a secure way again.

Forwarding ports, I will show an example on what forwarding port means. Technically speaking means making a port of some host, maybe you cannot reach some port, available else where. You can touch elsewhere a special port on some special host, like touching the hidden port, this is port forwarding. You can create a VPN, completely working **VPN based on tunnelling**, this is very old usage, forwarding the x system, I don't know if you are still knowing some information about x window system. The original windowing system offered in UNIX, in the first workstation the graphical environment offered to the user was offered on the basis of a graphical server, the name was the x server. **X server** still existing, you can use it for many applications, one of the nice features of that is that you can make a remote connection of a UNIX machine and you can have the windowing system on your client. You are home and you are having the windowing system. At the beginning x window was the only possibility for having windowing system for remote users. Many, many attacks.

Also offering the possibility to have **proxy services**, not exactly the same as http proxy, I expect you know what http proxy means. This is working at some lower level, is under http, over TCP and using a protocol whose name is socks and both version 4 and 5 offering authentication, offering **many good services** and you are offering to you proxy services without having the knowledge of what you are doing at http level. Proxying a connection, you can get some more privacy. Another feature is **mounting a remote file system**, you can mount it in your local file system having a logical view of remote file system like you are having the local, using your graphical interface for interacting with the remote system. This is just an **incomplete list**. But I would say that he most prominent feature are here listed and all of them maybe expect the x forwarding all of the deserve to be deepened.

You see here the name of the guy that invented the protocol, he was the writer of the first version of secure shell, at the beginning it was offered as **freeware** to everybody, after some time the guy understood it was a good idea to found a company, **ssh communication security** was founded, still today you can browse the website of the company and you will find many useful tools because the company is not run in a deeply business oriented way, ok they are making business actually but they offer many free services to people interested in the basis services of secure shell. When passing from secure shell to version 1 to 2, version 2 became a **trademark of the company** and at the same time the open source community started a project called open secure shell, what you can imagine is an open source project, and it's compatible with the official specification of secure shell for many details.

Because the good question is ok what details should be made compatible? The fact is that in many other specification of secure shell became into internet specification so you have official RFC documents describing the protocol, so the open secure shell is compatible and implementing the specification described by the official RFC documents, there are extra features not included in the official RFC documents, they are available just for commercial purposes and you cannot get them in the open secure shell case. So you see here the **timing**, this is when the first version was made available, but actually version 1 was having too many vulnerabilities, so only version 2 started to be very secure. And the open source community started working on a fork of this project starting from the last free version of secure shell, but the open secure shell is completely compatible with the secure shell version 2.0, so today this is used and many people are starting using it open source version then they decided to want some more extra services.

Many documents, describing the standard, so if you are interested there is much to study, many documents introducing some changes into the previous documents, this is why you read an high number of documents, also for small modifications you have a new version and you need a more recent document, if I'm not wrong the most recent are somewhat old but still today secure shell is frequently employed. The **basic idea** is that you have a client and a server, clients are making secure connections to servers and there is an handshaking at the beginning of the protocol where the two parties agree on some features to be used, so you can recognise something not so different from what happens, typically when you connect your client to a secure shell server you have in your local file system a **public key** offered by the server, next time you connect you have the public key and you check that the public key offered in the handshaking is just the same as you are storing in your local file system, the first time the user is prompted, do you want to access this public key?

In many cases people just accept the first time. What if I'm **under attack?** I accept a wrong public key, so I will be connecting to some wrong secure shell server and I will be providing my credentials. So be careful about these small details, let's see the basic architecture. In this picture here you see in the lower layer here TCP, actually secure shell is not really requiring the protocol must be TCP but a typical implementation is the one **based on TCP**. And the secure shell protocol is composed by **3 basic components**, a transport protocol, and authentication protocol, connection protocol. You see also the 3 components they are living at different levels, in that is again somewhat similar to what happens in the TLS case, because in TLS you have over TCP the record protocol and handshaking. This is somewhat similar, you have the transport protocol using TCP and over transport you have authentication and connection.

You see here just an example of a possible protocol at application level that is the sftp protocol that is the ftp secured by secure shell and in this case you have some application implementing such a protocol and they will ask services to the connection protocol to enter the file transfer. In the next slide we just see some basic information about authentication, connection, transport, the basic functions. The **transport** is providing the initial connection offering server authentication, this means that at the very beginning the server is authenticated, some basic encryption and integrity services. After establishing the transport connection the client asking for services is seeing the possibility of sending chunk of plaintext in a **secure way.**

The **authentication protocol** is allowing authentication at client side, so you understand the two directions of authentication are in different moments, the protocol is offering authentication server side then you can decide how to implement authentication client side, and many possible types are supported by secure shell, not just the interactive authentication based on username and password. Still secured by primitives of encryption and data integrity but weak, **basically weak** the idea of authenticating people by just username and password. When the authentication of the clients ends in a successfully way the connection protocol is able to work and the connection protocol is offering many services, in particular it can be used to set up several connections, what you can imagine is all the job until now has been done for setting up a channel of communication, what in TLS case was called session. Now you can have **multiple interaction** based on the same channel communication like in case of TLS we had the connection based on the same session, just a logical but not really the same of course. In particular the transport layer is implementing the **initial key exchange** the server authentication, you get the way for doing encryption you also agree the party also agree on compression and how to do data integrity, the layer is showing to upper layers an API for sending and receiving plaintext packets of 32 kbytes at most, but you can have implementations allowing different sizes. The layer is organised so that after some traffic the key should be renegotiated for other security or maybe after some time. This is the **transport layer**, in this picture is this layer over TCP.

Authentication layer, this is used for making the authentication of the client, it's the client deciding how to do authentication, and several types of authentication are supported, most secure way is associating a digital certificate so that the public key authentication based on asymmetric cryptography can be implemented, implemented also a way for having just a simple static password or one time passwords and also a type of authentication I'm not going into details, based on the fact you want to have a secure trusted third party, like in kerberos, this is interesting because in some case you can have **multiple parties** sharing **one only connection** between several points, based on secure shell think of the case you have a set of virtual machines, they are running, you want to connect to all of them sending them the same command you want to send the command in a secure way, you want to send one command only to all the virtual machines. This can be implemented. If you are interested go into details, so I'm not able to do that.

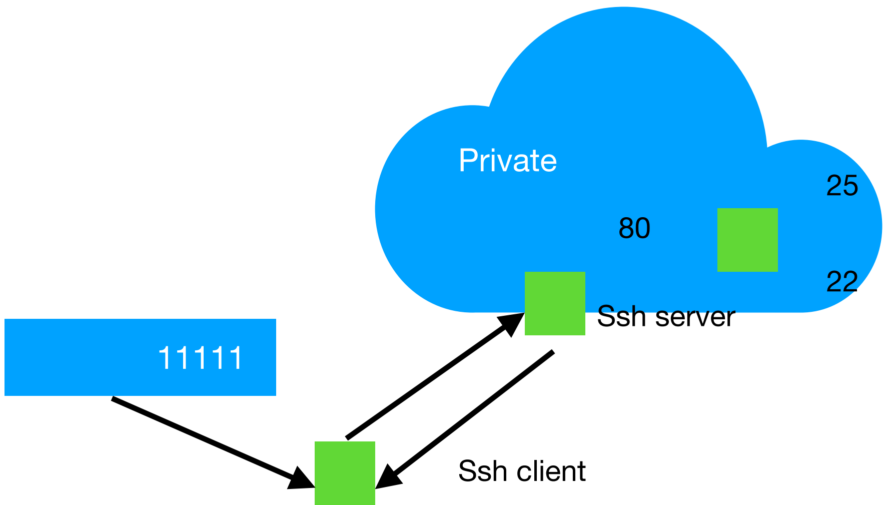
The connection layer is implementing the concept of **channel**, channel and global request. This is offering the possibility to have a communication out of band, so you can make further extra request doing a conversation just implementing out of band channel and this is what is used for implementing important services like secure copy or secure file transfer protocol, you can **negotiate the files** to be uploaded or downloaded and while doing the job you can ask for some extra operations, different channels, like ftp is working. What ports used by ftp? 21 and 20. There are several possibilities for implementing key exchange, some based on DH, it's possible to make authentication server side using **public key cryptography**, all what we have seen about the possibility to offer nonces and other information like timestamps and so on. We have already discussed in several ways. At the beginning secure shell was giving data integrity based on SHA1. Still true that AES is a strong algorithm. Few months **Wikileaks** published some documentations and showed how a security agency was able to write some tool and the tool is able to break the secure shell, this is working for UNIX based systems and also windows systems. Actually becomes like a **text interaction**, basically you connect to some remote host by identifying yourself as a user of that host, you need to be known. If you do that you get as a reply a prompt for authenticating yourself. If your client is having a digital certificate and configured so that the secure shell client is using the certificate you will be accepted immediately, because you are authenticated by your public key. You can pass some extra options been able to see extra messages about what is happening, you can enable some version, some type of IP addressing and so on. If you just type something like that you can interact.

Let me introduce the idea of **port forwarding**. In many practical cases you are the human using a machine, this is the network and you have some target. A mail out server and when you need to use a service you can make your connection source is the used machine, destination is the server, port is 23, means a protocol whose name is **telnet**. The weak one, because everything in plaintext. You may need to get your email from the mail server so you make a connection from your machine, the port you are connecting to is this number of port, a very old service for providing the identity, again is authentication service belonging to the old internet, no security at all. This is another possibility to connect to your mail server, you will be needing to make a connection on port 25, so in order to do that you need what? You need the ability at your machine to make connections with **arbitrary hosts on arbitrary ports**, this can be considered not much secure because I mean you have no control on the possible actions run by the user.

It depends on the framework, environment and organisation where the user is working. To secure the actions you may want to **disallow some** easy and vulnerable **connections**. One way to do that is using the port forwarding, if you remember the idea that a port number is the identity of an application on some target host what happens here is that if you are using port forwarding you are using secure shell on local machine and the secure shell run in the local machine is intercepting the traffic you want to send to some remote destination, some server and is making a translation of port numbers and sending the packets to another host, this host is running a secure shell server so that the secure shell server is obtaining your packets, such packets are meant to be send to another server and here happens that the secure shell enabled server will be making the connection to the remote host on your place and he will get the reply and offering the reply to you by just implementing a simple **port translation mechanism**. In order to better explain this I think is good we see an example. The first example is this picture, this is the secure communication offered by the secure shell mechanism so you can implement a secure connection to the server protected by the secure shell, remember secure shell is connecting to port and such connection can hide connections from other port numbers. In the case you want to make your connection to the server you can do such **a connection in 2 steps**, your client is connecting to the secure shell server, that will be connecting to the mail server. A similar thing happens for the other server, the **mail out server.**

You see strange port numbers are being introduced, what is the exact usage of such **port numbers?** It's better I show the exact usage of port numbers with a practical example, is somewhat easier to be explained. I was connecting to this host, do you remember? It is a host in via Ariosto, the oldest host we are using in our network, exposing a **secure version** of secure shell. Instead of just making a connection I'm providing some options here, what is the meaning? I want to make a real example. My computer in my office is a computer I normally don't turn off, just sleep mode, is having a **private IP number**, this means I cannot connect to my computer from outside, so the idea is to use port forwarding for realising a connection between my client here, the secure shell server that I can access in my department and then create another channel of communication between such server and my computer, this can be done just because the secure shell server is having 2 network adapters, one shown to the internet and the other one shown in the private network.

Using such a command I'm telling secure shell to **forward all information** that is coming from my computer, suppose it's the IP of my computer, the private one. Suppose this is the IP address of my computer, this is the port of my computer I'm interested, I want to see information coming from port 22 of the computer, I want to see such information on my local port, this is the number of my local port. I want to be sure you get the idea. I can't connect to my host, I connect to another host, this other host is able to connect to the remote host and by the secure shell server is showing to me by the connection **the packets coming** from the remote host to my local port, so that I can query my local port. Because the channel is bidirectional. I



should use high number ports. I connect to this secure shell server asking the protocol to see on some local port here, I can invent a port number, 11111, all the traffic going and coming from some port here. What port number? **Whatever I want**. Port 25, port 22, port 80. The server is making the connection to this host and when I am sending a command to my local port number the secure shell will be forwarding the command to the port number I specified in the line here on such remote host. I will get a reply that will be sent on my **local port here**. I can interact with an unreachable host, with some intermediate host where secure shell is running. I would like to have a shell on my local computer here, in this case I will ask to make a connection on port 22 on my computer so I will be using again secure shell here, so I will get by querying this port local I get a secure shell to this remote computer.

When doing such a connection you get a text interface to this host here, to have port forwarding you don't want to have the possibility to send commands to this host here. If you pass the option -M it means that ok you will be not offered a bash for interacting with this secure shell server, and if you also specify the -f option the process will go into the background, so you can reuse immediately the text window you are using for your commands. Just useful options. So let's see a command for **opening a remote shell** on my unreachable computer. If nothing happened i should have my connection here, what happens if I type exit? I exit from what? Again on my local machine in the office. The port forwarding is still up, now I want to mount my file system on my computer in my office here. Logically mount my file system, how can I do that? By creating a temporary directory. You can also have the symmetric case, less frequent. The first case is more frequent to make it reachable. They are different techniques, this is implemented by software thanks to the services of secure shell.

Firewalls

Let's start talking about firewalls. Firewalls have nothing to do with cryptography, they offer a **different type of security**. The security we can obtain is some added security, because they allow to implement policies like the simplest I can immediately describe is **blacklisting**. You don't want that people from your organisation connect to facebook, you can blacklist facebook and you can define a set of rules in your firewall that is preventing people from doing such connections. Basically the firewall is a device that is able to analyse datagrams that are going from the local network to outside and from outside to the local network. This is very important to be understood, because there are many many wrong assumptions or people that are wrong. This is the private network you want to protect, the main idea of a firewall is like a device able to analyse all the traffic, what traffic? Packets that want to go within the network and packets originated from network that want to go outside. Firewalls have the ability to **analyse every single datagram** and match the characteristics with some patterns. Implementation of policies, belong to the word of idea, you describe policies, implement them by rules that are the tools for **enforcing the policies** and you describe rules in a formal way and make such rules known to the firewall and the firewall will be applying the rules, every datagram entering the network here will be analysed by the firewall and matched against the rules and after that the firewall will decide if the datagram will be allowed to enter the network or not.

If not it will be dropped, this is a simple description, not really complete because we can have firewall more complex, but what I just describe here is a type of firewall inspecting datagrams, you can choose your firewall as a device analysing datagrams or maybe you want to **analyse TCP segments**, not single datagrams. Or maybe you want to analyse http messages, you want to **define rules at some application level,** not at network level, and you will have different types of firewall. Basically we have 3 types of firewalls working at different levels, they are allowing to obtain different types of security. Of course the higher the level the greater is the power of the firewall to protect the network from bad traffic. Keep in mind that the rules implemented by the firewall are applied to **all the incoming and outgoing packets**, in some cases you want to be allowing to internal users every action.

So outgoing packets are not filtered, just sent outside without any prevention, you want to protect the internal network from outside by applying some filters to the incoming packets, this is possible, but today it's believed we need to protect the network **also from outgoing packets**, in order to recognise schemes of traffic not good, showing for instance that some internal host being compromised, you can understand that by the scheme of the traffic intercepted by the firewall. This is a general scheme, this is the internet protected network, by needs of the firewall that is a device implement, what is called, a software typically the firewall will allow a special area, the **DMZ**, where some actions that are forbidden in the internal network here are allowed. For instance if your network should be offering a **web server** to the public you cannot put here in the protected area, this must be protected, you cannot allow any access from outside. So you will be positioning your web server here in the DMZ area, where the protection is not so strict, of course while defining the characteristics of the area is your choice, applying to the DMZ, **demilitarised zone and protected zone.**

Typically you have a gateway, that is the interface of such a network with the internet and just you have the firewall protecting the local network, in some cases the firewall is implemented at gateway level, this is what happens when you consider your ADSL connection from house to outside, you have a router containing a firewall. This is just a basic idea, you can have **several variants**, implementations and several ways to use a firewall, I want to say something. You can have 2 types of firewall, many types depending on the **level of abstraction** the firewall is working, but you can use a firewall for protecting a network so you can imagine it's a device protecting the perimeter of a network or you can use a firewall for **protecting one host**, that is different. What it means? It means that you have a single host here this is the host and the host is implementing a firewall here, **local**, and packets going to the host are analysed against the rules in order to understand wether such packets are allowed.

It makes sense, you can have **2 levels of protection**, this is a first level of protection, enforcing the rules for protecting the whole network, then you may need to have a special protection for some computer and may consider not sufficient the rules applied here, so you can use a **software firewall** running on this host protecting from inbound, outbound packets, I mean implementing rules that are the description of your policies for your personal computer and you can also have two levels of protection, the firewall for the whole network protecting the perimeter of the network and the firewall for the personal computer just protecting one single host, I hope you fully understand this difference, you can have both, in many cases you actually have both, maybe you are not aware of that but you have. This slide is just presenting what I discussed.

The firewall should be **massively protected.** I don't mean the firewall is replacing cryptography, firewall is adding extra security to what you get from cryptography, typically with cryptography you enforce information security, if you want to prevent unwanted accesses or other types of attacks like DoS you need some different types of protection and firewalls can help. On the other end firewall are useful but you cannot expect too much protection, in particular if some attack is **originated in the local network** the packets will be not analysed from the firewall, that will be useless in that cases. In many cases the firewall is completely unaware of what is happening at application level, trojan or other unwanted software, by inspecting packets I can't recognise viruses and so on, it's not really protecting you in a complete way. Just an extra ingredient, **not complete**. Actually you can have and decide wether the datagram can pass or not. That is **packet filtering.** Application level, firewalls they are acting like proxies, because they become the intermediates that will do the job for you, if the job is allowed, so you can send imagine you want to open an http connection, your computer will be sending an http string, well defined at application level, described by many datagrams maybe, and the firewall in this case in the application level firewall is analysing not every single datagram but just the http string to check if it’s a type of string you can allow or disallow.

Typically is also true that the higher the level of the firewall the stronger the ability of the firewall, if the firewall is able to work at higher level is also able to work at lower levels. Application level firewalls is also able to do **packet filtering**. They are not really much used, they are working. Most used ones are the one based on packet filtering and application level filtering, in addition you have your personal firewalls for protecting single host and personal firewalls can be working at application level or at level of packet filtering, in most cases people prefer having a personal firewall working at packet filtering level, it’s demanding in terms of cpu needs, you don’t want your computer doing all that job for protecting all the interaction with the network.

Another very important notion is the way of working, **stateless or stateful**. A firewall is working in a stateless way if every packet the firewall is analysing is independent on other packets, so every packet is analysed as single packet, not related to other packets. In this case the firewall is stateless, in the case the firewall maintains some memory about other packets the analysis is stateful because the firewall can have **full or partial memory** of other packets able to determine some relationships between packets, but making a firewall completely stateful is a very huge job, because of the huge amount of traffic typically managed by a firewall protecting a normal real private network. So in the case you have some stateful firewall it is **partially stateful**, it’s maintaining partial information about the state of the connection. Here you have 3 pictures describing the way the 3 types of firewall are working, this is the firewall based on packet filtering, in many cases such a capability is offered by the router, containing a firewall operating in packet filtering, the analysis of every single datagram to decide wether the datagram can pass or not, no other actions.

In the case of **application level firewall** they are working like proxies because they are offering special software for different types of protocols, you see this drawing here, the firewall is knowing such protocols so is able to receive a request from inside, if ok it will be the firewall asking the request to the remote host, acting like a proxy. The number of protocols the firewall is knowing is a finite number. I mean the firewall is **knowing just some protocols**, what if some day you need to use another protocol not known to the firewall? It’s an issue, because the administrator should decide wether some new protocol should be allowed or not. What is the default, negative or positive? Just a decision, depends on the model of security you want to adapt. The higher the security the more restrictive the model you want to adopt. It’s assumed to be very secure an approach based on **white listing**, rather than blacklisting.

Black list is just a list of forbidden connections, all what is not explicitly forbidden is allowed. White list is the contrary, everything is blocked, except what is belonging to a small set of actions. Maybe the white list is containing just the list of the few websites you are allowed to visit. Of course white listing is much more strict. Typically users hate white list approaches. White list approach is also called **positive approach.** Blacklist is called negative, it's listing what is forbidden. You see here also a logical scheme for the intermediate type of firewall, circuit level gateway, actually the idea is that the localhost is making some TCP connection with some port of the firewall, if such type of TCP connection is allowed, then the firewall will be making another TCP connection with the target and will be offering a shortcut linking the two connections, if ok, so the rules are **described at TCP level**.

Let's focus on **packet filtering**, because it's true that packet filtering is also the most used approach you can find whenever you need to be completely aware of what packet filtering is, remember you can have the stateful and stateless approach, the easiest case is packet filtering completely stateless, it means that every IP packet datagram is analysed on the base of the datagram itself, stop. No other information, you inspect one datagram and decide wether the datagram should be allowed or not. Keep in mind while inspecting a datagram you can decide to analyse the header of the datagram, containing some information defined at IP level like IP number of source and destination, other information associated with time to live and other useful things, data fragmentation and so on. You can decide to to **deep inspection** by opening the payload of the IP packet and the payload of the IP packet may be containing a TCP segment, you can read information like port numbers in order to make some further decision about allow or not allow the passage of the packet.

Typically packet filtering is based on **IP numbers and TCP port numbers.** This is very common. Typically the approach for defining the rules are implemented in a very simple way, you can imagine a firewall working in this way, I now make a very simple example. I want to point out this is not far from reality, imagine that your firewall is having a list of rules, the first rule may be I don't know IP number, IP number like 201.203.*, this is a **typical rule**, allow packets coming from subnet, you can define many rules, when a datagram, we are talking about packet filtering, so the analysis is made on **single datagrams**, when a datagram arrives it's matched against the first rule, if there is match here, the if the pattern is matching what is described here the decision is applied, if no match check the next rule. If match, apply the decision, if no match go the next rules, the rules are analysed one after the other.

The **order of the rules** is meaningful, suppose you want to allow packets coming from this subnet, 120.10.15.*, then you want to block every other packet. You can write like this, ***.*.*.* BLOCK**. You can do that, because the packet is arriving, I first check this rule, if belonging to the pattern I allow the packet, otherwise I block whatever it is. If you exchange the position of the rules every type of traffic is blocked. So the order of the rules is very important, when defining a rule you can also specify if the packet is incoming or outgoing or both cases and you can also define what is the **network adapter** where you want to apply that rules, maybe you have many network adapters, you want to configure only some network adapters, in a very protected system you have very strict firewalls with very strict rules and you have to define good rules in order to allow your host to communicate with local host, some application may require that different processes need to communicate on TCP/IP, so they need make a connection on some other port.

A connection to localhost is analysed against the rules, if you have something like that every connection to localhost will be blocked, so you need to define rules thinking of the several network adapters you have in your device. And also **logical adapter**s. You may think ok but what if I write this rule in a different way? I make a list of the good IP numbers. Very demanding, because if I want, I have to write a lot of things, it's easier to just block everything after this check. They are not defining a partitioning of all possibles states of packets, not a partition, so you have some overlapping, this set of IP numbers overlap with this set of IP numbers, you allow that because it's simpler for the network administrator to write rules but I believe me, I tell you that after many rules it's **very hard to understand** the scheme of the patterns you are going to define. Very hard. Typically the decision is accept or reject, when rejecting a packet you can use **2 different possibilities,** just dropping the packet, the packet is disappearing and the sender will not know about the dropping, the other possibility is that the packet is blocked but ICMP messages are sent back to the sender to inform the delivery of the packet failed. This is a software way but in many cases we just block the packet because we don't want extra traffic. You see tables like this one, there is an action block or allow, a column associated to our host, I mean the host belonging to the protected network, star means whatever host, in this other possible case we have our gateway.

Just a different way of defining your type of matching again it's true that when a packet is arriving is analysed by matching the packet against many order of the rules, that is critical, is this approach of **packet filtering robust?** Ok it's good because it's allowing the implementation of some basic policies, but it's not preventing attacks based on some specific application, again you can think of a virus, not containing any mechanism for authentication, it means that you can receive a **spoofed datagram**, if the datagram is spoofed the IP number of the sender is fake, you cannot take any good decision in this case. Another real problem is the fact that since 10-15 years people are offered to use firewalls that are already configured, the firewall is having a default configuration, it looks ok, typically is not ok, because the default configuration is **meaningless** in your case, your case is always a special case, you should be able to configure the firewall, in many cases there are many vulnerabilities related to this configuration of firewall, more frequent than expected. Also in the case of skilled network administrator you can meet these firewalls allowing some packet passing that should be prevented and this can be happening for a long time before it's discovered a vulnerability.

Fragmentation attack, you should remember that, is based on the idea of splitting a packet into many smaller packets, you can do that for IP datagrams and you have the reconstruct failing just because information about the offset of the single fragments of the datagram are wrong and the software is just failing and this becomes into a crash of operating system of the target. Also in the case you want to do a **syn flag attack**, I talked about that when we first met. Is consisting on the fact that a server is receiving many request to open TCP connections, the connections remain half open, meaning that the TCP protocol is based on a **3 way handshaking,** the client opens a connection, the server replies ack, now it's expected the third message from the client, never coming. Many clients make the half open a connection, the server will have to use resources for storing information about the half open connections, after some time such informations are released because there's a timeout, if a new request arrive more frequently it happens that some time the failure at server side to accept new request for opening TCP connection.

This is seen all these type of attacks are **not prevented by firewalls**, there is a long list of possible attacks that are possible due to the limit approach of packet filtering, so we don't go into many details because I think it's also easily understandable. Another limit of stateless filtering is the following, suppose that you have a mail server, the firewall for your local mail server is containing some rules that they should be allowing the mail server to receive information from outside and making the server responding in a correct way, in order to do the job should be able to receive and send packets, you agree, ok, since this is a mail server is receiving packets of port number 25, the client is using a port number what number? Some high number, whatever it is. When the server is replying to the client it is sending a packet from port 25, to high numbered port, now if you just observe this behaviour this single packet going from the mail server from port 25 to some remote client whatever it is, on a high port number **should be allowed or not?**

We don't know, to understand this you should say ok this is a reply to some already established connection, it's ok such a packet is sent from outside, in other cases you don't want that, there is no reason why a mail server should open a connection with some external client, do you get the idea? The same packet if you see the packet into some context you consider the packet ok, in other context should be considered not ok, so you should be able to analyse the packet **within some context**, if the analysis is stateless you have no information. No context, in any case. To make a deeper analysis you need to introduce **some small context,** indeed it's used what is called a **session filtering,** session filtering is an approach where you analyse a packet within the context of a TCP session, you can say ok this is a packet replying to a request of opening a TCP connection, this is a packet related to another connection, for instance we mentioned ftp, ftp the client is making the connection on port number 21, the client is making the request, get file blablabla, ok, the connection for sending the bytes is made on port 20, so should be the firewall accepting a packet exiting from port 20? Only if such connection is related to a connection on port 21, if no connection the connection on port 20 is not making any sense and you should block the connection.

The context in many cases is providing a very simple context like the TCP context, providing user information **enabling a deeper analysis** making the firewall able to distinguish between 2 identical packets belonging to different contexts. In a session filtering framework, session filtering is introducing some memory, so is a stateful analysis, that stateful means that we are memorising any information about the previous state? Answer is not, we just are **managing some small information** about the past because of the huge traffic you can expect. This is allowing to the analysis, of course if the packet is belonging to a UDP based connection, actually UDP it's connectionless, you **don't get further information,** this is good in the case of TCP connections, of course in IP tunnelling such type of filters are useless, you cannot open the payload in many cases and not understand what is the context the packet is working within.

So this is a typical data structure maintained by a firewall to remind information about connections, information like that, a connection between this IP number source address, port number, destination port has been established, so you see here a list of established connections, by using a data structure like this table you can make some deeper analysis while inspecting every single packet, packet filtering in the case of session filtering they are allowing a more careful analysis, that is somewhat stateful, not so stateful like in the case of application level firewall of course, just stateful because you remind some information about the TCP connection. Next we will be talking about a software, based firewall, we start with that next week, the software is **iptables**, a very well known software in the open source community, supported in all unix linux distributions, if someone of you is having some familiarity with linux I suggest just try to have a look to iptables, a powerful tool for doing a stateful analysis of packets implementing the session filtering, actually iptables is a software allowing to do more things like managing nat tables, and other things like that, but we in our course we will be just focusing a base usage of iptables just for implementing packet filtering in a session stateful context. Goodbye.

Back to firewalls

Let's go back. We were talking about firewalls, we saw **several classifications**. First case, we have a **personal** firewall, in the second case it defends a **whole network**, another classification is **stateful or stateless**. A first introduction of a stateful is a session filtering, you have a small memory, you maintain information about the state of TCP connection, it's partially stateful. Another classification is an approach based on **packet filtering**, circuit level filtering, and proxy application level filtering, the firewall is acting like a proxy. You can combine these different types of classification so that you can derive many different types of firewall, of course it makes no sense having an application level firewall working for one host only. Typically it will be protecting a network, packet filtering may be protecting a network or personal host, not all possible cases are useful. In theory you can have all possible cases.

We introduced the **Iptables**, implementing the services of a firewall, partially stateful, it allows to consider the concept of a session but the type of filtering is still packet level, not working at **application level**. Iptables is a software, not just a firewall, it has software to implement NAT tables, other possible systems for more **advanced routing,** very powerful and not the only one for firewall in Linux. Just for what is concerning the packet filtering. Iptables, is working on some tables, as you can imagine. A very important concept defined within Iptables is chain, what is a chain? A **chain** is a list, I hope you remember what we told last time, when we described the approach to packet filtering and every rule is defined by a **pattern**, something used to design wether the packet is matching a pattern, and a decision, **accept or reject**. This is the basic approach. We also mentioned the list of rules, processed in a sequential way, the first rule is checked, if not matching the second one is checked, and so on. What is important is that the pattern associated with the rules are not defining a partition of the set of possible packets. Not a partition, if you analyse every single rule and you define the set of possible packets matching the rule you will see the sets may overlap. Since there is this important principle of analysing the sequence of rules in a sequential way, the fact that the rules are overlapping is **no longer meaningful.**

A list of rules is checked, in the Iptables world the decision associated with a rule is called target. A target is just what to do with the packet **matching the rule**. Iptables works with several types of tables, what we are considering now is the first, the **filter table**, used to define the way the firewall is acting. There are other tables, you can see one of them is associated to the decision of a NAT system, there are other possible tables, they are **big details**. We just focus the application of Iptables with the firewall. Iptables is allowing user defined chains. There are some building chains. Every chains is a **list of rules**. You see here, actually the ones interesting for us are input, forward, output. The chain whose name is input is a chain so a sequence of rules that should be considered for deciding what to do with packet that are **incoming.** Iptables is running here, this can be protecting the perimeter of the network, or maybe just protecting the host. The second case we call it personal firewall, the first case is the general one. The **input chain** is a set of rules considered when a packet coming from outside is trying to reach the host. It's a packet meant to be sent to the host, what host? Where Iptables is running. You should understand the input chain is not meaningful when you are considering the Iptables like a tool for protecting the network. Input chain is just protecting against packets coming from outside, trying to enter the host.

In this case, **outside** what does it mean? Means not in the host, external. We will be more precise about that, we can define while using Iptables the network adapters where we want to use. We can be very precise in defining rules. In a symmetric way, output is the name of the chain that is used for matching outgoing packets, originated by the host where Iptables is running, they are trying to go outside. When you are using Iptables **as a personal firewall** the very important chains are input and output. This is the point that was not clear in the last exams. Of course when you are using Iptables for protecting a network the important is **forward,** in the forward there is the idea that is the packet coming from some point, external or internal, and they are trying to go in a opposite side, in this case typically the host if it's protecting a network, the host is having **2 network adapters,** one linking the host to outside, maybe to the gateway, and an **internal adapter**, seen by the local hosts, belonging to the local network, and maybe they are having 2 different IP numbers. In most cases inside this local network all IP numbers will be private, on the type class A, B or C, depends on the size of the network.

In this case you can have more private cases. Is that clear? Anyway, now I don't want you getting a wrong information, in the general case Iptables is interesting at the same time both in the input output case and forward. In the general case you use Iptables for **protecting** your network, the forward chain is very important, containing all the rules to be checked, but maybe you want to make some **maintenance** on the host, so you want to setup a new rule on the firewall and maybe ask the firewall give me the list of the currently employed rules. You want to directly interact with the firewall, even if it's protecting the network some administrator is interested in **connecting to the firewall**, maybe with secure shell. In this case you want to connect directly to the firewall, what is important is the input and output chain. In order to do administrator operation input and output chain are important, if you use a personal firewall of course input and output chains are important but the forward chain is no longer very important.

We are not having a real symmetry between the 2 possible cases. You have other chains to be used when you want to use your host in order to contribute to the **routing** of a network. In theory every host implementing TCP/IP can give some contribution to the routing. A chain is a list of rules, every rules is specifying some criteria to be checked wether the packet is matching. Associated to every criterium there is a target, a decision to be taken when the packet **is matching.** We don't have only accept or reject, we have more targets. Accept, we understand. Drop, means reject without leaving a feedback to the sender, no ICMP message sent back. There are also other rules not explicit here. I want to say that in many cases when a packet is going from originator to destination the packet is going across many firewalls, in many cases **just blocking ICMP**. The importance of ICMP is going reduced. ICMP messages are important to test your network, firewalls are just cutting away these types of packets. There are other possible targets. I want to skip this target but I say there are two important targets, return and the name of **user defined chain.**

These tools are allowing the implementation of a very basic idea, the idea of having a **subroutine**, a function with dealing with special cases. You can do that. Imagine this is the input chain, having several patterns here, to understand if a packet is matching or not. Maybe you want to do some deeper inspection, you can decide to write here as a target not drop, accept, just the **name of your subroutine**. Implemented like a sub chain, you create a new chain, user defined chain, working in the same way, several rules. For every rule there is a pattern and again you can have the same targets, so you can call other subroutines. In this case for packets matching this rule here the target the processing continues by checking these rules here. You understand the way it works, now you may have a target here like return. Now it makes sense. With return you continue here, just typical. Of course it's not good to use return, input is a building chain, no other chain calling services. **Return is a target** to be used in user defined chains.

We get the concept, but why I want to define a user defined chain? Imagine that now you start using a web server, you make a set of rules, extra rules to protect your web server, maybe you can define a rule saying if the IP of destination is the web server go to the chain, you may write here many possible cases, if none of them is matching you can go back and continue the analysis, maybe this subroutine was **not meaningful** in this moment. When this chain is concluded, you analyse all possible rules and no rule is matching, when the chain is completed there is a **default rule**. You may decide wether after checking the whole chain you want to drop the packet or accept it, your decision. Normally it's recommended you are dropping the packet, this is associated with the **white listing approach**, more secure but also more restrictive for users. The queue target is something able to pass the information to some local application, designed for these special cases, running outside the Iptables process.

The **building chains** are within the tables. The filter table is the one used for doing the firewall operations, the filter table contains building chains, input, output, forward. When you start running the Iptables maybe you have no rules at all, depends on your input distribution. There are other approaches, more user friendly, I remember I started installing Iptables, empty chains, every packet was accepted, so there is some **basic philosophy** in the way to use firewalls. Maybe you can tell me the default behaviour. We have other building chains, we are not talking again about tables. It's worth to mention the fact there is a diagram like this one, showing how packets are managed, it arrives here and the packet is **processed** by Iptables according to this logic you see here, at the beginning other tables are used, NAT. After that the filter table is used and the question is, is such data meant to be sent to the firewall, it means you send to the host where Iptables is running? Answer no, yes. In the case of **no** you just use the forward chain to analyse the decision of the firewall about the passage. If the packets are meant to be sent then input chain. Of course there are other possible tables again used for other services.

You have building chains, empty by default. You have to **add your rules.** Being able to add very quickly a rule, stop network, in case of some emergency, we will see some examples about that. The project Iptables is very big, there are several external modules provided by the community and submitted to the management. What happens is that you can extend the power of Iptables by using extra modules, you can decide use for adding extra capability to the firewall but the point is this, what extra capability you may want? You may want to decide in order about other possible issues, maybe you want to analyse the **rate of packets**, protecting about the DoS attack, maybe you want to accept at most 100 packets for second, in order to be able to define a pattern you need an extra module. There are many **extra modules** to power the more complicated patterns, after that you can imagine the administrator is changing rules from chains by acting on the command line. Working on a terminal, very important point to mention. When you are using Iptables there is a file containing the **configuration**, when it's started it reads a configuration and starts working according to it. To change the configuration you have to restart the service, then you will be ready again.

Since the configuration is not an easy task, it can be **hard.** There are several tools offering GUIs, not only command line. You can click, use menu and so on, very nice. The limits are, first you have to choose among a limited set of possibilities, you cannot use **any configuration**, just a set of decision offered by the GUI. second, when you are configuring the firewall using such a tool the configuration file produced is having a special format. One layer format, very special. Next time you use the same tool it will start reading the configuration file. The tool offering the GUI in most cases will be no longer able to open file. This is a **very bad limit**. You are not expecting that. This is why real administrators use Iptables only with command line, it's the only way to use the full power of the software, no limits about the contents of files, otherwise you will be limited in possibilities to configure Iptables. When you are using Iptables on the command line you are typing text and can ask to load some external module by using an explicit command line option, you define an option asking to use some external module, or in some cases you can have **some shortcut.**

In some cases you can load an external module by just naming a protocol by command line, in other cases you choose the official way. In the case of TCP after loading the module you can add extra options and then using this option here, and the name of a possible state to be chosen within this set. These 4 names are important, describing the **state of a connection**, useful when you want to use Iptables for session filtering, you want some packets or to recognise some packets to open a connection, useful to analyse different possible cases and this is not the only option when you use this module. I will be giving a few examples of command line usage of Iptables. Before that I want to just have a **very quick list** of some modules. For being more restrictive on the number of parallel connection at TCP level, ok.

For locally delivering you can check the **MAC address**, when you receive a datagram the datagram is coming from some local host, maybe coming from the gateway, originated outside, then the gateway is delivering the packet to you, to the firewall. So in that case you will see as a MAC address what? The gateway, not really interesting. You may want to kill packets because of too much traffic. You want to kill packets associated to a particular owner. This is just a selection over a very large set of external modules. Every time you need to implement some **complex rule**, before being crazy check wether this module is already done for you, you can inspect. Let me just jump to this slide here, this is a very typical way of writing a line of Iptables. You see the capitals letter here, Iptables -d table. You are working on the firewall, it happens that filter is default so if you just omit table you are defining something for **filter table**. You can specify what chain, you can specify the network interface associated with the rule and it can be an input datagram, an output datagram over such a network interface. You can specify -s for having the source IP address and -d for destination IP address. With -p you have details about **the protocol.**

Let's see some more particular examples. Assume we have a **network interface**, Iptables. No need to say we are working on filter table, -a that is append. The chain forward for the network interface eth0 for incoming packets, drop. The target. Maybe you want to limit, this is a very drastic approach, this is another very interesting tool, we want to accept all packets **coming from outside,** useful for TCP connections starting within the network. Started from the internal, of course if you start locally a TCP connection you want to allow those packets, and only those packets. Again you are protecting the network, forwarding is when you protect a network. When loaded the external module you can add **extra options** meaningful for that module. If you are receiving a packet coming to your network interface, if this packet is belonging to a already established TCP connection you accept it. This is the meaning of the rule. You can check also for TCP flag.

This example here, we have an explanation and a rule, let's look at the rule. We are wanting to append, remember -a is append, it means you are adding an extra rule, in the end you may want to add an extra rule not in the end, normally you are an administrator doing a good job you **don't want** to add rules in the beginning, but maybe you want to change some rules because you made bad choices. You have to change this flag, you can have a different options for saying the exact position you want to add the rule, you can also delete a rule by just writing -d. Anyway of course as you can imagine you have a very **big list of possible options**. In some linux distributions there is built in another service checking the configuration files of service you are interested in.

When you send packets of the type of request, you get the reply so you are able to specify the protocol, ICMP, and then after loading the module for ICMP you can specify extra options like type. You maybe want to accept such. This is interesting because it's showing how to accept no more than one ping per second. So you have a rule, this is the protocol ICMP, you load the module to define the limit, you introduce the limit here, one per second on the network interface eth0. You accept packets matching this pattern, if you get more bits the pattern is **no longer respected**. This is similar to TCP segments asking to open a TCP connection, they are SYN flag and so you are saying here let me define a pattern based on TCP. You can combine several types.

Notice please that again you see this criteria of selecting a high number of port, Low numbered port are associated with service, the **well known ports,** high number ports are associated with clients, even this is good practise, not a real rule, nobody is preventing you to open a service on high number rules, many firewalls will block attempts to open a TCP connection on high number ports. You should be able to consider very carefully the port numbers so it's good that a high numbered port tries to open a TCP connection. Not good the vice versa. A very long list, I'm bored. I want to show you this one, this is good, because in many cases your host it's different components talking about standard TCP connections, even if they are running **on the same host.** In this case you need to authorise input output packets for the interface. Normally you can't open a connection on some port, like 20, unless it's **related** to another connection. You have to load the module state.

Typically they are validating the connection in case of port numbers, what protocol is used, also username and password, so you can implement some logic at **higher level**, not really working at application level. If the request of connection is satisfying the rules the firewall is opening a TCP connection implementing a shortcut linking the two connections, actually the inside host will allow to connect to the remote host. If you want to reason at higher level will be the most natural choice is reasoning, the firewall working at this level, application level gateway, also called proxy, happens that the firewall is knowing **these protocols,** the firewall is having some rules defined for every protocol, this means that when the internal host wants to open a connection outside, the opening will be intercepted by the proxy and analysed according to the rules defined at http level, and then the proxy is acting as a delegate, if the protection is ok the proxy will ask to all information outside and acting like you expect from proxy. The main limit is that if you are using an application level gateway the gateway will be knowing just a few protocols, preventing you from using a **new protocol** not implemented. This is the main limit, very restrictive.

We are not going into details of application level gateways, a very wide area and there are several types of approaches, requesting time. Of course if you are interested in performance packet filtering are more performing than application level gateway. It's important to say in some cases you need to **modify the behaviour.** How should I design my network to use the firewall? Where to place it? The internet, protecting network, you have a router, that is your gateway allowing external packets to reach your network. In most cases the gateway operating like a router is implementing **packet filtering.** This is not meant to be the real firewall, you can add some extra protection with your stateful firewall, so you have to decide how to manage the protection on your local network and a very typical approach is the one based on the so called **bastion host**, very special. It is strongly protected, how?

You will be strongly protected by implementing physical measures for allowing access to the host only to authorised people. Also you can implement the so called hardening of the operating system. It means **remove** from the OS all the software you don't need. Imagine you have a host in ubuntu server distribution, you know that in ubuntu server distribution you have some standard software, compiled or some interpreter for python and other stuff. Having a compiler working in some host in principle if a vulnerability because if the attacker uploads a text file can run the compilation and getting an executable file, so just remove the compiler if you don't need that software in the host. Remove other **unwanted software**, all you don't really need. Is good to adopt a whitelist approach, you start saying I will remove every software. In principle every system interfaced to the internet should be, is an easy target for multiple attacks. Here you have a list of **possible measures** for implementing.

In addition you have some extra features, if you have decided to use a bastion host it will be the only directly linked from the internet. It's also good that it's working as a proxy server so can implement firewall at the application level, in order is good that is done according to the **change routing approach** for UNIX systems. You can run a command, whatever command, and you can tell apache that the route of the system is another. Even if apache is compromised so that the attacker can control the way to use the software it will be seen only the specified file system. Not so easy, because it's not sufficient to write change root and run the command. Is expecting that the file system when creating a local root as the root of the file system you need to create local directories. You will have to insert **standard files**.

Other measures for protecting the network is checking the processes running, check file system, especially the part containing operating system and so on. If you are using a bastion host this is a picture showing how to organise your network. It's good to use not this solution, with the bastion host showing **one network adapter,** it's better to have a dual bastion host, in this case we have two network adapters, you implement the physical connection of the two networks, there is no possibility to reach the internal network, only by allowing them by using the bastion host. Of course the compromising of bastion host is dangerous, this is why we are defining it as a special host to implement **special measures.** This is a very good type of organisation, with several network adapters, implementing this connection from external to internal network. Every packet will be forwarded by the bastion host. Another possible way to organise is using 2 routers, external and internal one. This is the only part of the internal network that can be seen from outside. I see bastion host, some server.

Sharing secret

We have a small time, this is like a game, you will have some time. It's a very advanced topic, even if the paper from Shamir is a paper of the 70s is still today an important topic doing some nice addition to the approach. This is why this is a good topic for a master thesis if you are interested. Let me explain. We are talking about **sharing a secret.** Whatever you want to consider a secret, a number or a string, ok? For our purposes we can consider a number, maybe a long number. What is your goal? Sharing the secret with n subjects, so I mean by sharing, I want to give to any of the subject some piece of information, and every subject by just studying the piece of information is **not able** to reconstruct the secret. You need every subject, giving a portion of information you need some algorithm, collecting inputs and reconstructing the secret. It's a very general problem, means that in the moment you have a secret, the algorithm is deriving some piece of information, delivered to some subject and then delivered to the memory.

Now subjects are having piece of data, they can do **nothing**. This is what I want by design, every subject can do nothing with a piece of data. Also in the easy case, we will see two cases, we want that in the easy case if I am providing raw data to 4 subjects here I want that they are not able to understand the secret, not even if this is cooperating with the other subjects. They cannot reconstruct the secret. We need all of them. If all of them is providing their contribution just in that case you can reconstruct the secret. This is what **we want by design**. Just not a feature we find in some algorithms. This approach is called to be information **theoretically secure**. Even if the adversary is having infinite computational power cannot use that to reconstruct the secret, he is missing information. It's like ok I think a number, I give you infinite computational power, find the number. You cannot. There is a very easy solution to this problem, very **nice and smart.**

The easy solution is this, the algorithm is based on the xor operation. I have a secret s, I consider this secret like a number, the algorithm is generating random data to be delivered to the subjects so I have my secret, now I generate a random number with subject here, I generate another random number delivering to the other subject and so on. All subjects are obtaining a random number, **except one**. The last subject is obtaining the result of this operation: all random numbers xored with them and xored with the secret. So he's getting another number, looking random again, because it's dependent on many random numbers but now if you compute the xor between all these items here you get **the original secret.** This is based on the property of xor operation. Easy and smart. You understand if you take all subjects except one they cannot reconstruct the secret. The fact that this subject is obtaining a special fragment is not making special this subject here. That contribution is the same of other raw data here. Maybe you will just generate strings whose length is the same.

Good morning. About Shamir **secret sharing**, I want first to remind you the model where we have pair two number (k, n) where k is not greater than n, we want to generate n fragments of information so that given some secret, you can imagine it as a number, then you can **reconstruct** the secret. You are given the secret, generate n fragments and whatever you collect you can reconstruct the secret. The idea for implementing such scheme is using **interpolation**, and I am mentioning the procedure we have already seen last time, but in order to be clear I prefer so we are given the **input**, is the secret, we have some information describing the framework we want to generate n fragments, and we want threshold is defined. We want to generate the fragments, basically generate a **random polynomial**, all the mathematics is mod p, where p is a prime number, the idea is to generate a random polynomial, it will be a polynomial over the GF mod p, the coefficient of the polynomial they are integer numbers, they are belonging to this interval, from 0 to p - 1. You see here the random polynomial, most is generated at random, but we have to focus on 2 significant steps, one is the last coefficient is just secret.

The degree of the polynomial is described by this number k, I mean the degree is k-1, meaning that this polynomial can be described by **k points in the plane.** So the fragments to be delivered to the so called shareholders, and the shares are just the ordinates of such polynomial computed until reaching number n. After the generation we can just discard the polynomial and the secret, just we remain with such **distributed information**, n points, (i, si). Number I and corresponding ordinate. The idea is generating a random polynomial controlled by 2 parameters, the degree, I mean there are 2 parameters not random, the degree of the polynomial and the last coefficient meaning the ordinate where the polynomial is intersecting a point over the access. So in this case we have just generated **7 fragments**, whatever is the degree of the polynomial, the number of shares is arbitrary, the degree of the polynomial is some number and you have to generate at least a number of points, that is at least the degree of the polynomial + 1. In this way we get some points, in this picture here we have 7 points, in particular you can expect that the abscissa they are known numbers, the real information is provided by the ordinates, so called **shares of fragments**, this is an approach where we are just following the original idea provided by Shamir, when you generate such shares you deliver the shares to the shareholders and you don’t even need to deliver the abscissa, it’s known by construction.

You also understand that you could choose not to use such abscissa 1, 2, 3, 4, 5, you can use whatever abscissa, you will be generating different points but still true that k points are able to **describe the polynomial**. All this game is somewhat ruled by the unisolvence theorem, saying that whatever you are giving R points in the plane there is a **unique polynomial of degree R - 1** going exactly through the R points. This still holds in the case of GF. If I consider more points I can think the points are describing another polynomial, this can be a question. Do you get me? Instead of considering k points, what if I consider more than k? The theorem says that if I consider a higher number of points there is a unique polynomial going through that number of points. If you are constructing the coefficient you will have some zero coefficients for the terms of degree too high. So if you just with reference to this picture if we just make the system considering 6 points we get a polynomial, if we **add another point** we can expect we get a different polynomial but when you are constructing the system what you get is that the coefficient of the term of highest degree will be 0.

It's like imagine you are giving two points, there is one line. What if you take 3 points on the same line? Given 3 points on the same line you don't get a quadratic, **you still get the line**. Because they are linear, so the same holds for all possible degrees of the polynomial, you can expect there is this information, they are not strongly related, the number of points is just associated with the number of shareholders, how many people should get the share. The degree of the polynomial is associated with the **threshold**. Any question? To be more precise I put here another drawing, I tried to draw this polynomial, you see the difference between a traditional polynomial and a polynomial mod something. This is a typical drawing, you see the cut here on number 7. Of course if the polynomial is mod 7 all coefficient will go from 0 to 6, they are integer numbers.

In order to **reconstruct the polynomial** in some subsequent phases where you have described the secret, after some time some shareholder can decide to reconstruct the secret, they can provide key fragments, key shares, and the problem is ok, fine that unique polynomial with k-1 **going through these points**, this is a classical problem, maybe you remember polynomial interpolation, not just a technique for reconstructing a polynomial given some points, is a **framework** provided in order to phase the following problem, we know some points of some unknown curve. Of these unknown functions we know a few points, now I want to **estimate the value of the function** over an abscissa that is not one I provided, I want to know the value of the function on some other points. The one of the possible approach is polynomial interpolation, means ok find the unique polynomial going through the given points and then compute the value over the new points.

This is **polynomial interpolation**. That is just another type of problem, since this is a very well known problem and there are several approaches to do the arithmetics we need for doing interpolation you can find many **well known formulas** already providing you the solution of the problem, I would say that the best known formula is the Lagrange one providing a way very efficient and very simple way to reconstruct the polynomial and this is the general case, so in the case you are given k + 1 points like this one here, you see, you can reconstruct the polynomial, here denoted by L(x), by computing the sum of such terms here, the terms are using the **ordinates** and other terms where every term is defined here, like the product of several components. Keep in mind that actually when we are reconstructing the secret we are not interested in knowing the polynomial, it's just a tool for describing.

This general formula you can immediately say ok I'm interested for the case **x = 0** so I just compute this number by inserting directly number 0 for x, it means that here in this product you will have terms where the x is disappearing, and the only terms we can observer are just abscissa coming from certain points. This is not really important but it's critic, if you need to do that really the secret you will need some **formula of procedure**. This is just the approach, what if you want to accept a new shareholder? Can you do that? So instead of n people you want n + 1. Can you do that? Yes. You just reconstruct the polynomial and you **generate another point.** What if you want to change the polynomial, not the secret? Easy. You need to reconstruct the secret of course and then generate new random polynomial and you will be **delivering the next version** of the shares. Keep in mind that in this case if you change the polynomial and generate the new fragment what about the old fragments? Still working? Yes. They work because they are describing the old polynomial, also describing the secret. Be careful about these details, if you change the polynomial you are not making useless the old fragments, if you change the secret ok, if you **don't change the secret**, I was describing the case.

If you want to give some shareholder a **bigger importance** with respect to the others you can provide to that person more than one share, you can give 2 shares to that person, 3 shares. What is important you are **not giving k shares**, otherwise he will be able to reconstruct the secret. Maybe you want that, just keep in mind. A possible **variant** is not really a variant but just a different way to implement the approach is based on the introduction of a third party, we already seen several approaches where we can imagine here suppose you have a trusted third party, what is the possible role you can assign to such a **third party?** The third party could be if you think of the whole process, who is generating the polynomial, providing the fragments to all the shareholders? One of the shareholders? Maybe, or could be a third party doing the extra job for generating the polynomial and providing the fragments to all the shareholders.

In this case you can think at some **additional services** you can get from a trusted third party. One of the possible extra services is based on the idea you are no longer using the first integer numbers for generating the fragments but you just **use some abscissa**, they are unknown, I mean they are random. What if they are random? Still mod p. What if we are generating random abscissa? Ok, not a real change because given the points you can reconstruct the secret, same as before. If you check the formula here we are not really using the fact that the abscissa this is working for whatever abscissa. So in this case you can **generate a random abscissa** and of course the maximal possible values of any abscissa is p - 1, where p is the module we are using for the polynomial. In this case you can think of such an approach, you can give only the fragments you see here to the shareholders, while the abscissas are just **stored in third party**, so the trusted third party not only is generating random polynomial and delivering fragments but can keep the used abscissa.

So if we do that we understand that of course for reconstructing the secret we need **at least k shareholders** providing the fragments, such information is no longer sufficient, still we don't know the abscissas. To actually reconstruct the secret we need to know all the abscissa and in this case we have to ask the help of the third party. Is that useful? It depends on what you what, depends on the framework you are considering, you can think at some **property in particular**, suppose that you are very concerned and want to know wether k shareholders decide to reconstruct the secret, if k shareholders decide to provide the shares there is no way for the other shareholders to know that the secret has been reconstructed, suppose you want to implement what I called here **crystal safe-box**, what is that? Is just a metaphor, it's the tool we use for giving emphasis on some fact, what I mean? By using the third party in order to reconstruct the secret you will need the help of the third party because it will provide the abscissa, so that is impossible that the k shareholder they want to reconstruct the secret by keeping their approach and **keeping hidden** what they are doing.

If we want to **get an alarm** every time the secret is reconstructed this is a good way. The third party becomes necessary to reconstruct the secret because contains needed information. If the third party is attacked and compromised the information in the third party is **not sufficient to reconstruct** the secret, we still need the k fragments, delivered to the shareholders, so you still need to attack k subjects for reconstruct the secret. What you can ask, what extra services you can ask? You can store in the third party some information derived from the fragments, not the original fragments of course, information derived by some hashing or more complicated process, this is useful because in order to check wether some of the shareholders is providing a fake share, how can you understand?

In this way if the third party is storing information derived from the shares the third party can check so that recognises **if there is a cheater** between I mean when you reconstruct the secret. In the website of the course I inserted another file of this topic, a powerpoint presentation I presented the architecture of an application implementing this framework in the web so that it's possible using this approach to have the generation and reconstruction and management of fragments by using such web application, if you are interested in some more detail you can **download the extra presentation** describing our implementation and you can also download the source code if you want to be. This is a topic the **original paper from Shamir** was in the 79. Good researchers happened after this paper, still today is a field of research where you can add extra security for **dealing with cheaters** or want just to maintain more than one secret, what can you do if you don't want to just replicate many times the same approach? About the application of this approach for a strong password based authentication it's available the possibility to develop a master thesis with myself. Questions?

Access control

No we come to the last topic of the course, the access control. We are doing just an introduction to that. In order to provide the basic framework for analysing solutions for designing new solutions. **Access control**. I'm using slides from Elisa Bertino, actually I provide a set of slide of what we are actually using, anyway it's good to have the possibility to read some **extra information** about the topic, here we are focusing on the very basic part of the problem and this is a type of problem that is strongly associated to logical security, also strongly associated with information security, but is dealing with different aspects and such aspects are not really coming from the usage of cryptography, this is why this type of topics they are typically described **without the usage of cryptography**, still important to have cryptography but focusing on different things.

In particular I would say the simplest example of access control is the one that is provided by a normal regular typical file system in some linux operating system, because you know and not only, you for every file in the file system you have some **permissions**, the permissions typically according to the traditional approach are describing what operations can be done by different subjects. And in the traditional approach you can easily define what are the permissions of one file for the owner of the file, for a group of subjects, just defined as the group to which the owner is belonging and you can define permissions also for other people. Permissions typically they are **read, to write, to execute** the file in the case the file is executable. This is just a very basic approach to access control, they are used for describing some policies so you have in many cases to remember that in all computer security the beginning of your security is the definition of the policy of the security and who can do that, who can read that and so on. Then you can **enforce your policy** by using several tools like cryptography, security protocols, firewalls and other tools very popular. So, access control is that approach where you can think you have to decide what possible usage are allowed to people for your resources, the resources are not necessary just files, they can be **other objects** like special processing units, graphics adapter, network resources, in many cases whatever can be modelled as a logical.

So, in order to have the possibility of enforcing the policy you need an **access control system**, that is the tool that is able to decide what operations can be done on some objects you want to protect. Such permissions, such policies they are deriving from an application scenario that we cannot analyse here. We can just say in the application we need to **define permissions** and policies just to reflect our necessary, our needs. Keep in mind that this type of approach is typically regularly used in database management system, today it's a wider approach, no longer I don't know how many of you know **SELinux?** This is a nice tool. We can decide to install in your linux box the SELinux system. Anyway, you can imagine an access control system like anything able to have an approach like that. You can imagine there is a subject, and the subject you'd like to do something with some object. Generating an **access request** for the object, to read, to write, to delete or whatever you can imagine.

The access request is processed by what is generically called here **reference monitor**, the reference monitor is the guy in charge to manage the permissions. The reference monitor is granting or denying access to the object, this is the basic model, still popular today. This was designed many years ago, when the first people starting to be concerned about this type of problem about level of secrecy to classify documents and starting from their needs. The first researchers describing such an approach was Lampson. You can find the original paper, a nice reading. Today the access control system is implemented in many cases, in most cases as a **software system**, typically it's considered to be a different software system that is capable to make operation on the protected objects from outside, you can integrate that in the same system of course but there is a need separation between such software and the other software running into the box. So what are the objects you **want to protect?**

You see here the definition, anything that **holds data**, a table in your database or a folder, messages, network packets, physical devices, containing information. They can be objects, and in some papers you will see the name protection objects, just the objects to be protected. Keep in mind that the first step in defining the policy is locating what objects need to be protected, not obvious to protect all possible object. Of course the higher the level of protection. The design of the policy and is implementation **until impacting on the performance** of the system. What is a subject? **Subject** typically is a user that wants to read, to use, to write, execute the file, wants to do something with the object. You can have other abstractions describing what is a subject, you can see here some typical examples, groups is a set of user, functions or roles, roles you can think of roles like the actions done by different users, so users imagine you can classify all the actions that can be done by the users.

Of course all such actions are **overlapping**, you analyse all the actions of one possible user, another user and so on. You get a set of actions and this set you can define a function or a role for each action, the same action can be done by different users. You can have **different points of view** denoting the function associated with some users and this means all the users able to run that function are defined by the function. Another possible subject is of course a running process, that can try to some possible action, very typical. In some cases is not so easy to describe a **logical structure** presenting all the interesting features in the set of possible users. The request to access the object is defined by **access mode**, when a subject is asking to do something on the object what is requesting to do? An access, the particular type of access is named access mode, of course in the very basic situation you can think as a simple access mode like read write execute, they are typical ones in the modern operating systems.

You can be **more selective** while describing the access mode, you see here a possible long list of access modes, read write execute, update, delete. You can continue, I don't know, append. Cut. Vertically split and whatever. Of course the access modes they are not defined in a general way, they depend on the framework, the environment you are considering, not always the same access mode are meaningful in different frameworks, different environments. It's interesting to say the same word like read, access mode read, actually is having a **different meaning depending on the object**, this is possible, in order to clarify that you can think of the classical permission, read write execute, the words when you are considering the object file, they are well defined, read a file, write a file, execute the file like a program. If the object is **no longer a file but a directory**, read write execute they are changing the meaning.

Read a directory means having the possibility to know the list of the content of the directory. The access mode write for a directory is granting the possibility to **create a file**, delete a file, rename a file inside the directory. Execute for a directory is meaning you can **change the current directory** to that directory. This is often denoted as search a directory, but it's not clear. Change the current directory to that directory. Just to demonstrate how the same words can have different meanings. **Reference monitor** is that component of the system that should decide wether to grant or deny the access. How such reference monitor should take the decision? There are many approaches, at the beginning two different approaches they were used like those names, discretionary access control, multi level, I don't like multi level. I would call it **mandatory access control**, they are based on different philosophies and the reference monitor can derive from different approaches, different features you may like one of the other depending on the context you are in.

Today there are **newer frameworks,** new paradigms for implementing access control, made by reference monitor you define you will find some examples. So, in any case the reference monitor in order to do the job should know the so called permissions, what is a **permission**? Whatever is describing the policy. You can imagine at the beginning there is the process of defining the access control policies, described in a formal way, you see here in this scheme, all the implementation, description of the policy and reference monitor when it obtains a request on some object will check this database of access permissions and will decide on the possible granting or denying operation. So if we want to actually talk in terms of access permissions we can define a very simple approach where one access permission is defined like a **tuple** like this one, an ordered tuple we have a subject, you have an object and access mode.

If you are introducing in database of access permissions tuples like that this is describing that your policy is **allowing subjects s** to operate over object **o** with access mode **a**. In a shorter way we say that s is having permission a on object o. This is a typical example, Bob is having read permission on file F1. **<Bob, F1, Read>.** In this way the tuple has a positive meaning, if no tuple means you have no right to access. If there is, this is saying ok this guy here can read file F1. You can implement also negative approaches, we will see later the way to specify a forbidden access. This is just for granting access to Bob for reading file F1. This means that if you are not grant permission for you for operating will mean you have no right on that. You can imagine this like a positive approach like a **white listing**, I just write the list of subjects that can do something with some objects. All that is not written is forbidden and reference monitor should deny such operations. Imagine a real system, how many objects? Many.

If you want to consider objects like files, many files, **how many possible subjects**, users they are possible subjects, processing they are possible subjects, many. So in many cases you want to describe the scenario of possible permission, many tuples for describing your policies. To simplify the description of policies it's good to organise subjects and also objects into **hierarchy**, having the possibility to describe a class of subjects, a set. A group of subjects. This is providing the possibility to describe your **policy in a shorter way**. Just can define a rule for the class. This is providing a way for describing your access policies in a shorter way. And this is powerful, so it's good to introduce some hierarchy in the framework, there are few types of possible hierarchies, one possible very interesting hierarchy is the so called part of hierarchy. You can imagine this like some object that is composed by many component objects, for instance the object in the directory.

Files you can say ok all files contained in the directory, providing a way to **define a set of possible objects**, another type of hierarchy this is for subjects, is the so called role hierarchy, you can imagine there are more important roles, less important roles like senior, junior roles, and maybe the senior role is the technical manager, in this way you can define a set of possible subjects, you can introduce a permission, a tuple describing the **permissions for all people** belonging to that role. The membership is also another possible hierarchy, you see here an example very interesting, imagine you have a group like university, a department describing subset of such stuff. In this example here you want to define the policy that in a calendar of the department can be read by university but **written only by the department.** Instead of listing possible subjects for the object calendar you can define a policy like this one, university, subject, calendar, read. All university can read the calendar, then the department calendar write. A short way, otherwise you will need to list many possible subjects. Also the management, I mean the department there are new subjects coming and going out from the set, so in this way you don't need to touch the permissions, touch the policy, just update the meaning of university. And there is another interesting hierarchy of yes.

Subsumption, if you are provided write permissions maybe you have the right to read also. So some permissions are just implied by other, can be implied by other. I mean this is not mandatory but a very natural choice. Under a pure theoretical point of view you can use such way for objects, in order to describe permissions. This way of introducing an intermediate here in the model is providing **some simplification** and you can group objects and subjects and by the belonging of subjects to groups you derive the permissions and define over by using such hierarchies. In this picture here you see some users they are subjects, they belong to some sets and s3 is belonging to these 2 sets. That are defined permissions of some objects here, so you can fuse a model like that in order to describe in a shorter form your policy.

Of course you can do that only in the case **you are able to do that**, I mean you should be able to classify in a correct way subjects, objects so that is not needed to have to define a direct permission like the possible permission for I don't know subjects s5 to do some job on object o1. The intermediation is just a possible feature not so simple to implement all possible case. Another detail is the negative permission, if you want to be able to specify a negative permission, ok this guy here cannot do that, the question could be ok why I need that? In my positive approach all what is **not authorised is denied**, why I need a **negative permission?** Just for cases like this one, imagine you have to deal with some exceptions, so your model of grouping they are not perfect, because for instance subject s1 would belong to this group g1 but actually your model is not perfect, all the components of such group should be granted to access object o1, but s1 is a particular case and should not be granted to use o1. To use this intermediation, you can still do that by accepting some modelling that is not completely, it's **imperfect.**

If you accept that you can say ok s1 belongs to g1 and I explicitly put the rule that s1 cannot use the object o1. This is the idea behind negative permissions. This is **symmetric** to the other possibility, your model I mean subject s3 belongs to this group here, and this group is providing access to this set of objects, you are not capturing the idea s3 is using the object o5, no intermediation here. There is a possible way to generalise the writing of the tuple, **<s, o, a, sign>** where sign can be + or -, meaning can or cannot. A negative permission is interesting in the case you are in some **intermediation**, otherwise you have no need to specify that. This notation you are granting more than what you want, still good is a shorter form but you may need to **deny some possible usage** for managing particular cases. This is nice, this looks simple, it's not.

The security analyst should be making, what is the ability of the security officer here? He should have the ability to **interview the boss**, the boss is not a computer engineer, so running a good interview is not so easy, then after collecting random information the security officer should be able to do the model, not an easy task, any model can be improved, this is a **classical rule**, after that describing the model you can start to implement the policies. When dealing with such control, how you implement the fact you can **change the permission** to some subject? This is not contradicting what I already said, but what is deriving from this description, some people, one officer having a big power can do whatever he wants, changing permissions, just because he's the officer that should implement the policy of the enterprise, he has the power of changing permission and you understand it's a possible point of failure, its password is captured by attackers, they can change permissions and so on.

Also there is a theoretical problem that is the following, this is a file created by user, the user is the owner according to the classical model, is good that the user has the possibility to choose **what subject can access the file?** I create a file, in a shared system, now after creating the file I can think of what I've created, I'm the owner, can I choose what subjects are authorised to read or change my file? Is it good? Is a classical point of view, the file is mine, so I decide what people can see the file. This is exploited in the case of trojan horses running in the system. It's a theoretically interesting problem deciding who is in charge to decide who can access the object, the owner or some other way to decide who can access the file, is good only the owner decides? Can the owner decide? Or is the system that decides general policies and setting permissions in some automatic way? Different ways. We will see deeply 2 examples.

I mentioned SELinux, that is a **mandatory approach**, today mandatory approach is considered to be more secure but the discretionary approach is considered **more natural**. We will see next time example of discretionary approach. Example of mandatory approach, they are typical instances representing the two approaches, very good examples. When you are continuing in the case continue to whatever the approach we have described, write the tuple of your permission and typically when you are considering the discretionary approach you define what is called an **access control matrix**, here you can see a matrix, you see here in this column the name of subjects, Alice, Bill. The name of object. In the cells you see the permissions, here you see that Alice is having a permission on file. This is a typical example, a matrix where every column is corresponding to an object, it will be a **big matrix.** The name is the access control matrix and what happens in the reality is that this matrix is a **sparse matrix**, you remember the meaning of sparse? What is a sparse matrix? If you describe the matrix by columns for ever object you are listing what subjects have the permission to do something. This is why we call it a description capability list. Good.

You can describe the sparse matrix by describing the **non null cells**, you can build a data structure listed by lines or columns, and since the role here of lines and columns is associated with the model you see here lines associated with subjects, columns associated with objects, if you describe the matrix by columns for every object you are listing what subjects have the permission to do something, this is why all such a description of capability is **access list capability**. Most cells will be empty and you have few cells different, to describe it by lines you will give an array of lists. Object and permissions, you have the same here of other possible subjects. Every list here is describing what the subject **can do**, so this is a capability list. You can describe the same matrix by columns, if you describe in this way you have a data structure like this one, the semantic is different, you have an array associate with the objects and you have a list of subjects for every object.the list is describing in the similar way what is the subject, what are the permissions and so on. This is what is called **access control list.** The usage of the traditional way of represent a sparse matrix in this framework here is just providing a different possible philosophy of enforcing the access control. This is why when we are looking at real solutions we base on capability list, solutions based on access control lists.

When you are considering framework like that in addition to permissions you have to manage to represent the policies there are some basic operations you need to perform in such a model, because when you have to manage the **dynamic of the system** you have to grant permission, to revoke permissions, to check wether some people they are having permissions to do something, they are the basic operations in access control. This is a practical way to reflect what we need in the discretionary access control, the natural part. The other possible mode is complex, we will see by second. Any question? Since the 2 examples more than the time we are remaining, I prefer to spend some time together for doing something we have not done a lot. **Checksum exercise** for the exam. If you are still here maybe you can handle the flow of information you understand what we are talking about, this is a good moment for checking some examples for assignment for the exam. I pick it random.

Exam cns20170623.odt solution

With high probability, paper like this one asking to use english, ok. In some cases there is just my I don't know how to call it, in some cases I provide something like that just a few simple list, I ask people to check wether, don't smile, people make mistakes here. So, true or false? F, F, F, F, T, T, T, ?, ?, ?

Question 2, hashing. Prove that a strongly collision resistant hashing function is also weakly collision resistant.

We prove what is asked here by not weakly implies not strongly. Describe the birthday attack, what makes a hash function weak or robust with respect to this attack? Is so boring to describe it, I would say it's nice to give an answer to the second part here, what makes a hashing function robust or weak with respect to the birthday attack. I remember a typical wrong answer here is I choose in order to resistant to the birthday attack I choose a strongly collision resistant. No, the birthday attack can be used **whatever type of hashing function**, if the hashing function is very good, strongly resistant collision, ok the birthday attack will do the damages we know, so you can run many attempts, there is the square root, that is the typical number of expected attempts. This holds for every hashing function, if the hashing function is not resistant the birthday attack will be more powerful, just that. But we are not concerned about that, we are just asking can you attack a hashing function by birthday attack? Yes. What makes difference is the size of output of the hashing function. The longer the output, the harder is the birthday attack, the number of expected attempts for the attacker is associated with the length of the hash, if you remember SHA-1 providing an output of 160 bits the expected number of attempts for finding a collision is 2^{80}. The number of expected attempts depends on the **size of the hash**, in order to make your function more robust you need to make the output longer. Just that. Of course if the function is not collision resistant the birthday attack will be more powerful but we are not considering such a detail.

Next question here is define what keyed hashing is. Can the attacker still run the birthday attack?

Keyed hashing is ok a theoretical question, you should answer ok I insert a key and the algorithm, what is nice to say if you decide to use some hashing function one hashing function like SHA-256 that is one precise hashing function, if you use a key it becomes into a family of hashing functions, why? Because you can consider one different member of the family every SHA-256 coupled with one particular key, it's like having a different hashing function when changing the key. You can **change your view**, when you read the formalism you see the key is concatenated with the message, it looks like we are changing the input, if you consider this possibility like key is belonging to the framework, hashing function is defined by the mathematical function and the key, you **change the key you change the function.** Like a family of algorithms. In any case, when you have a key hashing function the birthday attack is made useless, why? Just because the attacker cannot check wether he got success because it's still true that is easy to generate a collision but is that a good collision or bad? Because you need that the collision generated by using the same key, you don't know the key. So in the case of HMAC for instance you have a keyed hashing function that is robust against the birthday attack.

Then there is a funny question, how does answer to the question above changes if the attacker knows the key?

You use hashing function, the keyed approach, this is making harder the birthday attack, because the attacker doesn't know the key, what happens if the attacker knows the key? It means the hashing function is no longer keyed, you can consider as a normal not keyed hashing function, is vulnerable with respect to the birthday attack.

Provide a black box description of TLS.

This generated panic I remember, and many people answered writing lot of details about TLS. I appreciate many details but here I want to check your ability to describe a security protocol to somebody that is not a computer engineer. **Black box description means what you can do**, can obtain by using such a protocol, not how it works. So you can obtain features like data integrity, confidentiality, you can do that for an application, port to port, you run it over IP and whatever but here talking about handshaking, ways for setting up a session key, this is wrong, this is not belonging to the right answer, and this is not easy to describe, you have to be able to reason at some higher level, you can do that only if you have really understood the protocol.

Provide 2 cases of use of TLS.

This is trivial, one case of use is the best known is the **https protocol**, another case of use is the addition TLS to insecure protocol, the usage of TLS for making secure ftp and so on. Fully compliant. Assume we have some perfect software. Describe what attacks can be still successful against TLS. This is a strange question, the protocol is good, the software, what can the attacker do? The idea here is don't forget we need an infrastructure, not depending on your application, the infrastructure is based on digital certificate and you need something that is working, you can have the best TLS design, the best implementation but if certification authorities are attacked, if there is man in the middle, validity of digital certificate, your approach can be **broken**. This is a typical attack against a very good TLS, modern and operated by a perfect software that is not existing.

Describe at your best the following IPtables commands, also clarifying which perimeter they protect.

I normally don't ask to write IPtables command, I'm not even myself able to check the syntax, I don't ask you to remember the syntax. But you should be able to recognise the meaning of a command like this one. You should explain, we have just seen IPtables, it's easy to answer this question, I would say it's interesting the second part, clarifying the perimeter, what we are protecting? If the chain is output we are checking the packets originated by the firewall, not protecting with this line here a network, because for protecting the network we need to use the forward chain. The firewall is protecting the other host, in the case of output and input chains we are focusing on packets going to the firewall or exiting from the firewall. This is interesting from protecting the firewall itself, output input, interesting for protecting the network, I guess it's simple for you to understand the meaning, describing the case we are using the TCP protocol, packets are incoming in this network adapter, they are expected to be routed outside, destination port is 22, source port is belonging to this range. We want that the state is new, meaning the creation of a new TCP connection, in this case you want to accept.

Short questions, I want to see short answers. I don't want to see too many details, why HRU is vulnerable to trojan horse?

Let me see how you can compute $2^{200} \mod 127$ by hand. How to do that? You should use the Euler theorem, ensuring that some powers, in particular is good to see that 127 is a very special number, and so first it's a prime number, meaning that 2^{126} is equal to 1. So you can decrement the exponent, but also is true that 2^{128} you know the result. Is 1 again. So using these ingredients you can reduce the exponent to reach a very small number. No real computation arithmetic is needed. Questions? **Have a nice weekend.**

Access control models

Both types are the basis for further deepening, they were introduced several years ago, established the theory, new frameworks be developed, to go deeper you need to examine more recent papers, all the recent literature is **inspired to these basic results**, it's important you are familiar with the basic models. What we have already introduced is the typical approach describing the **discretionary access model**, and if you remember we introduced concepts like matrix of permissions, describing subjects, objects and permissions, the other possibility of using such matrix as data structure or managing access control list or capability list, just details.

Today we want to focus on some more detailed example, that establish the theoretical framework within we have to move for defining DAC and MAC approaches. According to the original definition a **discretionary access control model** is the model where the owner of some resource can decide on a discretionary base what other subjects are enabled to use the resources, so it's granting permissions, in the **mandatory access control model**, the approach for granting and revoking permissions is no longer based on the fact the owner can decide something, it's based on some general rules, defined on the system, at system level, meaning that even the owner of some resource cannot decide to grant permissions to somebody, the owner is constrained to operate within the framework defined by the rules.

Indeed in this last case many people talk about a rule based access control model, defined by rules, general rules, the main difference is that in the first case the owner can make decisions about other people, other subjects accessing the resources, in the second case the decisions not taken from the owner but they are coming from a general policy, defined for all possible owners, meaningful at system level, described from a set of rules, such rules describe the framework within people can get the permissions for reading or writing the files, this is the basic difference, very important to focus on such difference. In the first case **social engineering** is more important, some owner can be tricked into deciding some grant permissions for some resources.

In the second model a social engineering is harder, because people are **not having tools for granting permissions**, everything is ruled by a system accepting general rules, valid for all categories of users, we will see some examples. There are other models, we are not going into details, all these models can be used, either some **variant** of the discretionary access model or a variant of the mandatory access control mode, so again I want to point out the idea that these **two basic approaches** are enabling you to deepen and study recent proposals, keep in mind that there is no general solution, I mean general solution would be a solution working in every possible case, we can derive good solutions for well defined settings, very specific, and in this case we are allowed to go into further details and make the general models more adaptive to the current situation.

I have already defined the meaning of **DAC**, where the power of granting permission is given to subjects, subjects that are having the right in particular the ownership of some resources, the concept of ownership is well defined, the idea is that when we are operating according to the original model, where a subject makes a request for accessing some object and while making the request it specifies the **mode of access**, what permission the subject is asking, there is a **reference monitor**, a sub system obtaining information from some database describing a set of permissions, and by examining such database the reference model can decide wether to allow or deny the access to the object. You remember the terminology, subject wants to access some object, the object is the resource, the subject is the person or the software or whatever actor that is having the ability to use some resources.

Some simple advantages of such approach are based on the idea that is very simple and flexible the way to **specify the policy**, we can specify whatever policy by just updating the matrix of access permissions and the matrix is actually defining the current state, but the database can be thought as a table storing all information about such policies, in addition to being flexible this approach is also well known and supported by most modern operating systems and many DBMS. There are several drawbacks, the main two are the **vulnerability** to social engineering attacks and trojan horses attacks. The **theoretical model** has been developed by these 3 researchers, they provided the HRU model, a well known model, it's the origin of several new models, is defining the framework, it also defines the **limits of the approach**, because we will see the approach actually is suffering from some practical drawbacks, the researchers introduced the idea of authorisation system, and this is the base of the classical DAC model.

In this paper they introduce the notion of **authorisation system and safety**. I guess the 2 names authorisation system and safety are very intuitive, but we will be giving precise definitions of them. So, where are we moving? In the usual framework, defined by the environment where we introduced the concept of **subjects, objects, access right**. So we have a set of objects, subjects and a set of access right. We have an **access matrix**, every entry of the matrix, I mean an entry is defined by a line and by a column, if we associate lines to subjects and columns to objects the entry of the matrix is specifying what is the set of permissions granted to the subject of the line on the object corresponding to the column. If the cell is empty it means the subject is having no permission on the object. This is clear and simple.

Now we want to be able to **operate in practise** with such type of system, what does it mean? Use it in the reality, it means that during operations people may be creating files, this is very natural in every computational system, if I create a file I'm creating an object, so it means I have to add a column to my matrix, and I have to define what subjects are allowed to use in what way such an object, I want to introduce the clear idea that **the matrix should be managed**, what does it mean? Basically means being able to change the matrix, otherwise we cannot reflect the current state of the system. In order to be able to change the matrix some primitive operations are defined, you see here the list of primitive operations, this is defined by 6 primitives operations and they allow to manipulate the set of subjects, objects and content of the matrix.

So we can **create a subject**, meaning add a line to the matrix, we can delete a subject, meaning delete a line from the matrix, we can create a object, adding a column to the matrix, also we can delete the object. We need to be able to change the content of the matrix, so we consider two primitive operations like enter, an operation for adding a new permission into some cell of the matrix, and delete a permission. You see the entry of the matrix is described like M_{so}, an entry for subject s and object o. This description is independent by the way we represent the matrix by access lists, by capabilities. We are reasoning at **higher level**, and it's sufficient to talk about the matrix. Also I want to point out that this set of primitive operation is not sufficient for reflecting our needs, when I'm creating a file I'm doing some operations like add a column, as a minimum I should add in this column in one entry of the new column that is the entry associated with the owner of the file, I should **define the permission for the owner**, read and write as a minimum approach. We understand that associated with a very simple operation we need to have the possibility of executing a list of primitive operations, not a single one. Such primitive operations are too primitive for reflecting a single action from a user.

This is why in such type of system other type of operations are introduced, they are known as **commands**, it's possible what is a command? Looks like an algorithm, using simple basic control structures, and primitive operations, here we can see a general structure of command, in this case the command. Characterised by a name, and a set of parameters. The set of parameters here is described like a **sequence of symbols**, what is such a set of parameters? It can be composed by some subjects, by some objects, and let me stop here. Again, the structure of the command is based on the fact we can use selection like in this way, if permission r1 is in this entry, this is the matrix, unique. What are s1 o1? Just a pair of numbers, information, derived from the parameters, some of this x **acting like parameters** are describing s1 and o1 and again s2 and o2 and so on. So, the structure of the command is defined by a **set of test** like you see here, the tests are meant to be executed in logical and, if it's true that such a permission is in this entry, another permission is in another entry and so on, then a **list of operations**, a list of primitive operations defining a sequence of operations to be carried out to reflect a higher level operation made by the user.

And here we have **an example**. Command create file, subject s, object f. You see here in particular we have no need to specify the selection to use the if, it's very simple, create the object f, create the file, enter permission o into the entry of the matrix associated with the subject s and the new column f, what is permission o? **Ownership**. So this command is reflecting the fact subject s is creating a file f and he's the owner of such a file. Then you can easily guess that is very natural to insert in the same entry of the matrix two further permissions, read and write, enter permission r, read, into the entry, and enter permission write into the entry. I guess it's simple. This is the **general scheme**, very general scheme, you can have whatever number of test, in particular 0 test. Of course you can have many types of commands, there are 2 particular cases very interesting, the command called to be **mono operational**, the mono operational command is the one using one primitive operation, not a sequence. I mean the sequence is long 1. And there is also the **mono conditional command**, meaning I specify only one condition, only one if, or I can define a **bi conditional**, bi operational, you can combine the number of conditions and operations.

Let's see a second example, **command grant read**, the command grant read is a command reflecting the fact that some owner of a resource wants to grant the read permission to other subject. The parameters here are the owner s, the other subject p and the object f. In this case for granting the permission we need to check wether the permission o ownership is in the entry. So the command is checking wether subject s is the owner of file f. If yes, he's having the right to **grant permissions to other subjects**, then enter permission r into another entry of the matrix, the entry associated with subject p and object f. Be aware that this is just a possible choice of implementing the grant read command, the policy is stating that all the owners of some objects can grant permissions to do other subjects for reading such objects, the command grant read is not written by the subjects, **defined at system level**, because it's obtained by the policy of the organisation.

The basic ingredients we have seen are the concepts of matrix, permissions, the access matrix, the concept of subjects, objects, permissions, primitive operations, and more general operations. We have called them commands. This is just the basic framework, basic terminology. According to the HRU model a **protection system** is defined as a finite set of rights, a finite set of commands. You may think at the very beginning just the basic I mean the commands are defined and at the very beginning the access matrix is very elementary, just contains the possibility of using the commands over the matrix, to change their characteristics from the object. **No permissions for all subjects,** then you can use the commands for inserting information into the entry of the matrix. So I want to stress the concept that a protection system is evolving over time, after every command I expect after the execution of a command the content of the matrix will be changed.

It's not mandatory, I mean, given a command to be executed there is a good probability that after the execution of the command **the matrix it's changed**, but maybe not changed, imagine that such a command grant read is requested by a subject not having the ownership of the object, the command is executed but the test is failing here, no action is done, meaning that the matrix is not changed after that. **The current state** of the system is described by the access matrix. This is introduced in some formal way, so here in practise you read two definitions, this is the protection system, a finite set of rights and commands, and the definition of the **state of the protection system**, what is the definition? The access matrix, we know already what it is. The concept of state is so important, right? You understand the content of the matrix is important, what if the matrix is allowing everybody to do whatever on all possible resources? That is a state. **Is it safe and secure?** No. Of course.

An **empty matrix** would define a system where nobody could do anything, useless, not acceptable. So you understand that there are possible different states and you can consider a state as a good state more good, better, worse, bad. It's easy to define what is a good state? Not so easy, more technical question. Of course in general when you consider a whatever system, what is the state of a system? Software-hardware platform, **what is the state?** Is the content of all memories including the cash, including the registries, including the ram, this is defining the state of course. This is a very **general concept**, useless to make reference to this, it's too general. But while defining the state of the protection system we consider a subset of the state of the platform that can be described by the access control matrix. So I try to relate the concept of state of protection system defined as access control matrix to a **more general concept of state of a system.** And actually the access control matrix is describing a subset of information belonging to the state of a hardware software platform. And such a matrix is sufficient for describing our needs in terms of policies to be defined, enforced, checked, so that our system can operate and while the system is operating the content of the access matrix is changing in the time, is evolving.

We try to introduce the concept of **safeness**, what is a good state? A bad state? What is a safe state? This is not quite clear how we should define the concept of good state. In order to be enabled to define an acceptable concept a good compromise we use such terminology, an access matrix it's said to be **leaking a right.** In this framework leak is not providing a negative concept, can be negative but also positive, depends from what you want. The access matrix is said to be leaking a right r if there is a command, one of the commands defining the system, that is **adding the right r into an entry** of the access matrix, such entry previously was not containing permission r. Very simple. We say that the state of the system is leaking a right r if the read possibility of entering the right r into an entry of the matrix, previously not containing right r.

You can rephrase that in some **more formal way**, there exist a subject s, there exist an object o such that right r is not belonging to such an entry and there exists a command c such that after the execution of the command permission r is belonging to the entry when I'm considering the new version of the matrix m'. Meaning that the matrix is changed after a command. So the leakage is **not necessarily a bad concept**, in some cases you want to leak a right, when we are creating a file we need to leak a right. So when we can say that the leak is bad? Because actually you want to try to associate the concept of safeness of the control system with the concept of leakage of permissions. This is the intent of the authors. So under a more general point of view we could try to follow the definition, the state is safe definition 1, just a proposal.

We could define the state as a safe state if the access to a resource **without the knowledge of the owner** is impossible, this is a good definition, very strong, but not really what we want, it's too strict, why? Because every time somebody has to read a file we need an explicit approval by the owner of the file, this is the meaning, **too strict and not practical.** So we may be wanting to accept the idea that the owner of the resource, of the file is granting a permission to somebody for reading the file, so that next time the other person that got the permission is not asking to the owner hey let me see your file. The system is implementing the policy allowing the guy to read the file, the owner **has provided the authorisation**. So it means under a philosophical point of view that still true that the owner of the resource is knowing what are the possible consequences of, I mean the possible new evolution of the state of the matrix. In other terms we don't like that leaking a permission can lead to **further leakage** of the permission, to other subjects that would be not authorised.

What is a **bad state?** A state where a subject can grant a permission to some other subject over an object, as a consequence there is another command allowing to entry another permission in some other entries that was not approved by the owner. This is a leakage, not authorised, this is an example of **unsafe protection system.** Of course this is a teaching example, very easy, the commands here are not looking very smart but anyway imagine we have a command grant execution permission, grant execute, meaning that subject s is wanting to **grant some execution permission to another subject** p, over an object f. So the command can be operating like the follows, test if s is the owner of file f, if permission ownership is in the entry, subject s file f, then the system can change the content of the matrix for subject p and object f by adding the new permission execution. This is just the definition of granting the execution permission. Imagine we have another command like this one, this is a stupid command, we call it **modify own right.** Meaning that if some subject is having the permission to execute an object, then can enter a new permission w right into the same entry. So you see here the content of the command, if subject s is having permission of execution over the object f then he can ask for the execution of this command here and since this test is successful then the new permission w is entered into the entry just because the subject was having the permission to execute the file.

Here we have an unwanted leakage. So this is unsafe. Why? Because when the owner of the resource is granting the permission it's not obvious that the owner was granting the permission to write. This is the **original statement** by the authors, the definition of safety. Suppose a subject is planning to give a subject s' the right r to object o. The natural question is wether the current access matrix, with r entered into (s', o) is such that right r could be subsequently be entered somewhere new place. This slide here is explaining the issue with previous command. So we get some **formal definition** from the previous statements, if we are given a protection system, means giving a set of permissions, a set of commands. So giving a protection system and a permission r also we can imagine there is some initial configuration for the access matrix, there is any initial configuration, it depends on what commands are providing what type of operations. So if we are given some protection system we consider some particular right, start by a particular configuration q0, we say the **initial configuration** is unsafe the initial configuration if there exist a configuration q that can be reached by starting by configuration q0 and using commands, the available commands, suppose by starting by q0 by using the provided commands we can **reach some configuration** so q is reachable from q0. And there exist a command here indicated by the symbol alpha that **leaks the right r** from the current configuration q.

This is the definition, so we are saying when a protection system is unsafe with respect to the specific permission. Unsafe from a specific permission if starting by some initial configuration, following all the official rules, initial configuration, use the official commands, then I can reach a configuration q and then there exist another command alpha that leaks right r from q. Giving the permission to add the command r to some entry of the matrix q. We say q0 is safe for some permission if it's not unsafe. You see here also **the simpler way to give definition** more I would say less formal, the non formal definition you see here a state of protection system the state is the matrix, is safe with respect to some permission r if there is no sequence of commands that can transform the matrix into a new matrix that leaks r. Not hard. After some definition we are able to understand, very natural. Is interesting that in the original paper the authors proved that the problem of establishing wether a given access matrix and a given right r are verifying the safeness, the definition of safety such problem is note decidable.

So **verifying the safety of a matrix** with respect to a permission r is not decidable, you remember what is meaning not decidable. Means we are talking about decision problem, this is a problem where the answer is yes or not. The problem is **undecidable** if there is no algorithm providing an answer, providing an answer yes when it should be yes and no. I mean **the algorithm is not existing**, difference from saying we don't know the algorithm, in many cases we don't know the algorithm but we can find it. Here there is the proof that the problem is not decidable, by reducing the well known **halting problem** for a Turing machine from this problem. So if we make the hypothesis of being able to decide such a problem we can use the algorithm for solving the halting problem and we know that is impossible, a reduction from halting problem.

The author shows **some particular cases,** the safety is decidable for mono operational protection systems. Undecidable for bi conditional monotonic protection system. A system is said to be monotonic if permission **can be only added**, not removed. So if you have a monotonic protection system. In the case of monotonic protection system permissions are just implemented, if we have 2 conditions, it's still undecidable. It's decidable for mono conditional protection systems, we are near the boundary between what is decidable or not. What is interesting to say is that is a model like this one is able to capture in some intuitive way what we want and don't want from a protection system, the consequent problems are not manageable, we cannot accept there are undecidable problems behind a **theoretical framework**, so this is motivating some more recent approaches. We are not having all general possibilities for the system to evolve in all possible ways.

This is allowing a somewhat simpler approach where you can define simpler models and from simpler models you can derive simpler approaches not hiding untraceable problems or undecidable problems. You see here some **possible extensions of the approach**, just the names, I don't comment that, it's a very wide subject, so just be aware we can define stronger versions, versions where you have some hypothesis implicit, other protection systems aware of the context. Also available in this type of approach in most of DBMS. I want to show you now the reason why this type of model is **failing against a trojan horse.** A trojan horse is an application typically people are tricked to download trojan horse because they think it's some utility for the maintenance of the system and actually the software is maybe providing that utility is what I want, they also doing some other hidden job in the system. This is why they are called trojan horses. Typically if a user is downloading a trojan it will be a software running with the same permissions of the users, the same capabilities of the user.

Suppose there is some object o1 and this is an object for which user ada is having the read permission. Suppose there is user Bob, another subject, Bob wants to read such an object o1 but Bob is not allowed to read it, he's not having the permission to access the object. So what happens when there is a trojan horse? If ada is downloading some software when running it's creating a process, such a process is having **the same rights of ada** because it's owned by ada and the process makes several steps, it can read object o1 and write the content of object o1 to some other object o2. And maybe it happens that for object o2 ada is having read permission, write permission so that the trojan horse can write information to o2 and Bob is having read permission. In this case we get away for which a user not having a permission to read some file can actually read the file because **a process has copied the content of a file** on some new object or not necessarily new, also some old object, just append or rewrite the content of the object by inserting the content of object o1.

This is like a trojan horse is acting in order to **violate the privacy** or I mean the clearance for the users that are allowed or not allowed to see some information. So DAC models in general are not protecting information against trojan horse, no mechanism for **controlling the flow of information**, this is a general drawback of the approach, no way to control the flow of information, you can create some flows making able a software to create a new object, allowing some other user to read information that the subject would not read, not authorised to read. In the **mandatory access control model** the design has been developed so this type of access is prevented. They are providing a solution to this type of problem, MAC models are not perfect. I would start MAC models. We can anticipate the break.

The good way is that of course when people are able grant permissions of course you get the state is unsafe, also we told that we agreed the fact that being leaked is not necessarily a bad thing, it could be ok, but actually the real question is if I want to exclude the classical cases where the leakage is what I want, the case where I get leakages when I don't want **how can I approach?** Temporarily removing from the matrix the line of the subject allowing to grant permission so that type of trivial consequences is removed from the analysis. If you **remove from the matrix** all the subjects able to grant permissions so that they are introducing by definition changes in the matrix that are leaked permissions we remove such subjects and we get the consequence is not possible to reach the introduction of new entries that are wanted, so you can continue the analysis by considering other introduction of leakages. This is somewhat also discussed in the original paper.

Bell and Labadula model

Second I have another comment but I just want to announce I made **a mistake while talking of IPtables** and I forgot to discuss my mistake so I prepared a slide, would be to be shown last time but I forgot, don't let me forget again making some new introducing some further details. I told something not correct, this will be done. Now we want to discuss an example of **MAC model** and the mandatory access control example that you want to discuss is a **multi level access control model**. So the multi level security is a particular type of mandatory access control model, introducing some confusion in the meaning of MAC, MAC is both for mandatory access control and multi level access control, meaning that multi level is a particular case of mandatory, very wide used today and more modern solutions all of them are derived from the original one, the model by Bell and Lapadula. This is followed in many approaches both for secure operating systems and for secure DBMS. The Bell and Lapadula model introduced this new way to see a rule based security allowing the introduction of a multi level security.

Of course this is originated in some **military environment** with unclassified documents, confidential documents, secret documents, top secret documents and so on, Bell and Lapadula had a proposal for reflecting **the needs the application** needs. So again in the model we have objects and subjects and the meaning of subjects and object is still the same. So no need to discuss further such definitions. In the model only 3 simple access mode are considered. **Read, append, or write.** And the meaning of these 3 permissions are intuitive one, reading ok, append is a modification operation meaning you can **add extra information** to the resource, you may think of the file, add extra content, not meaning the ability to read the content of the file, append means you can just add lines, ok. Lines of whatever string of bits and write permission meaning the ability of modifying in some arbitrary way the file also in the ability of reading the file. Write is implying the ability to read, append is not.

In the BL model we are having 3 access modes, read append write. In order to reflect the need of having some levels of authorisation for accessing information the model is providing so called **access classes**, so we are now going to define what an access class is. Said in some intuitive way an access class represents the concept that some people, I would say subjects, they are having some clearance level for seeing information. Means ok I can see all the information **up to that level**, I cannot go further to higher levels. And a very similar concept is defined over objects, an object is also characterised by some level I will need some clearance to access the object, we talk about **sensitivity of object**, the general framework for describing clearance of subject and sensitivity of the object is the concept of **access class,** general description applied to subjects allowing to describe the clearance of the subject, understanding until what level subjects can access information.

And in the case of objects, the access class is describing the sensitivity of the object, the clearance of the subject should be at least equal to the sensitivity of the object. We need to be more careful, it's not so simple and we need to introduce some definition for being able to better describe what we are talking about. The important concept is access class, when you consider an object or a subject it's characterised by an access class, the terms **clearance and sensitivity** are describing the same concept but clearance applied to subjects, sensitivity applied to objects. An access class is characterised by definition by 2 components, a

security level and a category set. Let’s understand what is a security level and category set. You can imagine the security level like an element of some total order set. So even just a number belonging to some predefined interval. And ok. Security level is just one of these number, in this example here the security levels are defined by military classical definition document, top secret secret confidential, unclassified. We have a **total order** meaning that top secret is greater than secret greater than confidential greater than unclassified. The category set is just a set of elements and they are useful for defining the context of subjects or objects work in some particular access class. In this example here you see a set of 4 categories: army, navy, air force, nuclear. Again deriving the terminology from military settings. An access class is a **security level** meaning secret or unclassified and a set of elements. If this is the set of possible categories an access class could be composed by one security level and a subset of this category set. In order to be able to better understand the meaning and why we are defining in this way access class let me skip, let’s go to the **2 main points of the policy.**

According to the BL model there are 2 basic principles to be respected for ensuring the multi level access control. One of them is **no read up,** meaning that subjects should not be allowed to read objects having a sensitivity higher than clearance level. No read up. Don’t read too much. There is a symmetric policy, **no write down.** We don’t want the system is allowing to some subjects having maybe a high clearance to write objects having a lower clearance. In a more precise way, we don’t want people having some particular clearance being able to write objects having a sensitivity level associated to a lower clearance. So it means that if I am a subject entitled to read top secret documents I should not be able to write unclassified documents. You understand the reason, I could access a very sensitive document and I could write the content into a some unclassified document, making it available to everybody. This is the **basic philosophy** of no read up, no write down. Both them make sense, you need both of them, one of them is not sufficient, the intuitive one is no read up. You cannot read documents more sensitive than your clearance.

The write down could be the solution for the leakage information, for not having the leakage you could prevent people entitled to read very sensitive documents to write less sensitive documents. You understand that is an approach of type mandatory, we are just providing rules and the owner of a resource cannot just decide to grant or revoking permission for accessing resources. We have a system wide level definition for policy and all users whatever is the corresponding clearance should respect such rules. So in practise we **define an access class**, in a more formal way, as a pair, the pair is defined like a level, an access class is defined as a pair where the first item is a level of security and what is a level of security? The security level is just an ordered set of elements so it’s just an element from the ordered set, this can be confidential, you remember the unclassified, confidential, secret, top secret, could be one of the 4 possible levels.

The second item is a **set of categories**, could be army, army and navy, and so on. This is just the definition of access class, now we are going to introduce a **dominance relationship** between the access classes, this is a partial order, a relation where we have some properties, the transitive properties, the partial order is corresponding to such dominant relations that is defined as you see here, an access class is dominating another access class, if it happens that Li is greater or equal to Lk, security levels, if the set category I is containing the set category k, the domination, they are points in the **plane access classes**. But the plane is not because the plane is providing good properties and possibility of comparing items. Here in some cases access classes are not comparable. You see here an example of lattice, a diagram describing the partial order, where you see here the arrows linking 2 nodes, if it happens the destination is a destination of the arrow is dominating the node in the original arrow, so secret army and nuclear is dominating this other pair, secret and nuclear, secret is greater or equal and this set army nuclear is containing the set nuclear, so this is dominating.

We have an arrow describing such permission. At the bottom level you have the empty set that is the set containing no items and the unclassified security level. In this diagram there are no **transitive arrows**, I mean if this guy here is dominating this guy here and this one here then the top here is dominating the bottom one. You could draw an arrow, transitive arrows are not shown in this type of diagram. Is the opposite, the transitive reduction. This node here is not comparable to this node here. Why? Because even if it’s true that top secret is greater than secret army is not comparable to nuclear, there is no containment between them. Remember this is a partial order, not a total order. We can have pair of access classes not comparable if any pair would be comparable we would have a total order, but what we need here to better describe the requirements of the security policy is describing it by using a **partial order** and the lattice is the structure showing this here.

We can talk about **strict domination**, so we can have here three examples, 3 access classes, top secret, nuclear army, top secret nuclear, confidential army, it happens that c1 is dominating c2. Because ts is equal to ds and this set here includes contains as a subset this here. C1 is strictly greater than c3, a strict domination, because ts is strictly greater than confidential. C2 and c3 are incomparable, even if the first item is comparable the second one is not. In general the state of the system so again we talk about **protection system**, here the authors are not using the term protection system. The state of a system is described by a pair where the first item is the set of current accesses, what is such a set? A set of triples of the type subject, object, mode of access. Meaning that actually having currently subject s is making an access to object o by using the permission m. Remember the permissions are read, append, write. L is the level function, what is that? A mapping you see here the good definition of level function, a mapping transforming every object or subject into an access class. So every object and every subject is associated to an access class. Our preliminary discussion where we were saying in order to characterise the security model we were associating an access class to every subject, to every object, using the term clearance for subjects, sensitiveness for objects. This is just a **mapping from objects and subjects** to access classes.

Now we can state in some formal way the no read up policy, so the idea is the intuitive one we have already provided, you can define some more formal way the property, no read up, the property is also known as the simple security property, no read up. A given state is satisfying the property if it happens that for each element that is a triple of a, remember that a is the set of **current accesses**, so for every element of a one of the following conditions holds: m is equal to append, of course if you are accessing some objects by using append you cannot read the object, you can violate the no read up, you cannot read at all. Or you can read permission is read, or permission is write because the write permission is implying the read permission, this means the ability to read and it happens that the level of subject is greater than or equal to the level of object. So this is the meaning that the subject s cannot access in read mode some object having a **higher sensitivity**. Remember these are access classes, access class of s, access class of o. So we are using the relation domination. For example a subject with access class confidential army is not allowed to read objects with access class confidential. Why? Because there is no property satisfied they are not comparable army, navy air force, not dominance. Or unclassified air force, not allowed because there is no dominance, army is not comparable to air force.

This is allowing of defining a more powerful concept of **access clearance**, not just related to the access level, security level, top secret, secret, confidential, also related to the category because some documents can be confidential for some categories but not necessarily for other categories. This is the **formalisation** of the property the simplest security, no read up. The formalisation of the other property no write down is also very simple, this is called also **star property**, this is a synonymous. A given state Al is satisfying no write down if for each element A described by a triple subject object mode of access, belonging to A, the current set of accesses, one of the following conditions holds, just one: remember, the goal here is **no write down**. M is equal to read, if the access mode is read you cannot write at all, no danger.

If m is equal to append so append is implying the ability to **add information** to a resource, even if the guy is not allowed to read, and the sensitivity level of the access class of the object o is greater or equal to the access class of subject s. S can work with higher level objects in append mode. Meaning that ok they are objects belonging to some higher level of security but I'm just appending information, not reading so no read up is preserved and I'm adding some extra information, I don't provide information to lower sensitivity class. Or even m is the write permission, I can write and read document, and the two items, subject and objects are having the same level access class. According to this policy a subject having an access class like this one, confidential army nuclear is not allowed to append data into object having this access class, unclassified army nuclear.

You see that this access class here is **dominating** the second one because confidential is greater than unclassified, so it's not allowed to append information to a lower clearance of access class. It's violating. Cannot write down, cannot write up. Sorry, I want to make explicit what is obvious, it's not good to write up. Why not good? Because it implies to be able to read up, we don't want that. So if we are having the write permission we cannot have the write permission to higher level objects, it means reading up, and we cannot write at lower permission level, this is why the case of write is needed to be equal. If I want to provide information to higher levels I can operate in append mode, if I access some information in read mode no problem as long as I respect the other rule no read up of course.

So, the model is **fully described** by these 2 principles, you can find the axiom star property, remember simple security property means no read up, star property means no write down. Now if we want to carefully apply this framework it happens that you may need a problem, like the problem in this example here, imagine that a colonel is having such clearance, this is the security level, secret, nuclear army. The major, lower level, is having secret army clearance. So the access class of the colonel is dominating the access class of the major, very simple. Now how to face the problem raising when the colonel wants to **send a message** to the major? Can the colonel give an order to the major? But the colonel needs to write a message to the major, the major is having a lower level so this violates no write down. You get the problem? Very easy.

The need of not disclosing information to lower level is contrasting with the need to **give simple orders** to lower levels, this is very appropriate, being discussed in the military example of course. In order to solve this problem, we need to modify the concept of access class. When we say that there is a subject having an access class this access class is considered the **maximum possible access class**. So the subject is having the capability to change the current access class, of course I cannot change my access class to bring it higher, I cannot increment over the limit my access class. But I can go lower, I can **reduce my access class temporarily**, just for the time I need to do some operation. If we just modify the approach by saying the access class must be considered the maximum, and the current access class, the constraint is that the current access class is always dominated by the maximum access class, this is the requirement, to solve the problem the colonel can change the current access class, here you see the statement describing the issue that the colonel cannot write an order to the major, in order to fix this problem the colonel before sending a message to the major is decreasing the access class, making his access class lower just the same as the major. In that case the colonel is no longer accessing too sensitive documents and it can **provide information to the major** and can happen that the major will receive information having the same sensitivity as information the major can access.

This is the solution, the max level for every subject is **dominating the current level** in any case. This change of access class, current maximum is interesting only in the case of subjects. The access class of objects is not varying. This is implemented in many secure operating systems, in many secure DBMS, two limits of the model are dealing with confidentiality, not integrity, you may now there are extensions of the models introducing some primitives for **preserving data integrity.** The BL model is contrasting the trojan horses, because it includes a mechanism for controlling the flow of information. Just because the information that the colonel can write to some other subject by lowering the access class is information accessible to that access class, no mechanism for providing more confidential information. This is removing the problem of trojan horses but still vulnerable to **cover channels.** A channel of communication that is somewhat hidden and can be implemented for transmitting information to some arbitrary destination, a cover channel.

How to implement a cover channel even if the system is protected? The classic example is defined in environment where we are having concurrent transactions and we need to implement to use a two face lock mechanism, very basic mechanism I'm sure you know that mechanism. Consider there are **2 transactions**, t low and t high, meaning that a t low is having low access class, and t high is having high access class. Consider a data item dl at very low class, and assume such transaction is running. Suppose that t high requires a read lock on dl. **Th is requiring a read lock on dl.** The lock is granted, no other transaction running. Th gets the lock. Suppose the transaction tl wants to write on the same item, so it's required a write lock on dl. But the lock is hold by the other transaction, so tl is forced to wait. After the lock is released tl is able to make the write lock on the item.

What happens? The transaction th can somewhat change, imagine the transaction implemented by some software that has been programmed so that is implementing a cover channel, wants to communicate in some improper way high sensitive information to a lower level subject. A way to implement this communication is to make a modulation over the time of the lock, I may have some **technical timer** for reading the item. But I may decide not to release the lock immediately, to keep it for extra time and the amount of extra time is changing over the time. This means I have the possibility to have a modulation of information. Tl can learn from th some information that is deriving from this **variation in time**. So this is just a very simple system for implementing a cover channel, a channel of communication that is not explicit but it's actually providing a way to make subjects communicating out of the band. Removing the constraints posed by the system. And the cover channel in general they may be defined on timing schemes like this one or also defined over storage scheme. The storage scheme advices on the fact a transaction can make the system **change the basic state** describing the environment, this state can be read by another transaction and the meaning is giving the ability of inferring information described from the high level transaction, so again some higher level class can communicate information to some lower level class violating the no write down policy. Depending on the characteristics of hw/sw platform.

Opportunity to mention some fields of research I'm involved. I just want to mention the ability of **transmitting information in a covert way**. Simple example, imagine you take a picture of the colosseum, you have a good camera so the size is big, images are represented like pictures represented as bitmap, a matrix of pixel of coloured pixels, compressing such information or not. Not important, what happens if you just change the lower level bits? For small changes of the lower level bits the change of the image is not impacting the view of the viewer, for the viewer the original and changed having lower level bits changed is making no difference, you need a software to understand the change. If you just try a search you will find many tools allowing you to change a picture so that you can encode a text within the picture to change. How big the text?

You can **encode within a traditional picture** and cannot see any difference you make a comparison for any viewer human viewer, there is no evident difference between the 2 images, if you are checking analysing the picture by some software is not so obvious to understand the low level bits can be changed by some approach. This is just a simple example on pictures. You can have on videos, and several reasons in order to do that. In some cases you just want to **watermark** your file, watermarking is some information you are adding to your picture to be able to recognise the ownership of the image, and watermarking can be done in some hidden way or even by some no hidden way. In order to make you act in the domain of time or in domain of frequency. Ok, you can change the approach in several ways and get some interesting results, I don't want to go further, I give the flavour of the idea, on this topic I am following some master thesis, there is the space to work in this area. In particular I'm interested in the possibility of introducing some watermarking in digital images, so that the ownership and transactions ownership for the image are stored within the image. So that you can not hope to take the image and change delete the watermark. This is available.

Before I forget, I want to. Correcting a mistake with IPtables. I got an explicit question about the need of restarting the service, when I make a change in IPtables by adding a rule do I need to stop the server and restart? I answered yes. **I was lying**. Time passes so it was true long time ago, it changes. So I added an extra slide somewhere here maybe you have noticed it because it's available for download. This is the extra slide. When you are adding a new rule it immediately works. Keep in mind that when you are rebooting the system if you have added a new rule the new rule is lost, the configuration of the firewall is maintained in the kernel. A practical way to implement configuration on iptables is imagine that you work a lot for introducing rules for iptables, then you can give the command iptables save, this is saving description of the tables.

You can **redirect the output** to a file and you can insert in the boot sequence the possibility to read from the same file, so using the iptables restore command, if you want to preserve the rules introduced in your firewall just save the rules and insert into the boot sequence the iptables restore command, not the unique way but very common and useful, also I want to give a practical suggestion, very easy to make mistakes when defining rules for IPtables, it happens frequently that you are connecting to a system bu secure shell, add a rule, immediately holds and you are immediately disconnected because you made a mistake, if you disconnect you can no longer connect, this is very bad. Many administrators are implementing a practical solution. A **chron table** contains a list of command automatically implemented by the system every x minutes or days or seconds, whatever you want. In the table you may insert the command iptables restore, whatever is your mistake maybe you can ask your system to automatically restore. Every 10 minutes you reload the configuration, if you want to make a new one if it works you may save the configuration in some file. After some time the configuration is lost. This is a **very practical solution**, not very theoretical, this makes the life easier for many administrators. Net filter website, the organisation maintaining the iptables software and other softwares, this was what I want to add because I was lying about the need of restarting the service. questions? I finished.

Last lecture

Have you ever tried to search **cyber security on wikipedia?** You are redirected to computer security. Here computer and network security. Much more. Under the term title computer security there is any aspect of security, physical and logical security. We studied logical security, what about **physical security?** I'm not an expert, there are some easy comments, remarks we can do. For instance you will see with professor Marchetti in the case you follow web security and privacy the last rules about protection of sensitive data, in European Union. More demanding. Issued last year but they will be enforced here in Italy next year. One of the first aspect you derive is ok whatever data I keep it encrypted, because if I have data bridge I have a checklist, a to do list, I have to inform people, authorities, organisations and so on. But if data are encrypted the list is smaller, of course there is much to talk about, but why you have a data bridge? There are attackers, and they are smart. When a computer is **compromised** what happens?

The attackers gain a high power on the computer, maybe they can scale up privileges so they can become root of the computer, control whatever aspect of the computer, so ok I encrypt data, I control the computer, I find the key for decrypting data, so it's almost useless encrypting data. It's very important to understand what are the consequences of **encrypting data.** You know that you can buy disks mass storage, self encrypting data. What I mean? Ok, you use a disk such that every data written in the disk is encrypted, so that if somebody manages to enter the room and open the box of the computer, the case and extract the disk he will find encrypted data. Good. They are very good for physical protection, what if I'm just using standard disks, not encrypting data?

I use a file system and in this file system I have chosen to encrypt my data by using some process running in my computer, all modern operating systems allow you to encrypt your file system. What is the difference? Encrypted disk and encryption managed by the OS. There is a simple **difference in performance**, because encryption in the first case is made by hardware, a very strong firmware. What else in terms of security? In both cases if somebody manages to extract the disk from the case he will find encrypted data, actually there is a strong discussion about this topic, it's not so clear about the consequences, I mean to study the differences in terms of security, not performance, what happens typically is that the key in the case you are encrypting by the operating systems the decryption key is unlocked by the pass phrase when the user is log in and this is able to unblock the encryption decryption key, it's a symmetric.

The encryption is **depending on a password** in the second case, in the first case on a key, very small difference, I invite you to never be superficial about that, because it depends on who the operators are, you know a nightmare for many organisation is to provide an answer, what is the real power of the administrator of the network? Can the administrator of the network open sensitive documents? For every user? Whatever system even if it's encrypted needs a **decryption key**. So how to access this key? If the system can access the key maybe the network administrator can access the key. There is no clear answer, this is good for research, good for a master thesis. How to manage encryption of data limiting the power of the root of the network administrator to access keys and decrypt sensitive documents? There are several approaches but I would say there is **no final approach**.

You can also decide to encrypt only sensitive documents by a user decision, just encrypt, use some library, some tools for encrypting your sensitive documents, you can of course. What are the consequences? You can manage the privacy of your documents but there are several aspects that are unsolved, because what about logs? If logs are not encrypted any attacker can make a very deep analysis of what the user is doing. Especially in the case where the logging is made in a deep way. Last aspect of protection data is ok I have my data in a database, so I add to my database management system a module for encrypting the content. Can I? yes, of course. **Another level of encryption.** Can I combine different levels of encryption? Yes I can. With some motivation. These are topics very critical in these years, I invite all of you to be very sensitive and very deep about these topics, they will be the topics in the next 10 years I would say. Many organisations are knocking the doors of university and they ask hey you please ask you to protect data with some good tools. **They ask for tools,** and the answer is no tool, you have to make a security model you have to make the analysis of your risk, the risk is depending on your business and we cannot do that.

After that, you can design a policy and when you have the policy you can derive what tools you can use for enforcing the policy, so I would say that many medium small enterprises are not yet ready, this is very important for every modern computer engineer to be able to face such challenges. Are you after this course expert about encryption, cryptography? **No. You are not**. Not even myself. Because the real expert is not an engineer, they exist but they are mathematicians, it's all **mathematics** today. You need a very deep knowledge of discrete mathematics. So, you will be ready to check what type of encryption is offered, you will be ready to make choices, ready to understand a new approaches, for instance there are some approaches in cryptography not covered in this course:

- Elliptic curves

- Homomorphic crypto
- Quantum crypto

If you want to deepen you can start studying reading the topics. What about **elliptic curves**? Not so important but is a nice approach for introducing **another level of security** to afford keys of smaller length while maintaining the same security. Something you know, DH, and you will see ECDH. Just a variant when you are adding an extra ingredient, that is meaning you are using a polynomial that is introducing some more entropy in the process of setting up a key for encrypting and decrypting. So this is **standard**, this is good, this is I would say very employed, this is more interesting in my opinion.

The **homomorphic encryption** is very useful in case of think of data in the cloud, you have your data in the cloud storage, and you think of the case the user makes a classical action, please computer find me documents containing the word university. To the traditional process you have to take the encrypted document, decrypt the document, process for understand if it's containing the word, I'm not talking about the case there is an index, in many cases is no possible to manage such an index. This is **very demanding**, the homomorphic encryption is able to, this is the document, you want to check, want to grep the document searching for university, instead of decrypting documents you encrypt the word university and you search the encrypted document. This is **much faster**, a very complex topic modern very promising, there is a lot of research about that. For people interested i would say that is very important and very modern and current, master thesis about that? Not with me. Not my field, homomorphic encryption.

Quantum cryptography, what is that? I guess you know what quantum computers are. What are them? Do they exist? There is a theory developed about quantum computers, showing that they will change the rules because they will be able to solve NP complete problems in polynomial time. So current approaches for internet security will be broken by quantum computers. Do they exist? **There are several projects**. The projects want to be approved for concept, they want to show the approach can work and I remember 2 years ago I was reading an article announcing a big success, an organisation managed to build one of the first quantum computers. I have to redesign the course, I immediately looked for the results, what could they do with such a computer? I found the result. They were able to factorise number 27 in 2 seconds. It's a quantum computer doing that, you have to improve the technology for introducing the parameter 10^{-25}, in the running time. A lot of improvement in the technology, a theoretical time.

You know that the internet is full of **theory of conspirator** and so on, there are people stating that quantum computer exist, used from government agency, I don't know, I don't think so. But there is a **new type of cryptography**, researchers are trying to be prepared about the new type of computing, starting to design a new type of cryptography, robust about quantum computers. A totally new field, much to do. I would say that for the cryptography topics those 3 are the main limits of our course, whatever topic we have faced in the course could be deepened so if you think that one of the protocols like secure shell or TLS or IPsec described by 4-5 big documents every document containing 90 pages, to study all details of IPsec we need a course just on IPsec, but you should be ready to deepen whatever protocol if you need, because the course is providing the base for facing every type of classical cryptography.

Also we have understood we can use cryptography for facing not only the privacy, also other needs like **integrity, authentication**, like non repudiation, and we saw lots of techniques, many techniques, the course is full of small ingredients, if you check the list you will see a big one, just the page of describing the content of our lessons you will be many details. Many things for teaching purposes, we have seen bugged protocols for authentication and improvements, new bugs, until reaching the final version of classical protocols authentication, other protocols are not yet covered, we didn't. I suggested to do a homework on SRP, a very good protocol for doing authentication. Homeworks I would say that are useful not only because I use them to keep you in the classroom. It's important to interact, they are trying also to open windows on some possible deepening about different types of subjects.

Today I would be proposing the **last homework** on IPtables, you can provide your contribution. Not only cryptography, in the course we have faced more general concepts related to information security of course, if you remember at the beginning we defined the frameworks, the needs in terms of information security requirements, then another limits is that we cannot face topics at application level, lots of new topics so our analysis will be developed until the TCP level, until reaching TLS. Again we can see details about the logical security, at application level in the course web security and privacy, an important topic is privacy, the needs of privacy, the security of the browser and the security of email. Email is very very insecure, you know how to spoof emails? Have you ever spoofed an email?

We also tried to give some **basic knowledge about systems**, protection to be setup for systems like firewalling, not so much, we have given basic ingredients, we have seen IPtables and ok, **firewalling** is providing an orthogonal protection, not depending on cryptography, different type of protection, it's useful, we approached the topic in a very way, in addition to firewalls if you want to really protect your local network you will need other tools like IDS and IPS and hybrid systems doing both operations. What is an IDS? And what an IPS? They are services offered by devices connected to the network, making analysis of the traffic in the LAN. They collect data about the traffic and they match at any time data about the traffic against patterns because you can define dangerous patterns, what is a dangerous pattern?

You may write rules saying ok if you see 2 computer are sending packets to same destination, same time, you can define dangerous patterns, are coming from knowledge basis, describing **patterns of attacks**, and other power provided by these systems is that they can learn, some of them, after some time they learn what are the regular patterns of your traffic, able to detect any anomaly in the pattern of the traffic, so a good usage of a tools like that is allowing to quickly detect an intrusion. Not necessarily a data bridge, it's corresponding to a scheme of network traffic you don't want, violating your policies. It means whatever is the attacker outsider, insider, traffic corresponding to violation of the policies.

Such type of traffic can be prevented by prevention systems by different approaches and this is I would say it's very in theory very powerful as approach, they can be coupled to application level firewalls, so that they can be able to authorise or deny traffic that is violating some typical pattern, I mean think of an application level firewall, if we are talking about http protocol, it's able to make decisions about http connections, this is application level, what if you have an attacker where you have 2 types of connection, http and ftp connection coupled with an ssh connection? 3 protocols, they should be running at the same time on same host. Normally not detected by the application level firewall, you need a more powerful tool, checking packets at application level for authorising some operations. This is while the IDS is just a device that is in the network, analysing the traffic of the network by using some agents, the IPS is changing the topology of the network because it's becoming an object requiring packets to pass through. This is one of the approaches, not the unique one.

We are **not covering these topics**, other courses are talking about that, I expect that you will be familiar with such tools. At least this one, IDS. We have shown other limits of our course. Again back to **cyber security**, what is the definition? I told you that wikipedia is redirecting to computer security, there is an original definition of cyber security. The original definition was aiming to capture the idea that there are physical systems for instance power utilities, doing **physical things**. Providing electricity to your city maybe. Until 40 years ago when you need to reconfigure the distribution of the power organisation where needing to send a group of people to some places to operate in the local management system and to do things. Then technologies allowing to connect from remote to such distribution centrals have been introduced and one of these technologies is the SCADA technology. it's a technology allowing to reconfigure physical systems by using remote connections running over the internet.

Of course if you expose interface to the internet for reconfiguring you will be exposed to attacks. It's a very important topic and cyber security was introduced for describing the needs of **security against attacks** coming from the network aiming to attack physical systems. You may think also about medical systems. And many many more, other systems. This is the original meaning of cyber security, after that a lot of chaos around the topic so today all researchers are using cyber security for computer security, whatever aspect of security, where a computer is somewhat involved. Questions about that? Let me I have already told you some possible topics of master thesis, one I told you today, even in some previous meetings i told you something. Basically my master thesis are related to cyber security, very wide topic, steganography and I have already mentioned this and watermarking. Modern watermarking is using modern cryptography in order to introduce secure information hidden or shown, depends on what you want in your documents. privacy. When you install your OS, in the case your are using Microsoft or apple, what about your privacy?

Do you imagine what type of information is running from your computer to the database of apple or Microsoft? Lot of information. There is a **normative aspect** because when you are the first time switching your computer on so the self installation process is activated you have to accept several disclaimers do you read? In many cases they are a big amount of pages. I think that apple is having the best size documents of 60 pages. When you are accepting all the statements you are accepting that what applications you are using, all information transmitted, after that many organisations like **EFF, electronic frontier foundation**. Write the name. Non profit organisation, based in San Fransisco, and working for human rights in the internet. They are the one providing the well known tool https everywhere. Also OS, there are some commercial linux distribution, the type of traffic after the user has decided to remove the authorisation to provide data usage, you are asked to use, if you deny the permission what data is still provided? I like to analyse such type of information and i'm concerned about what happens in mobile devices, what information you provide to google about android?

I want to know, every detail, in the case of mobile systems the tools provided to the user for controlling the privacy they are less developed, so you have less control, still today. So there are possible thesis in this framework. Other classical thesis, http proxy for secure shell. Authentication password based, managed with the approach of secret shelling. Use of cryptography for detecting **software plagiarism**. I just want to mention a small detail, I'm interested in 2 areas, cyber security and algorithms. It means that i'm also offering master thesis for special algorithms, doing analysis, implementations, solving new problems with smart algorithms, just mentioning that, I don't want to open a discussion about algorithms, this is the CNS course. I'm also open to proposal coming from students of course. More than 10 per year it’s too demanding for me, I don’t like to say students just go ahead. I want to interact with students, I cannot follow too many thesis. How many of you consider cryptography as a necessary tool for security but not looking good? This is stupid question. questions? Last homework, then details about the exam.

General information. I noticed that some of distributed system are too close. Anybody doing them in the same session? First important communication, the faculty is in panic mode for the need of rooms, a big emergency. It’s possible that dates of exams are just moved, this is possible, you cannot prevent them, not depending on anybody. Actually ok not in my power, of our administrative stuff to fix huge problems, anyway the dates are already published, and I am not opening registration on infostud, why? Because there is a practical reason, since people have to take 2 different exams, most people they want to take one exam and take the other in some other session, if I just open the registration on infostud if people just take one exam what I should write as an exam? I cannot write half credits, I should write a negative result, not good, not fair. The registration of the exam is done by a google form, published in the web page of the course. So you will be compiling your form, providing your data and the form is also giving the possibility to give emphasis to possible issues deadlines, problems.

This is **another problem**, I’m not fast in analysing your contributions, I have too many exams and for every subject I’m doing the exam I have a subset of cases they are urgent, I have to focus on them, the non urgent are just postponed, it can takes even 3 weeks, depends on the number of people taking the exam. please, if you have issues, you need a quick correction write it in the google form. I will provide you the correction even if I have to do that in nighttime, no problem. If not just be patient. After you take the exam I know all of you pass the exam with a good rate, and then I just take notice of the outcome of the exam and the outcome will be coupled with the outcome of the other exam, we will be making the average between the 2 grades and will be registering the result. The process is again slow, since there is a lot of manual handling I can do mistakes, so I have to check multiple times, cross checks, the process is slow. If you need a quick registration just ask, I will do that, but it’s very demanding doing one registration for special purposes. I agree with the other professor to maintain the outcome of the exam for a long time, I’m not expected that people will take one of the exams and the other after 1 years, if just happens there is no problem, I will guarantee the memory about your result for at least 2 years. What after 2 years? I don’t know.

The outcomes are in the cloud, I don’t know. You have already seen more times examples of how the exam is presented, you will see a piece of paper, many questions, you have to write your answers and you have also to remind me how many homeworks you have sent. Please remind me, scripts are not really reliable, I know that there are some errors, so please I need an extra check, remind me how many homeworks you provided. The exam is written, I'm not doing an oral part, the outcome of the exam is coming from my analysis of your document. I may ask you to come to my office to clarify some aspects. Maybe I can do s**ome further questions**, if it's my decision. The way I provide my scores is very analytic, I assume you are perfect, for every error or omission I see I remove a small part of the maximum. I see what I get. Then I do that for many, then I check if I was consistent in my operations, and then I make a statistical analysis of what I got, if I see some anomaly I make some change for ensuring that what I did is fully consistent, and then I will be publishing the result in the web. Sometimes I get emails, professor when you publish the result? My typical answer is I don't know. It means I have finished checking your contributions. After some time I stopped replying to those emails, but if some need is emerging please write me and tell me I need that, no problem. Your success is my success. Your failure is my failure. The final score of the exam is independent on the context. My last exam. Don't ask me to take the exam in the extra calls. My office hours, they are not clearly stated, I typically put my office hours on Tuesday just after lunch, I have to move and just you will need to reserve a slot by using the web. You go to the web, choose your slot, if you see no slot it means they are full, but if you want to come to visit me and you find no slot, check again, especially the day before, there are many people reserving slots and the last day, they are releasing the lock. Very bad. I'm taking note of this. I set up a calendar on google. Every time a student locks and unlocks a slot I get a text message on my phone. **Just be fair.** I have some typical averages scores and in the first session the average is 28. With many maximum scores, many. The number of good scores is lower in the session in summer, it's very bad in September. I suggest you to just decide to plan when you want to take the exam and then make a fair plan and then you will see I will be not asking to solve strange problems, I can propose a protocol that is clearly not respecting the information security requirements, may be asking what is bad in this protocol, fix the protocol, I may be asking to you to reason about hashing functions, for instance, birthday attack, asking to you describe make examples of multi session attacks, birthday attacks, man in the middle, meet in the middle, all types of our attacks we have considered. Nothing not considered. I may be describing a framework and asking you to make an analysis on the security requirements. The exam takes **2 hours** and you will find some examples of past exams in the website. Actually I'm opening the registration on infostud for old cases, there are people belonging to old settings of the master and they have the exam of 5 credits, in Italian because long time ago we had 2 courses. For those people I will be opening the registration on infostud. For old people, the program to be for the exam is what we have done in the exam in this year, what is in the log in the webpage.