POLITECNICO
DI TORINO

Recap Computer System Security (02KRQOV)

Jacopo Nasi
Computer Engineer
Politecnico di Torino

I Period - 2018/2019

December 2, 2018

Contents

License

Acknowledgments

Questo breve riepilogo non ha alcuno scopo se non quello di agevolare lo studio di me stesso, se vi fosse di aiuto siete liberi di usarlo.
Le fonti su cui mi sono basato sono quelle relative al corso offerto (**Computer System Security (02KRQOV)**) dal Politecnico di Torino durante l'anno accademico 2018/2019.
Non mi assumo nessuna responsabilità in merito ad errori o qualsiasi altra cosa. Fatene buon uso!

1 Introduction Security ICT System

Why is securoty an important issue? Nowadays that everything is online and connected to a world wild network, the security over the ICT system has become fundamental. A lack of the secutiry could generate lose for milions of money. Also data breach become a problem.
Everydays technology improve and drive innovation but security must be improved together with the innovations.
With the increase of the number of connected devices, the IoT (Internet of Things), security start to facing a lot of more problem, the complexity of the scenario has become really really big. From personal devices, like desktop, laptop, fridge or car, by communications networks, and to distributed services, everything must be secured!

Complexity enemy of security based on one of the first axiom of engineering: *"The more complex a syste, is, the more difficult its correctness verification will be."*. Keep a system as simple as possible is always a good idea. The KISS rules (***Keep It Simple, Stupid***) is one of the most important rule over the system security.

Definition of ICT Security "It is the set of products, services, organization rules and individual behaviours that protect the ICT system of a company.

It has the duty to protect the resources from undesired access, guarantee the privacy of information, ensure the service operation and availability in case of unpredictable events (C.I.A. = Confidentiality, Integrity, Availability).

The objective is to guard the information with the same professionalism and attention as for the jewels and deposit certificates stored in a bank caveau.

The ICT system is the safe of our most valuable information; ICT security is the equivalent of the locks, combinations and keys required to protect it."
— **Italian Bank**

An important part of the security study is the Risk Estimation, is a fundamental step that take in account all the assets and events to evaluate the risk of something. The flow is showed in figure 1:
Where the assets is composed by everythings needed by a service to work, both soft and hard part, also human resources. The vulnerabilities, intrinsic of an asset, represent the weakness of it. The threats is a deliberate action, or an accidental event, that can produce the loss of a security properties exploting a vulnerability. The event is also characterized by an impact and a probability

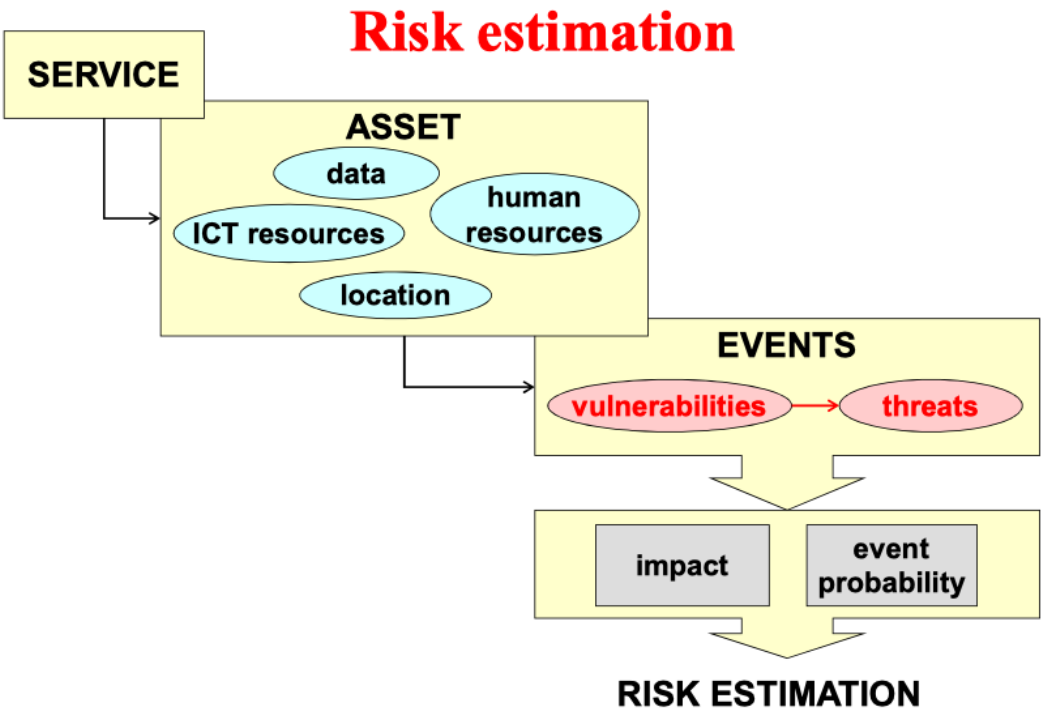

Figure 1: Risk Estimation

that could be high, low or other middle values.
Direct following of the risk estimation is the Analysis and management of security. After the evaluation of risks, is necessary to:

1. Select Countermeasures
2. Implement Countermeasures
3. Audit (check if works)

The security implement is not a phase of the developmente process, is part of each sigle part of it. Security can't be compute at the end of the development, it must be implement from the beginning of the process. **Security is a process, not a product!** The following figure 2 show the parallel line followed by the security development.

An important definition, before speaking about security itself, is the **Window of Exposure** the represent the time when an attack could be perfomed and there are no countermeasures to avoid it. This window could potentially be infinite and this is the real problem. The figure 3 show how this windows id divided in different part:

As already says, security is not a product but is a proccess. Computer flaws are inevitable and this is way we can't use devote our security to only secured products. The only way to effectively do business is an insecure world is to put processes in place that recognize the inheritent insecurity in the products. **The**

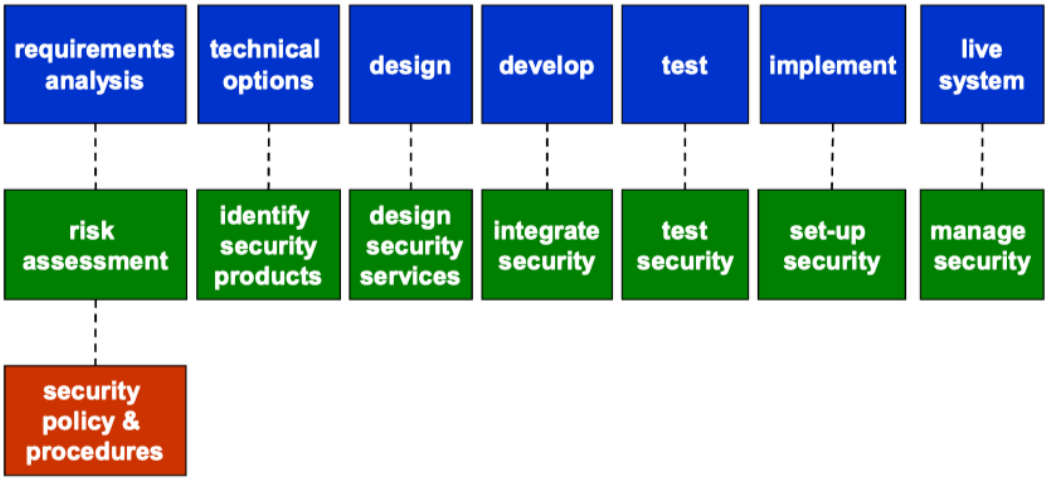

Figure 2: Security Life Cycle

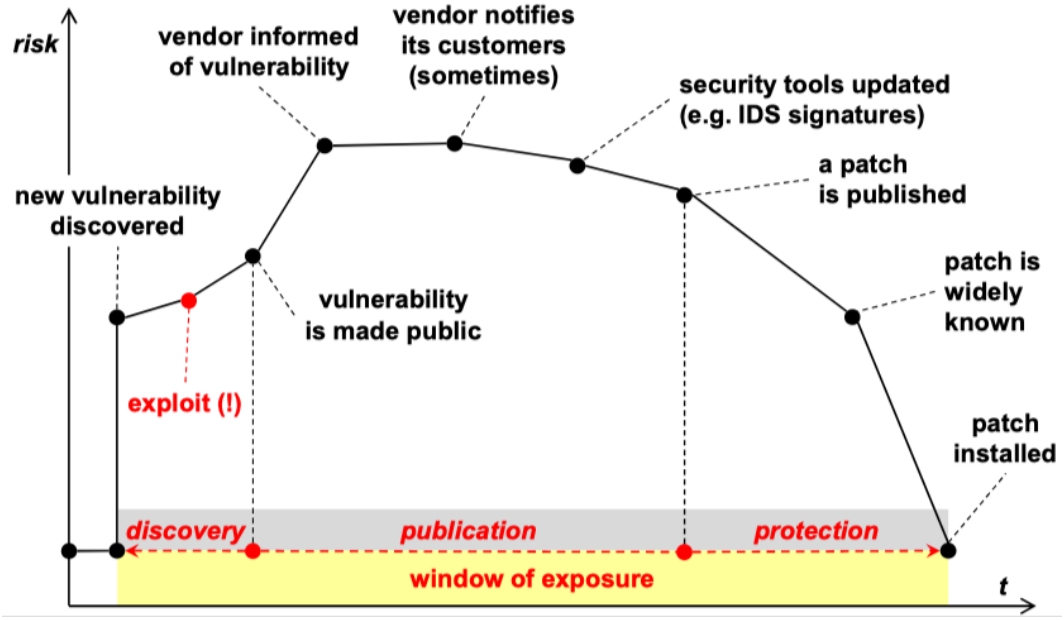

Figure 3: Window of Exposure

trick is to reduce your risk of exposure regardless of the products or patches.

Security Principles Here some of the most important security principles:

- Security by Design
- Least Privilege: Only correct rights and the only few needed
- Need-to-know: Access to the only piece of stuff that are needed

- Security by Default
- Security in Depth: More the importance of the system, more the obstacles

Security Properties The following list show the most importante properties of the security world:

- Authentication (Simple/Mutual): Source (or both) must prove themself
- Data Origin/Authentication
- Authorization and Access Control
- Confidentiality/Privacy/Secrecy
- Non Repudiation: Formal proof, acceptable by a court of justice, that gives undeniable evidence of the data creator
- Availability
- Accountability
- Integrity: Modification, Filtering

Where is the enemy? The enemy normally is supposed to be outside of our system but is not simple as it seems. The possible locations are:

- Outside our organization (Firewall)
- Outside our organization, with exceptionn of our parters (VPN)
- Inside our organitazion
- Everywhere!

The last item is probably the more true. The distinction between internal/external and good/bad guys is no more sufficent. From the *Verizon Data Breach Invetigation Report* the percentage of source of attacks is: 20% Internal and 80% External, probably the internal percentage is a little bit higher due to the fact that Verizon is a provider and can't show the internal side so much.

Basic Problems There are some basic problem for security:

- Networks are insecure:
  - Clear communications
  - LAN use broadcast
  - Not E2E geographical connections

- Weak user Authentication
- No server Authentication
- **Software with bugs!**

Classes of Attacks Some of the most common type of attacks:

- IP Spoofing / Shadow Server
- Packet Sniffing
- Connection Hijacking / Data Spoofing
- Denial-of-Service (Distributed DoS)

The **IP Spoofig**, or source address spoofing, is forginng the sourcce network address, typically performed at LV3 (IP), but also at LV2 can be performed. The typical attacks are: Data Forging and unauthorized access to systems.
The **Packet Spoofing** it reads the packet addresses to another network node, it easy to do in LAN or at the switching nodes. It allows to intercept password, data and other stuffs.
The **Denial-of-Service (DoS)** it keeps a host busy so that it can't provide its services. There are a lots of examples: mail/log saturation, ping flooding, SYN attacks. The main problem of this kind of attacks is that there are no countermeasures. The **Distributed DoS** are similar to the previous one but perfomed by a greater number of hosts (botnet), controlled by a master. The power is the same of a normal DoS multiplied by the number of deamons of the botnet. There are also some techniques to improve the attack like using a reflector to hide the attacker's track. One of the more important DDoS was performed agains Yahoo! during Feb 2000.
The **Shadow Server** is a technique that host that manage the attacks show itself (to victims) as a service provider without having the right to do so. It provide a "wrong" serice to victims, like bank sites or other stuffs.
Connection Hijacking / MITM, AKA Data Spoofing, is performed when attacker takes control of a communicatio channel to insert, delete, or manipulate traffics. It can edit data is all forms, and change the messages of the communications. Another similar for is the **Trojan / MITB**, it used the fact that network channels are more protected, but users terminals not. The behaviour is to install a keylogger to store everything. It could be also passed via browser extension.
The are other application-leve problems:

- Buffer Overflow
- Cookies
- Clear password in DB

- "invent" a protection system

We make now some clearance about name of malwares:

- Virus: Damage the target and replicate itself, propagated by humans (require complicity)
- Worm: Damage target because replications (resource saturation)
- Trojan: Malware vector
- Backdoor: Unauthorized access point
- Rootkit: Privileged access tools, hidden and stealth
- Ransomware: Make hosts unreadable (can be also silent)

Non technological problems Prorably the greatest majority of the problems and leaks come from here! The are some basic problems: due to low awareness, mistakes, tendency to trust and other facts the major cause of leaks are humans.
The **Social Engineering** are sets of techniques used to asks the involutary user's partecipatio to the attack action, usually the naïve users are targeted (e.g. *"do change immediately your password with the following one, because your PC is under attack"*), also experienced users are targeted (e.g. by copying an authentic mail but changing its attachment or URL).
One of the most used technique is the **Phishing** (fishing) where the attacker try to stole information (fishing) from the target (fish), it can be achieved, for example, by showing acquaintance with the company's procedures, habits and personnel helps in gaining trust and make the target lower his defences. Is often performed by using fake mail, SMS or IM. The normal procedures works by attarcting the fish in a shadow server where it will leave of the sensitive informations or persuade to install plugins or other stuffs. Two variants exist, **spear phishing** (include several personal data to disguise the fake message as a good one, e.g. mail address, name of Dept/Office, phone no.) or **whailing** (targeted to VIP such as CEO or CIO). The **Pharming** is a set of several techniques to re-direct a user towards a shadow server, chaging the hosts file, nameserver, poisoning cache of DNS. Some important examples of this techniques are T.J.Maxx attack, Transformed3 phishing or Stuxnet.
The typical path of attackers is showed in figure 4.

From this brief introduction we can define he three pillars of security:

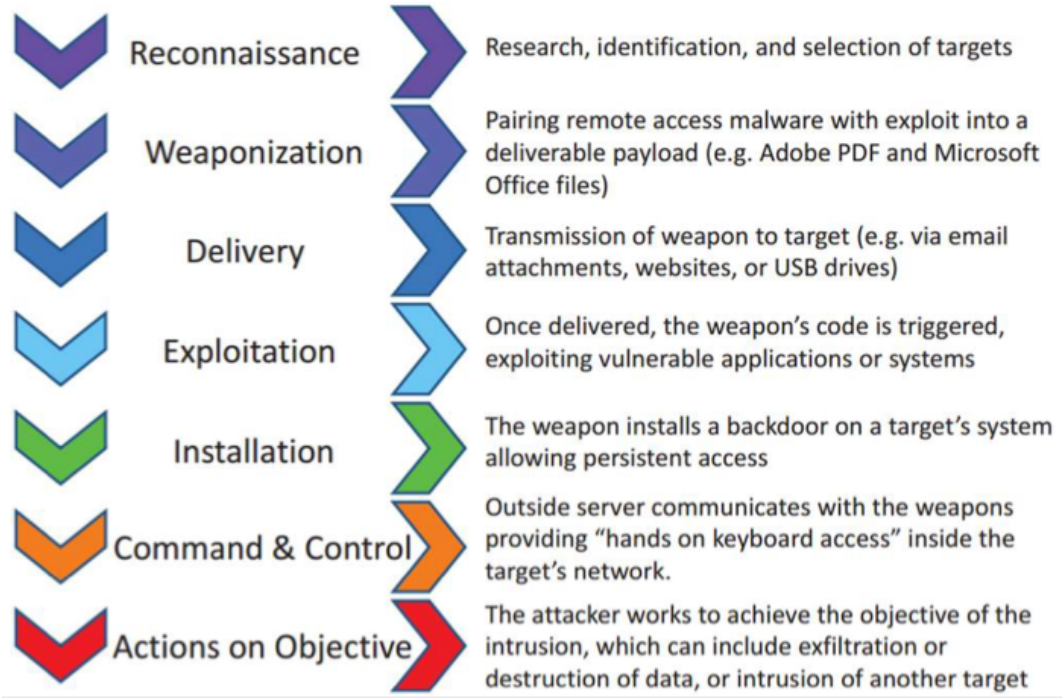

Figure 4: Cyber (intrusion) Kill Chain

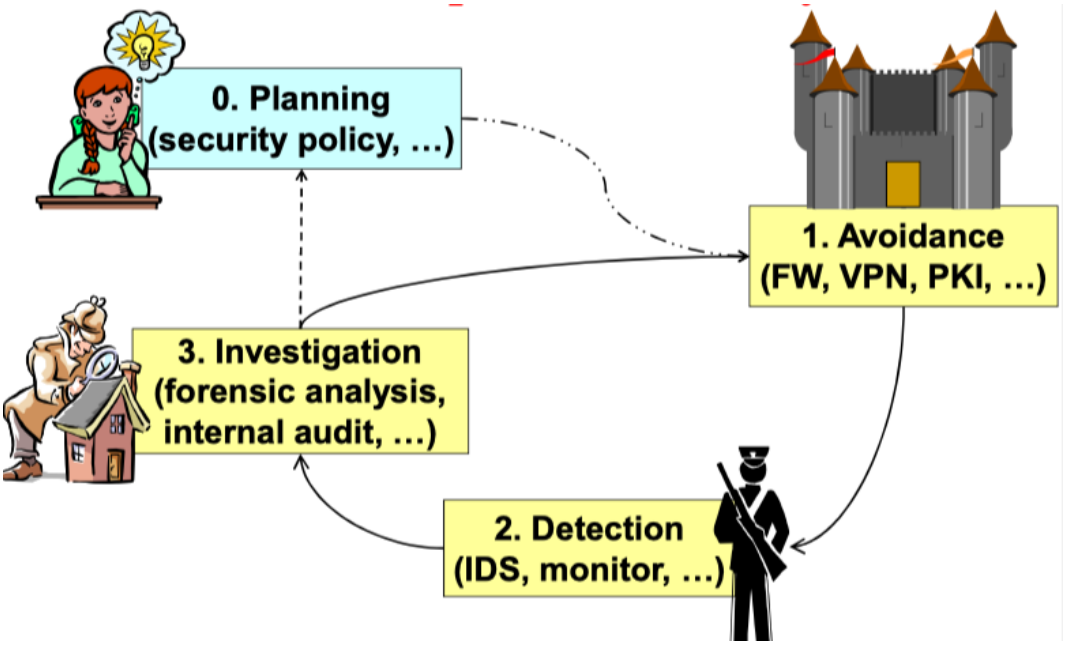

Figure 5: Pillars of Security

and we can define the main figures:

- Hacker: Good and skilled
- Cracker: Bad but skilled

- Script Kiddie: Bad but NOT skilled
- Wannabe Lamer: Not good at all

2 Basic of ICT security

2.1 Cryptography Introduction

The basic principles of Cryptography are showed in the following figure

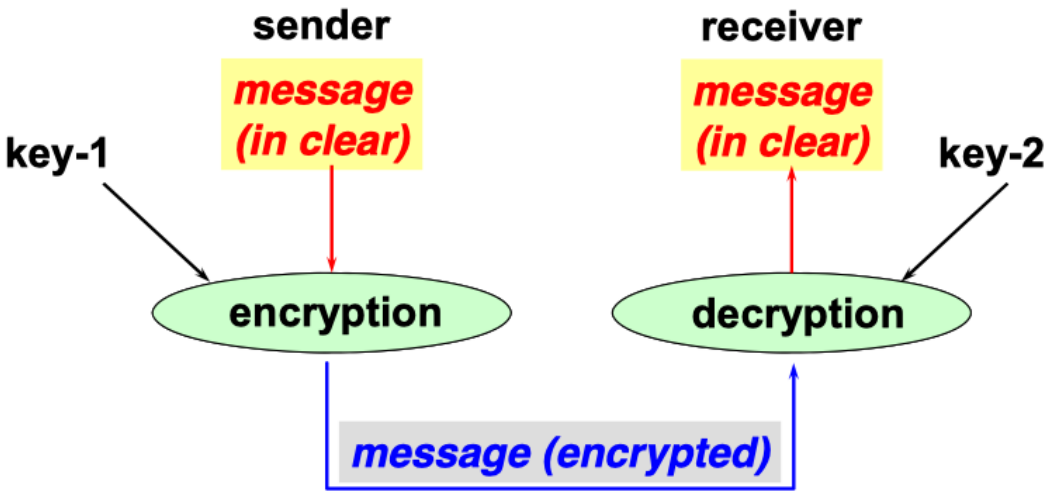

Figure 6: Cryptography flow

Some important terminology. The message in clear is called:

- Cleartext or plaintext
- Refer with **P**

Instead the encrypted message:

- Ciphertext
- Refer with **C**

Some principles of Cryptography, written by Kerchoffs, are: If the keys:

- Are kept secret
- Are managed only by trusted systems
- Are of adeguate lenght

then...

- it has no importance that the ecryption and decryption algorithms are kept secret
- on the contrary it is better to make the algorithms public so that they can be widely analysed and their possible weaknesses identified

In computer system STO (Security Through Obscurity) is not good.
An important operator of this world is the XOR function that is the ideal confusion operator, because it not change the probability. Is also a primitive operation present in all CPUs.

2.2 Symmetric Cryptography

Are all the algorithms based on a secret key shared between sender and receiver, used for encrypt and decrypt. Figure 7 show the general flow of this kind of algorithms. The advantage are low computational cost, in fact is used for data encryption.

- $C = enc(K, P)$
- $P = dec(K, C) = enc^{-1}(K, C)$

One of the main problem is *"How to share (securely) the secret key among sender and receiver?"*. T

Figure 7: Symmetric General flow

There is not a confirmation of correct decryption, is up to the user to understand if the result is correct or not, in any case a result is provided with any key.

DES (Data Encryption Standard) one of the first used by US government. Is now obsolete, is based on a key of 64 bits, composed by:

- Key: 56 bits
- Parity: 8 bits

This means that the actual bit of resistence is 56. It's based on 64 bits of data blocks. It is also designed to be efficent in hardware with poor performance, the flow is:

1. XOR
2. Shift
3. Permutation (not so good performance)

3DES is the triple repeted application of DES, it used two of three 56 bits keys. Is different from passe to a keys long 56*3 because the key lenght will not match with that number of bits. It can be computed in two ways:

- 2 Keys: $C = enc(K_1, dec(K_2, enc(K_1, P)))$
- 3 Keys: $C = enc(K_3, dec(K_2, enc(K_1, P)))$

The central step of decryption is made to generated mess.
An important issue of the encryption is the doubling of an algorithms. Double application of encryption algorithms is subject to a know-plaintext attack named **meet-in-the-middle** which allows to decrypt data with at most 2^N+1 attempts (if the keys are N-bits long). Thus the double version is never used, it double the computation time and increase they key lenght of only one bit. The formulas are:

$$C = enc(K2, enc(K1, P))$$

$$dec(C, K2) = dec(K2, enc(K2, enc(K1, P)))$$

$$dec(C, K2) = enc(K1, P)$$

The attacker can compute ENC(K1, P) for all values of K1 and DEC(K2, C) for all possible values of K2, and add only 1 bit of strenght for 2 keys of the same size.

IDEA International Data Encryption Algorithm it was patented but with low royalty, developed for 16 bits architectures. It use a 128 bits key and 64 bits of data block, it's famous because is used in PGP. The operation used are:

- XOR
- Addition Modulo 16
- Multiplication Modulo $2^{16} + 1$

RC2, RC4 other algorithms developed by Ron Rivest (Ron's Code), they are algorithm proprietary of RSA but not patented, 3 to 10 times faster than DES. RC2 is a block algorithm, RC4 is a stream one. They are using a variable length key.

Application of block algorithms applying these algorithm over data of size different from the block size require a little effort. When the data size is greater that the algorithm's block size:

- ECB (Electronic Code Block)
- CBC (Cipher Block Chaining)

In the other case:

- Padding
- CFB (Chiper FeedBack), OFB (Output FeedBack)
- CTR (Counter Mode)

The frist solution **ECB** is a bad idea, it based on the division of the data in chunks with the same size of a block, by encrypting them with a K (key), the formula is: $C_i = enc(K, P_i)$. The problems of this solution are:

- Swapping of two blocks of cipher goes undetected
- Identical blocks generated identical cipher texts hence it is vulnerable to *know-plaintext* attacks. (Word example)

These are the main reason to avoid the use of this solution for big data. The decryption is made by reversing the process. The figure 8 show the process.

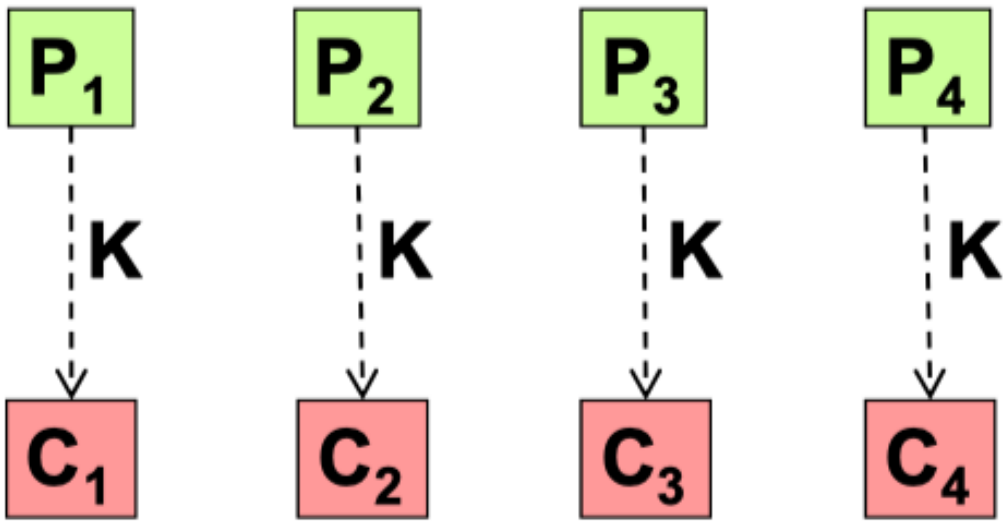

Figure 8: ECB

The solution of **CBC** [figure 9] is more secure, by using an IV (Initialization Vector), it encrypt the block by XORing it with the previous block and with

the key, this avoid the problem of the *know-plainntext*, because same text in different position will generate different encrypted text. The IV is added for increasing the difficulties of decryption. The formula is the following:

$$C_i = enc(K, P_i \oplus C_{i-1})$$

The decryption require the Initialization Vector (C_0), the formula is the revert the process by the following formula:

$$P_i = dec(K, C_i) \oplus C_{i-1}$$

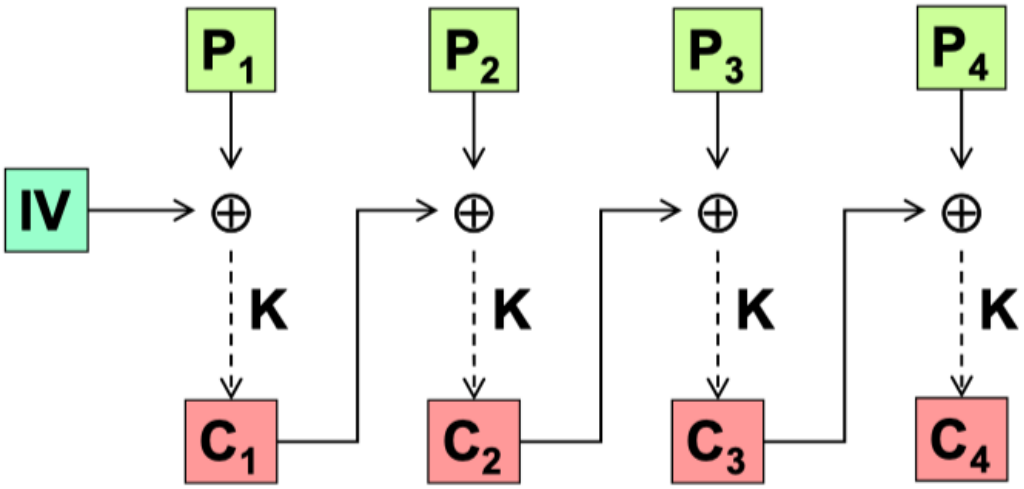

Figure 9: CBC Encryption

When the data is shorted than the block size, a **padding** (or aligning, filling) is required. It consist by adding some data at the end to completely fill the block. The are several techniques used for padding:

- If the length is know: 0x00 bytes
- Original DES: 1 bit followed by 0
- One byte 128 (0x80) followed by null 0x00
- Last byte value equal to length of padding

The are also some techniques with explicit length for padding:

- (SSL/TLS) bytes with value L
- (SSH2) random bytes
- (IPsec/ESP) progressive numbers

Padding is typically applied to large data, on the last fragment resulting from the division in blocks (ECB or CBC), for $|D| < |B|$ we prefer ad hoc techniques like CFB, OFB or CTR. Another important note is that, even if the plaintext is an exact multiple of the block, padding must be added anyhow to avoid errors in the interpretation of the last block.

CTS (CipherText Stealing) permits to use block algorithms without padding:

- Last (partial) block filled with bytes from the second-to-last block
- These bytes are removed from the second-to-last block (which become partial)
- After encryption, exchange the position of the last and second-to-last blocks.

The figure shows better the solution algorithm:

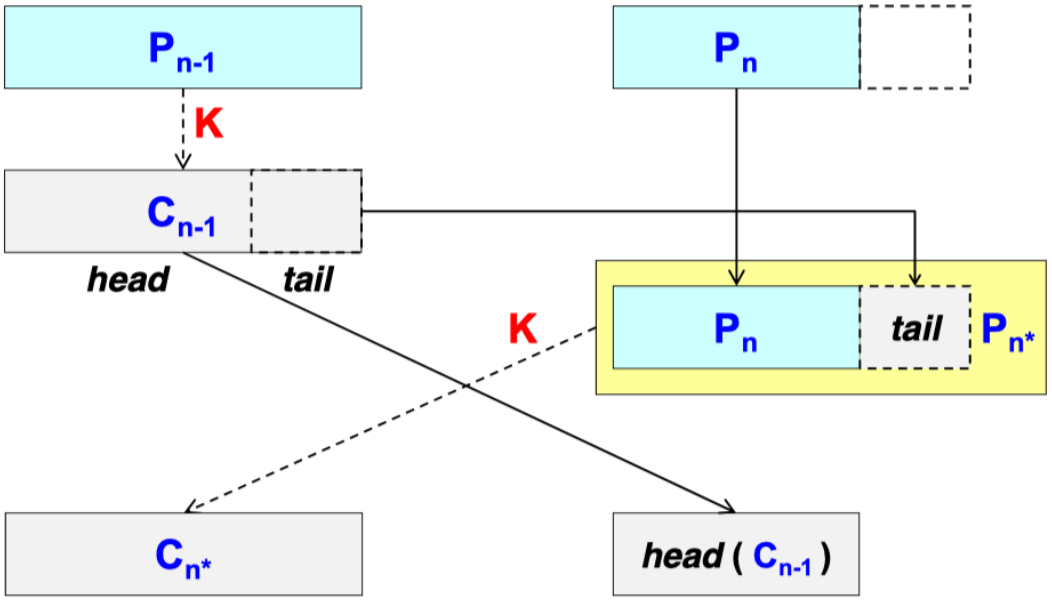

Figure 10: CTS with ECB Encryption

This technique is useful when we cannot increase the size of the data after the encryption. The tail, store, after the encryption in the second-to-last block is encrypted 2 times and it requires 2 decryption.
The **CRT** (Counter mode), really used, uses a block algorithm to cipher N bits at a time. It require:

- **NONCE** = Number used ONCE
- counter

The flow is showed in the following figure 11:

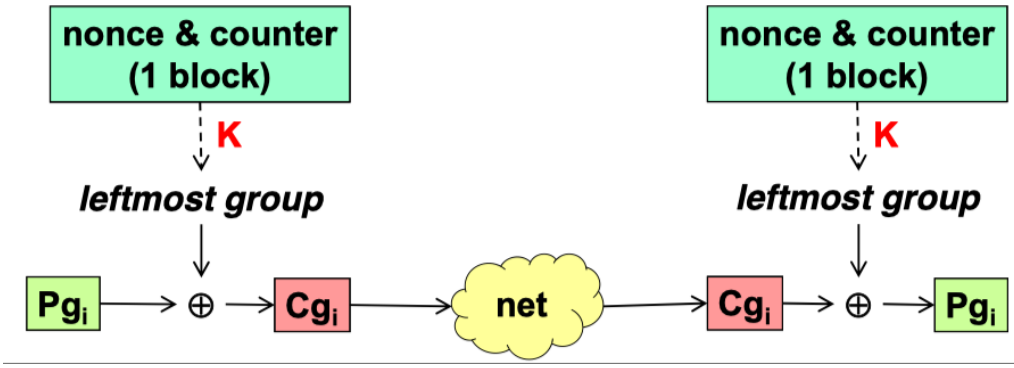

Figure 11: CRT flow

Symmetric Stream Algorithms this kind of algorithms not require to separate datas in blocks. They operate over one bit/byte at the time. The ideal algorithms require a key which is as long as the message to protect, of course this is not possible. The real algorithms use pseudo-random key generators, synchronized between the sender and receiver, some examples are RC4 and SEAL. The idea is show in figure

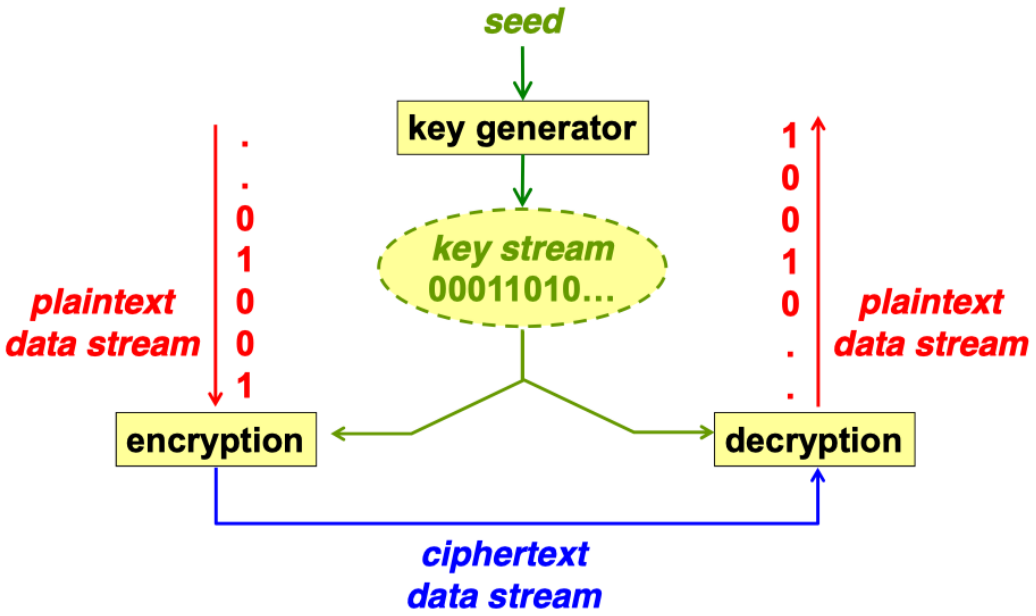

Figure 12: Algorithms of type stream

A lot of challenges were started for stream ciphers, some algorithms are:

- HC-128, Salsa, SOSEMANUK
- Grain v1, MICKEY 2.0

- Salsa20 and ChaCha20
- Camellia
- SEED and ARIA

Problem of symmetric cryptography one of the main problem of all this technique is that one key is required for each couple / group of user in order to guarantee privacy within groups. This means that, for a complete private communication between N parties, $N * (N-1)/2$ keys are necessary, these require key exchange algorithms and generate a problem for large groups.
Another know problem is the **length of secret keys**, if:

- The encryption algorithm was well designed
- the keys - N bit length - are kept secret

... then the only possible attack is the brute force (exhaustive) attack which requires a number of trials equal to:

$$T = 2^{Nbit}$$

This means that and increase of the computational power, reduce the strenght of the keys, that require a growning number over years. For this reason a challenges (AES) was promoted to develop another algorithm, the winner was **Rijndael**.

2.3 Asymmetric Cryptography

The Asymmetric Cryptography is based on 2 different keys generated in pairs, the two keys are different, one is called *public* (**Kpub**) and the other *private* (**Kpri**). If one key is used for encryption then the other one must be used for decryption, they have an inverse functionality. The computation load is high, due to this fact asymmetric encryption is used only for data streams and should not be used for data storage. The figure 13 show the process. The principals algorithms are:

- Diffie-Hellman, RSA, DSA, El Gamal, etc...

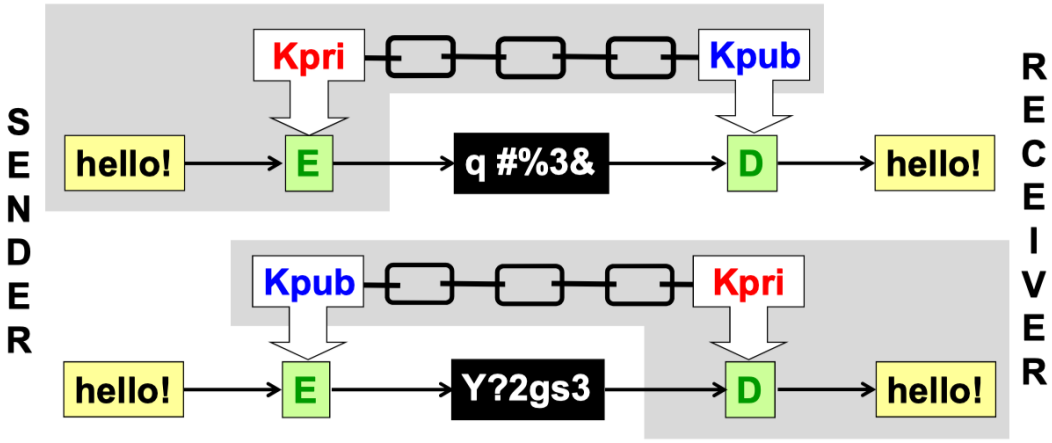

Figure 13: Asymmetric Cryptography with pairs of keys

Another possible application os this kind of cryptography is the **Digital Signature**, used to provide data authentication and integrity. Usually not all the data is encrypted but only its summary (***digest***).
Another important feature is the possibility to make secret message for a particular receiver given only its public key.

2.3.1 RSA

This algoritmh is based on a public module:

$$N = P * Q$$

known to anybody P and Q are ***prime number, large and secret***. Another value, $PHI = (P-1)(Q-1)$, along a public exponent E such that arbitrarily $1 < E < PHI$ and it is relative prime with respect to PHI.
From these element, a private exponent is computed:

$$D = E^{-1} mod(PHI)$$

from this last value the 2 keys can be computed:

- public key = (N, E)
- private key = (N, D)

After that computation P and Q must be deleted, discarded, killed!!!
Otherwise the process can be inverted.
An important note about RSA is that it can cipher/decipher only data whose value is less than the value of the module N, similar to a block algorithm but with a variable lenght. The final result is computed:

- encrypt: $c = p^E mod(N)$

- decrypt: $c = p^E mod(N)$

For matematichal reasons the roles of the 2 exponents, E and D, are interchangeable:

$$(x^D)^E mod(N) == (x^E)^D mod(N)$$

The advantage of using modular arithmetic is that the inverse of a reminder of a module could be "any number", there are infinite possibilities. An example in figure 14:

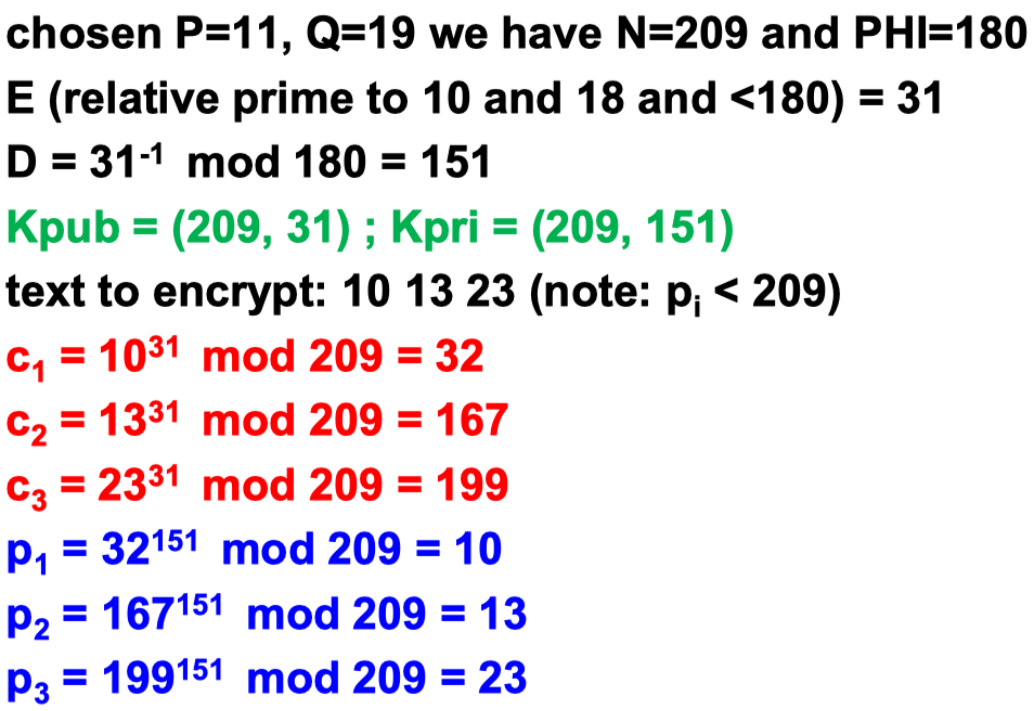

Figure 14: RSA an example

During years some optimizations were developed, like the one based on the CRT (Chinese Remainder Theorem) that makes the procedure 4x times faster.

Weaknesses There are some know weaknesses:

- Small encryption exponent
- Same keys for encryption and signing
- Acting as an oracle

Nowadays the lenght of the keys is important:

- 512 bits keys $\sim$ Some weeks

- 1024 bits keys $\sim$ Some Months
- **2048 bits keys $\sim$ Several Years**

During Eurocrypt of 1999 a devices for decrypting RSA faster was annouced, it has never been presented, probably for interest reasons. For that, some governments, no longer want to use RSA solutions.

2.3.2 Key Distribution

The distribution is a fundamental step of asymmetric cryptography, the private key must kept SECRET, the ublic instead must be distributed as widely as possible. But how is possible to guarantee the binding between public key and person identity? There are 2 possible solutions:

- Exchange of keys Out-Of-Band (e.g. keys party)
- Distribution by specific data structure (public key certificate)

Confidentiality without shared secrets is often used to send the secret key chosen for a symmetric algorithm, looking at figure 15.

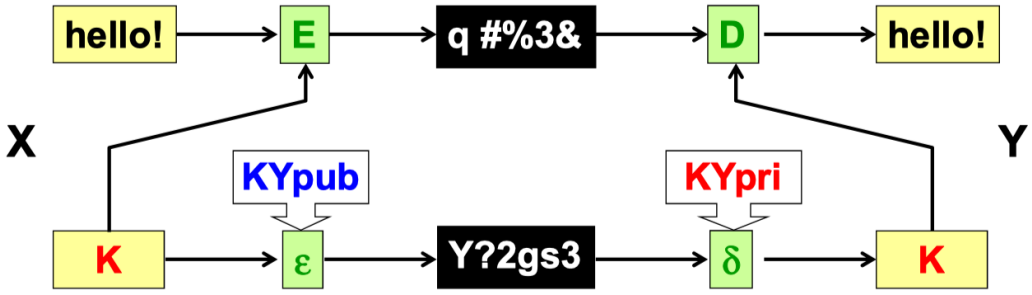

Figure 15: Confidentiality, no shared secret

2.3.3 Diffie-Hellman

Is the other well-know algorithm for asymmetric encryption. The idea behind the DH algorithm is good represented by figure 16.

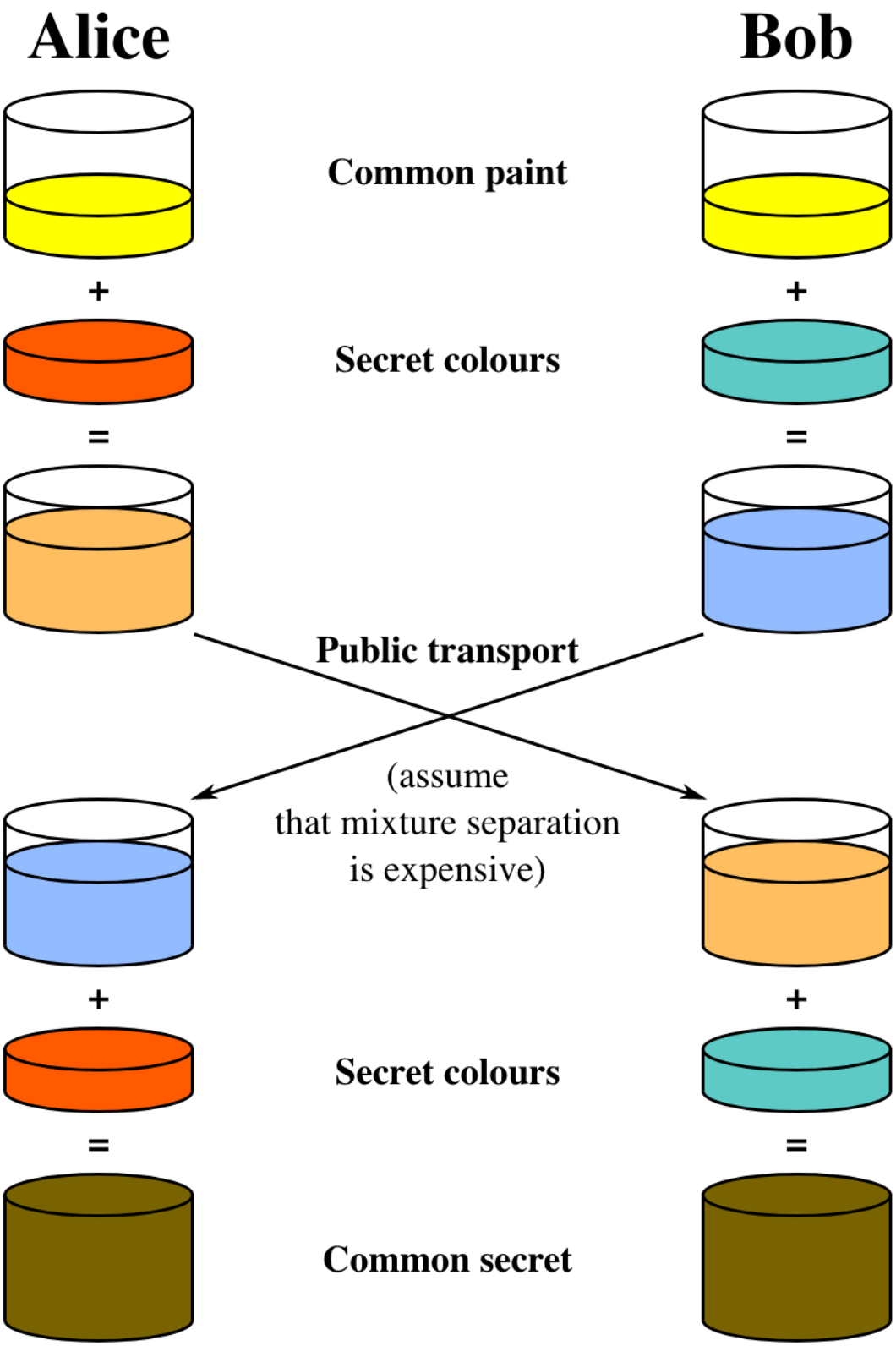

Figure 16: Diffie-Hellman idea

It was the first developed public-key algorithm, is frequently used to agree on a sceret key (*key agreement*), it was patented but is now expired. One advantage is that is sniffing resistant. It can be exploited by a MITM attack by manipulating the data like in figure 17.

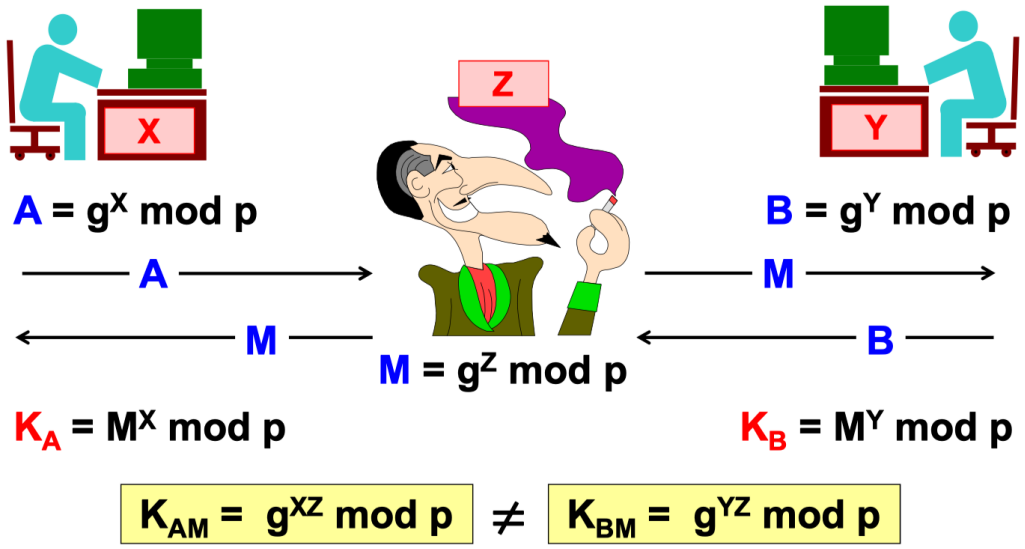

Figure 17: Diffie-Hellman MITM

The complexity of DH is a discrete logarithm, in any case also RSA, required the usage of quantum computer to be exploited in a reasonable amount of time.

2.4 Elliptic Curve Cryptography

This solution instead of using modular arithmetic, the operations are executed on the surface of a 2D (elliptic) curve. The problem of discrete logarithm on such a curve is more **more complex** than modular and allows **shorter keys** (about 1/10). A lot of algorithms has been rewritten, **EC**DSA or **EC**DH.

ECDH the EC version of Diffie-Hellman is the same as before but with the math of an elliptic curve.

- A and B select same elliptic curve and a point G of its
- A chose a random value x and computes: $X = xG$
- B chose a random value y and computes: $Y = yG$
- A and B exchange (publish) X and Y

- A computes $K = xY$
- B computes $K' = yX$

but the final result will be that:

$$K = K' = xyG$$

There are also other algorithms like ECDSA:

- Message digest computed with normal hash function (SHA-256)
- Signature = pair of scalars derived from the digest plus some operations on the curve

The ECIES instead:

- Generates a symmetric encryption key (AES-128) with operations on the curve
- Gives to the receiver the information (based on his public key) needed to recompute the encryption key

2.5 Integrity

In most of the cases the real needes of the people around the world is not the secrecy of the data, but is its integrity. Intercepting a communication and change it, in an unpredictable way, could be worse than read it. For this reasons the digest come in.
The **digest** is a summary of something, the name is related to the *reader digest* a magazine with the *summary* of famous books.
The message digest is a fixed-length summary of the message to be protected (of any lenght). Is important for a digest to be of a fixed-size without depending upon the lenght of the data to be summarizzed. It must be:

- **FIXED-SIZE**
- Fast to compute
- Impossible (or very difficult) to invert
- Difficult to create "collisions"

The digest is often used to avoid performing operations on the whole message, especially when the message is very large (slow pub-key crypto). The digest can be calculated in many ways, usually is calculated via ***(Cryptographic)* Hash Function**. The hashing algorithms are the faster in the world of cryptography, more than the symmetric ones and more than asymmetric, that are the slower.

The flow is simple (figure 18), the message M is splitted in N blocks $M_1...M_N$, the blocks are then computed by a base function f. Each block generate and hash code:

$$V_k = f(V_{k-1}, M_k)$$

For the first block the $V_0 = IV$ (Initialization Vector), public and specified by the algorithm code. The digest is generated by the last computation of the function:

$$hash = V_N$$

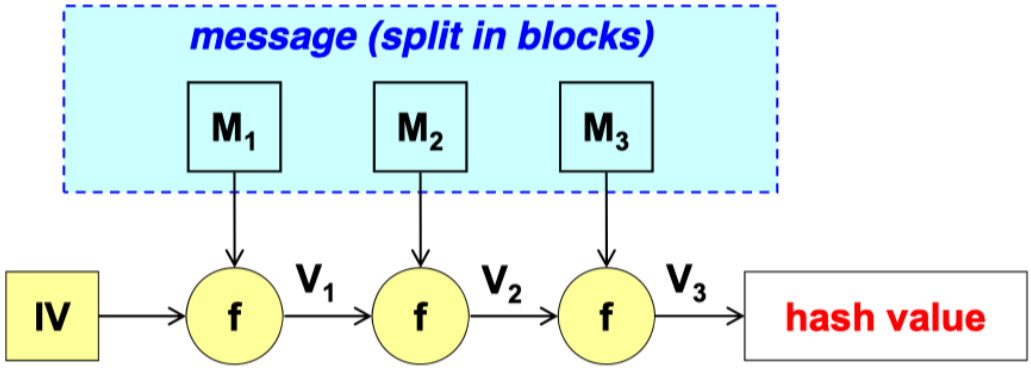

Figure 18: Hashing flow

The table show some advice about digest algorithms:

name	*block*	*digest*	*definition*	*notes*
MD2	8 bit	128 bit	RFC-1319	obsolete
MD4	512 bit	128 bit	RFC-1320	obsolete
MD5	512 bit	128 bit	RFC-1321	obsolete
RIPEMD-160	512 bit	160 bit	ISO/IEC 10118-3	good
SHA-1	512 bit	160 bit	FIPS 180-1	sufficient
SHA-224	512 bit	224 bit		
SHA-256	512 bit	256 bit	FIPS 180-2	good
SHA-384	512 bit	384 bit	FIPS 180-3	
SHA-512	512 bit	512 bit		
	SHA-2			
SHA-3	1152-576	224-512	FIPS 202 FIPS 180-4	excellent

Figure 19: Digest Algorithms

The lenght of the digest is important, must be long enough to avoid ***aliasing*** = **Collisions**:

- md1 = h(m1)

- md2 = h(m2)
- if (**m1** $\neq$ **m2**) $\Rightarrow$ md1 $\neq$ md2

For an algorithm well designed the probability of aliasing is:

$$P_A \propto 1/2^{Nbit}$$

for this reasons digest function requires a good number of bits (birthday paradox).

The are some code used to guarantee the integrity of the messages: **MIC** *(Message Integrity Code)*, to guarantee the authentication: **MAC** *(Message Authentication Code)* and to avoid the replay attacks **MID** *(Message IDentifier)*.

2.6 Authentication

The authentication is the process or action of verifying the identity of a user or process, or showing something to be true, genuine, or valid.

2.6.1 Symmetric Cryptography

Authentication can be performed in different ways. One possibility is the **auth by Symmetric Encryption**. This solution send also an encrypted copy of data:

- (sender) M* = enc(K, M)
- (transmission) M || M*
- (receiver) X = dec(K, M*)
- (verification) if (X == M) $\Rightarrow$ OK, else ALARM

Using this technique only who knows the (secret) key can compare the copy with the original. The disadvantage is that the performance are really poor, every operation must be executed 2 times, also the transmission.
Another similar technique is the **authentication by digest and symmetric encryption**, the adavantages are the data transmission is not doubled and only a digest is sent. The drawback is that two operations are required (digest + encryption).

- (sender) H = enc(K, hash(M))
- (transmission) M || H
- (receiver) X = dec(K, H)

- (verification) if (X == hash(M)) ⇒ OK, else ALARM

Is also possible to enforce authentication by means of **keyed-digest**, in this solution the digest sended along with the data is calculated over data an a private key together.

- (sender) d = digest(K, M)
- (transmission) M || d
- (receiver) d* = dec(K, M)
- (verification) if (d == d*) ⇒ OK, else ALARM

The solution require less overhead and only a single operation. One of the most used solution is the **HMAC** where:

$$hmac = H(K' \oplus opad || H(K' \oplus ipad || data))$$

In case of size problem CBC-MAC can be used, it exploits a block-oriented symmetric encryption algorithm, in CBC mode with null IV, taking MAC as last encrypted block. Both these solution are good, the only problem is the secret **shared** key, because the identity of the signer can't be distinguished, for the reasons is not good to be use in a commercial environment.
How is possible to combine both integrity and secrecy?

- Secrecy = Symmetric Encryption with K_1
- Integrity = Keyed-digest (MAC) with K_2

Some possibilities are:

- Auth&Encr (eg. SSH)
  - enc(p, K_1) || mac(p, K_2)
  - Need decrypt before check integrity
- Auth-then-Encr (eg. TLS and SSL)
  - enc(p || mac(p, K_2), K_1)
- Encr-then-Auth (eg. IPsec)
  - enc(p, K_1) || mac(enc(p, K_1), K_2)

Improper combination of secure algorithms may leed to an insecure result!
The AE tecniques allows to achieve both privacy and authentication (and integrity):

- One key

- One algorithm
- Better Speed
- Less error in combining

Are often used in applications, e.g. emails, networks, etc...
The RFC 5116 release define Interface and algorithms for authenticated encryption , for example the AEAD in figure 20.

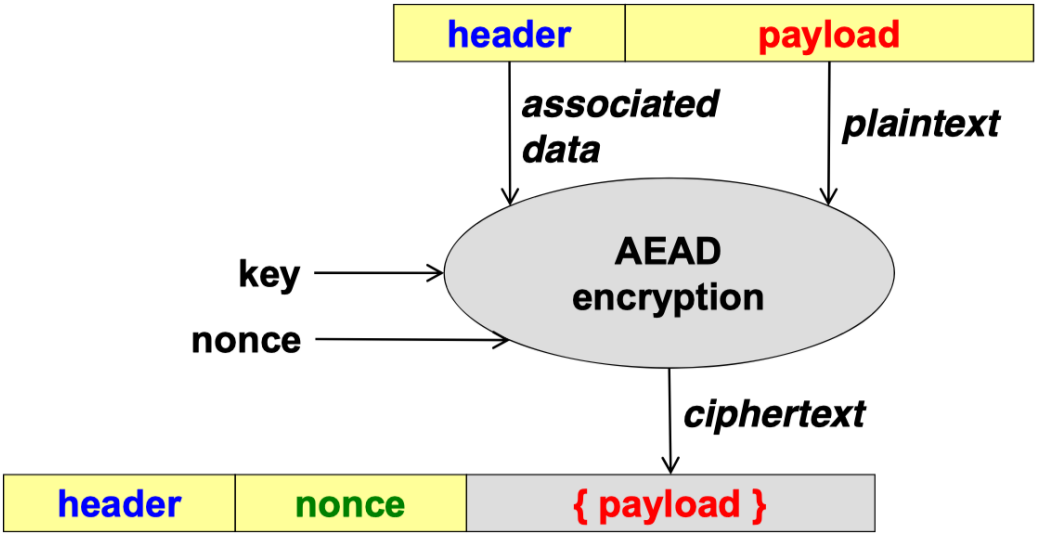

Figure 20: Authenticated Encryption with Asociated Data (AEAD)

The idea of AEAD is that comparing the two header, the one ciphered and the plain one, will guarantee the integrity.
The **IGE** (*Infinite Garble Extension*) is a modde of application and is not too much used because is too simple to be exploited, the flow is in figure 21.

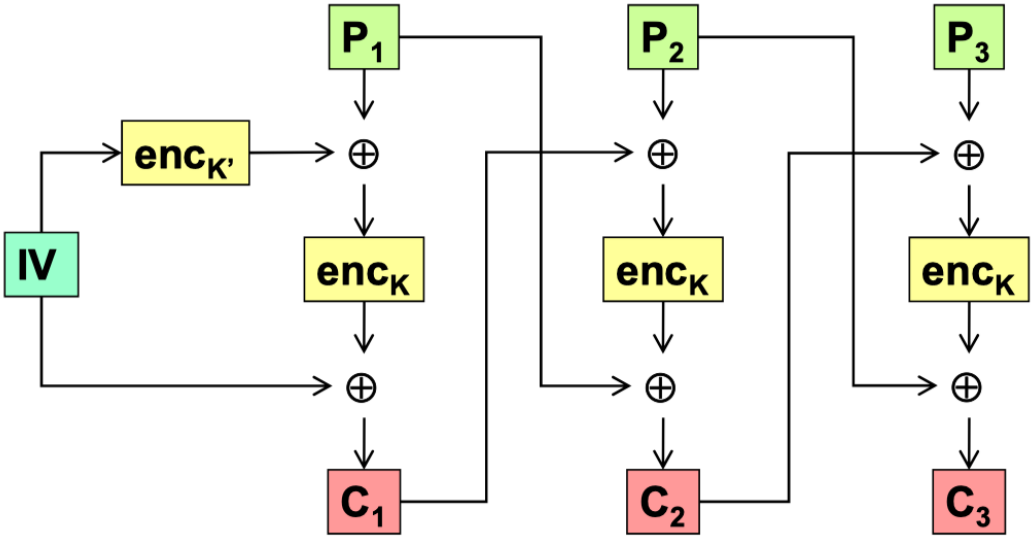

Figure 21: IGE (Infinite Garble Extension)

The AE standard are:

- OCB 2.0
- AESKW (AE Key Wrap)
- CCM (CTR mode with CBC-MAC)
- EAX
- Encrypt-then-MAC
- GCM

2.6.2 Asymmetric Cryptography

Authentication can be also achieved by using **asymmetric** encryption. For example the **auth by digest and asymmetric crypto** is one of the first implementations:

- (signer S) H = enc(S_Kpri, hash(M))
- (transmission) M || H
- (verifier V) X = dec(S_Kpub, H)
- (verification) if(x == hash(M)) ⇒ OK, else ALARM

those who knows the public key of the sender can compare the transmitted digest with the digest calculated on the received data. This is also know as **Digital Signature**! The figure show the behaviour.

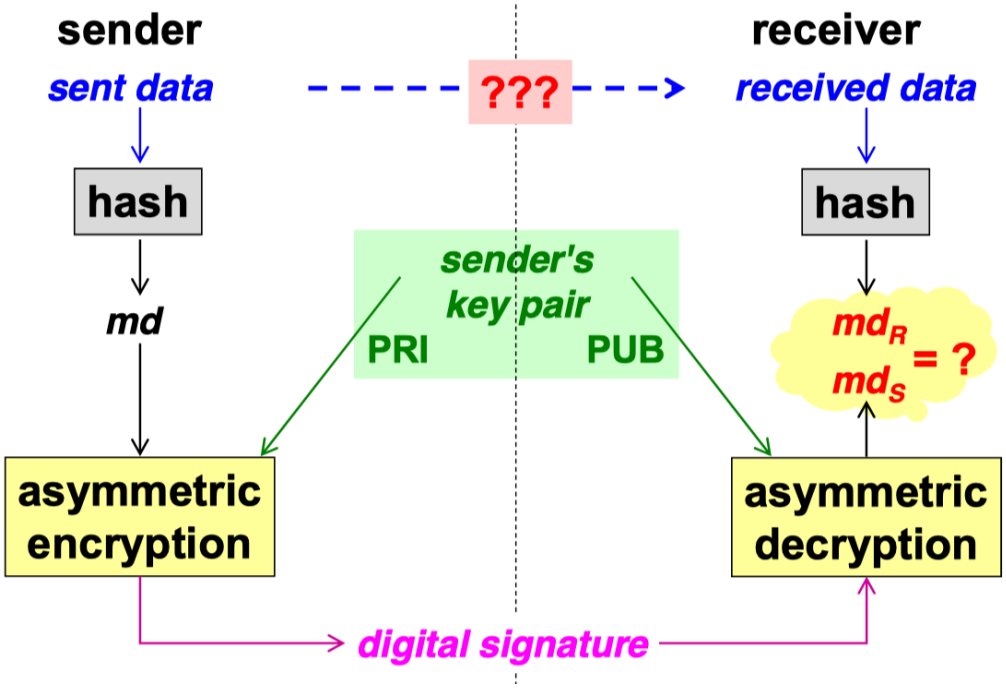

Figure 22: Digital Signature

DS can be achieved with RSA-based algorithms, they must be:

- Resistant to collisions (even just to avoid generating accidentally the same signature)
- Difficult to invert

2.6.3 Analysis

The A-I by means of a shared secret:

- Useful only for the receiver
- Cannot be used as a proof without disclosinng the secret key
- Not useful for non repudiation

Instead, by means of asymmetric encryption:

- Being slow it is applied to the digest only
- Can be used as a formal proof
- Can be used for non repudiation
- = digital signature

The main advantage of a digital signature over an handwritten one is that it can guarantee both authentication and integrity because is tightly bound to the data.

2.6.4 Public Key Certificate

A data structure used to securely bind a public key to some attributes, the idea is to bind a key to an identity, also other association are possible. The key will be digitally signed by the issuer: the **Certification Authority** for a limited amount of time. It can be revoked on request both by the user and the issuer. Some of the most common formats for pub key certificates are:

- X.509: ISO + IETF
- non X.509: PGP
- PKCS#6: RSA

X.509 is oe of the most used certificate format, the structure il composed of:

- Version
- Serial number
- Signature Algorithm
- Issuer
- Validity
- SUBJECT
- SUBJECT PubKeyInfo
- CA Digital Signature

An example in figure 23:

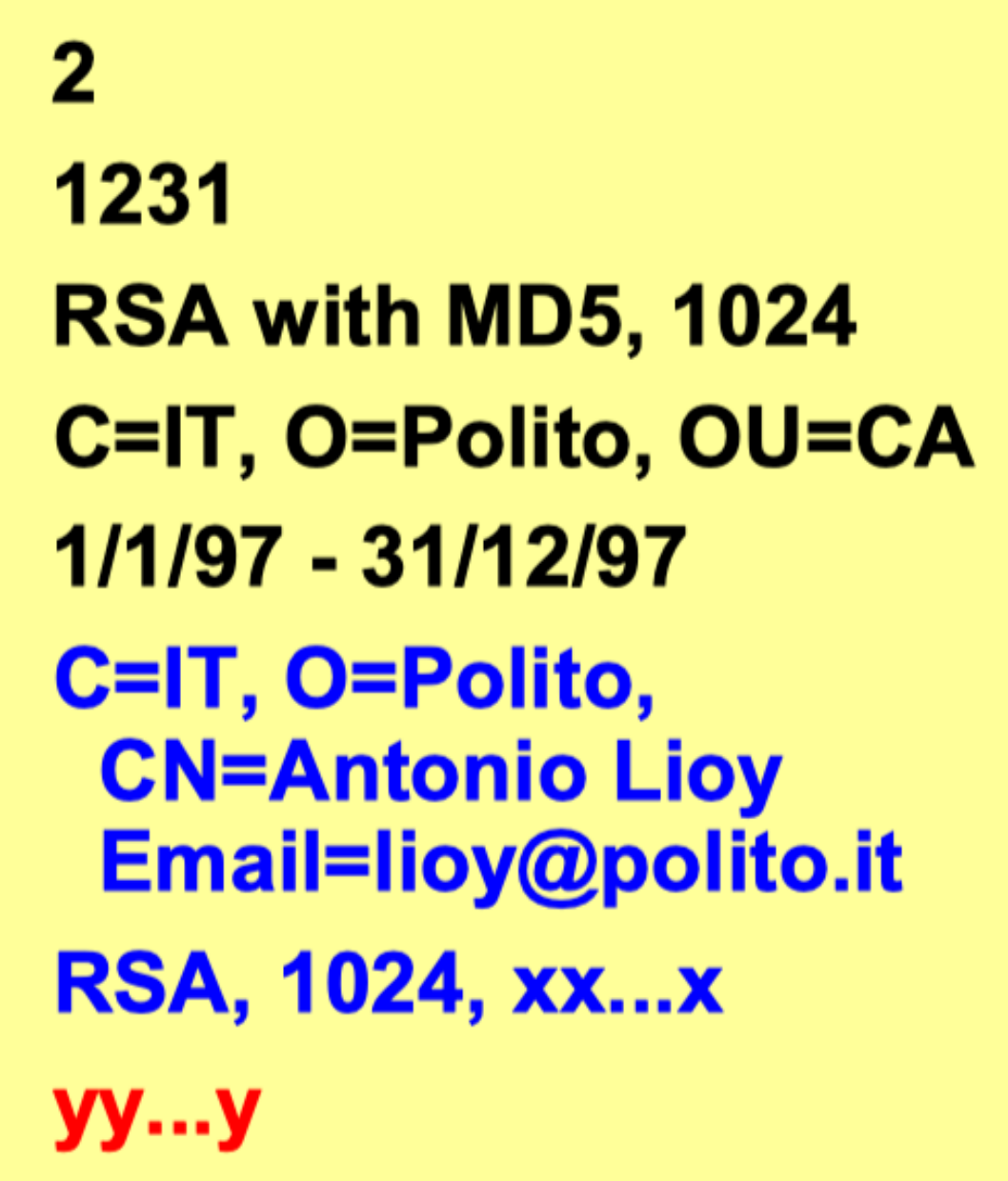

Figure 23: X.509 Certificate Example

The **PKI** (*Public-Key Infrastructure*) is the Infrastructure, technical and administrative, put in place for the creation, distribution and revocation of public key certificates. Any certificate can be revoked before its expiration date:

- By the owner
- By the creator

when a signature is verified, the receiver must (and is a responbilitity) check

that the certificate was valid at signature time. There are 2 type or revocation mechanisms:

- CRL (Certificate Revocation List):
  - List of revoked certificates
  - Signed by the CA or by delegated party
- OCSP (On-line Certificate Status Protocol)
  - Response containing information about certificates status
  - Signed by the server

How is made the verification of a signature? Look at figure 24.

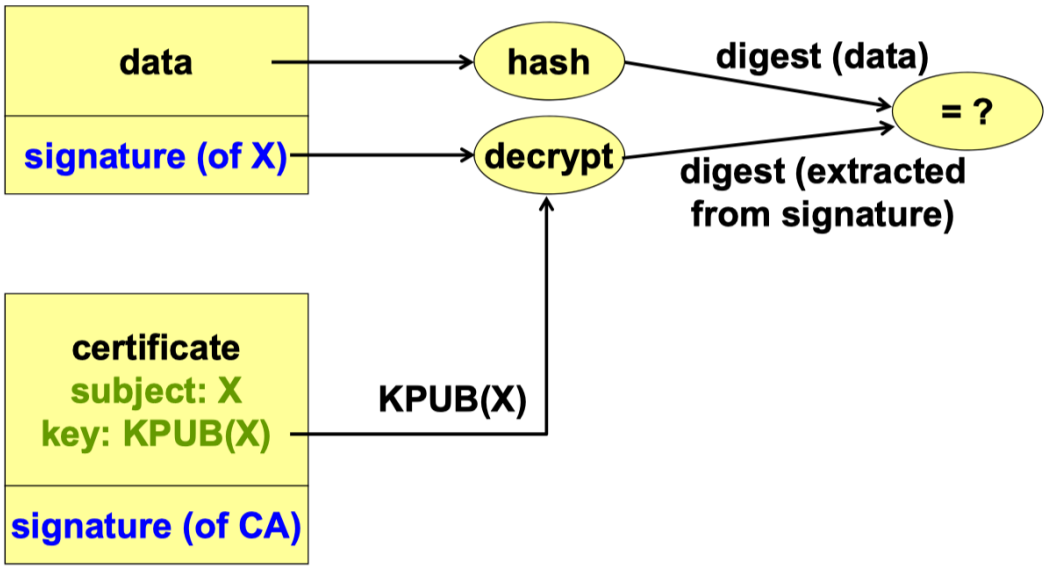

Figure 24: Verification of signature/certificate

A hierarchical Infrastructure is necessary to correctly verify the authenticity. At a certain point a TOP-LEVEL CA is needed, like "god". An example is showed in figure 25:

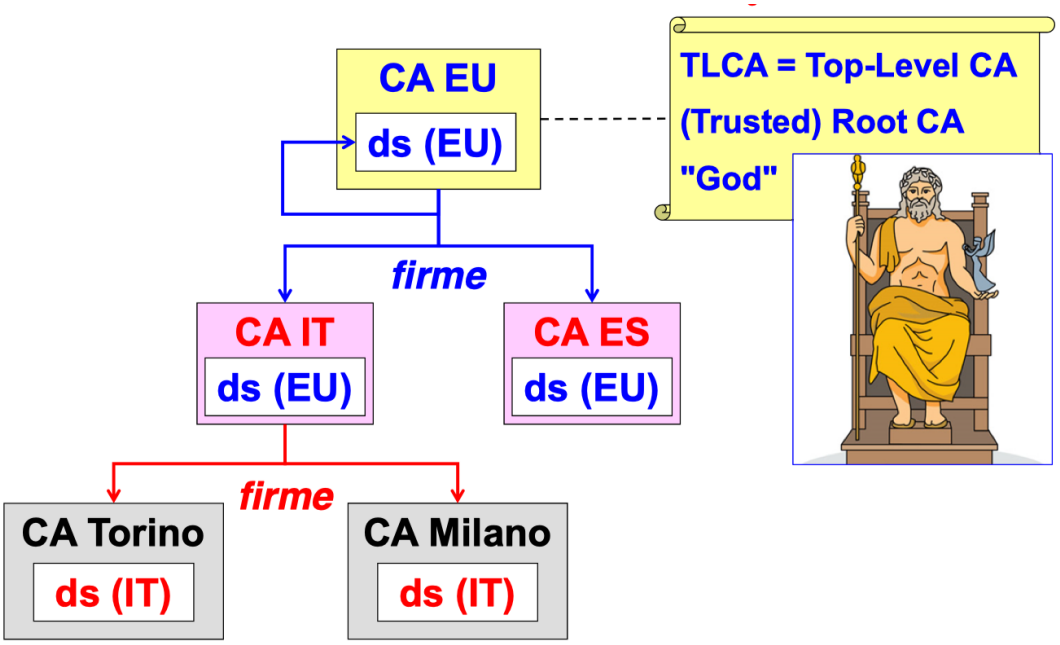

Figure 25: Certification Hierarchy

3 Security of IP networks

3.1 Introduction

Nowadays networks are become a fundamental basis of out life. Every devices is now connected or link the internet in some ways. The lead to a lot of security problem. Computers are faceing these issue from years, but refrigerators not. Is important to ensure a reliable and secure connections between the appliance and the internet world.

3.2 Authentication

The modern way to allow network access is to using an authentication protocol to an AAA server. The most used protocol is the auth via PPP channels. **PPP** is a protocol to encapsulate network packets (LV3, e.g. IP) and charry them over a point-to-point link. I could be of different type:

- Physical (RTC, ISDN)
- Virtual L2 (xDSL with PPPoE)
- Vistual L3 (L2TP over UDP/IP)

Activated in 3 sequential steps:

1. LCP (Link Control Protocol)
2. Authentication (optionall PAP, CHAP or EAP)
3. L3 Encapsulation (IPCP, IP Control Protocol)

The three possible protocols for authentications are:

- **PAP** - Not more good!
  - Password Auth Protocol
  - Pwd send in clear
- **CHAP**
  - Challenge Handshake Auth Protocol
  - Symmetric Challenge (based on user password)
- **EAP** - Framework (USED)
  - Extensible Auth Protocol
  - External Techniques

The EAP is a flexible L2 authentication framework, it use come predefined mechanis such as: MD5-challenge (similar to CHAP), OTP or Generic Token Card. Also other mechanisms may be added by the RFC standard, like the RADIUS support.
The authentication data are transported via its own encapsulation protocol (because L3 packets are not yet available), the main features are:

- Indipendent of IP (Supports any link layers PPP, 802)
- Explicit ACK/NAK (no windowing)
  - Assumes no reordering (PPP guarantees ordering, UDP or raw IP do not)
- Rentransmission (3-5 max)
- No fragmentation

The link created by EAP is not assumed to be physically secure, EAP auth methods must provide security on their own. Some methods are:

- EAP-TLS
- EAP-MD5
- EAP-TTLS
- EAP-SRP
- GSS_API (ticket for Kerberos realm)
- AKA-SIM (Used for mobile SIM auth)

In figure 26 the schema of EAP architecure:

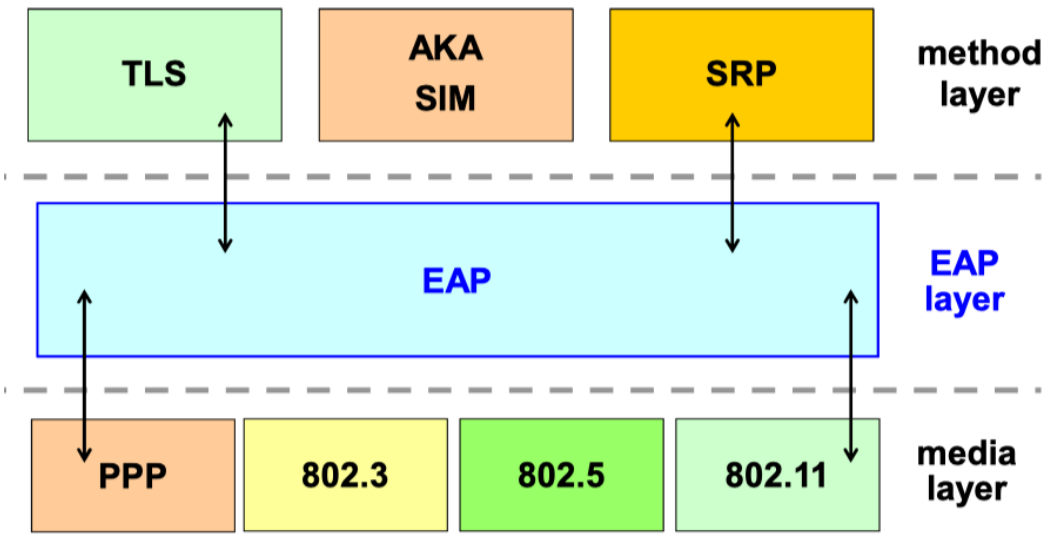

Figure 26: EAP Architecture

The request will be sended to a **NAS** (Network Access Server), the feature required by the NAS manufacturer are:

- Authentication: *Entity's identity is authenticated based on credentials*
- Authorization: *Determining whether an entity is authorized to perform a given activity or access to resources*
- Accounting: *Tracking network resource usage for audit support, capacity analysis or cost billing*

the Autentication Server performs these three functions talking with one or more NAS via one or more protocols.

3.2.1 Authentication Protocols

The most known are:

- **RADIUS**: de-facto standard, proxy towards other AS.
- **DIAMETER**: evolution of RADIUS, roaming between different ISP, security.
- **TACACS+**: Better than RADIUS but proprietary solution of Cisco.

RADIUS *Remote Authentication Dial-In User Service* was developed in 1991 by Livingston technologies. It supports authentication, authorization and accounting to control network access, both on virtual and physical ports. It allow a centralized administration and accounting service, with a client-server schema between NAS and AS. It use port 1812 UDP for authentication and 1813 UDP for accounting.
One of the main feature is the ability to act like a proxy for other RADIUS db, an example is eudoram, that is the greatest RADIUS server in the world, and it's used by all the European University. In figure an example of work:

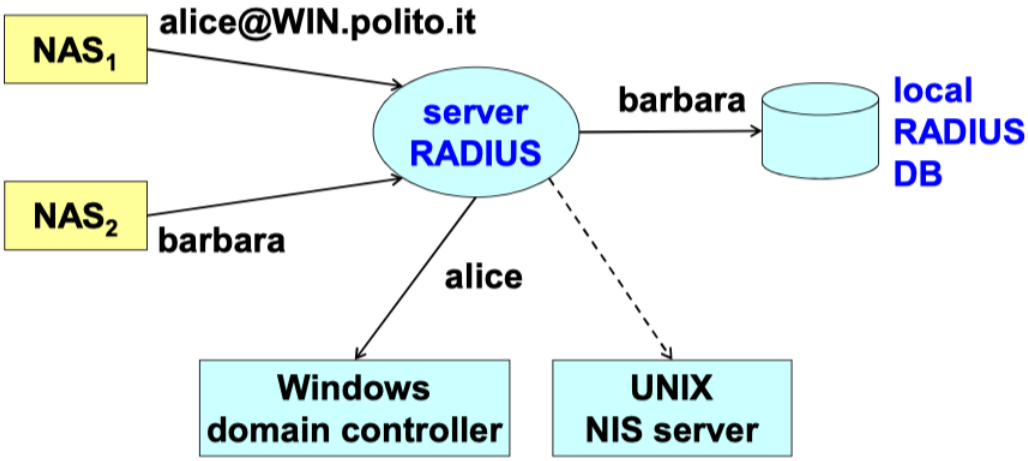

Figure 27: RADIUS example

Some security functions over attacks are implemented:

- Sniffing NAS request
- Fake AS response
- Chaging AS response
- PWD enumeration

Packet integrity and authentication via keyed MD5:

$$pwd \oplus MD5(key + authenticator)$$

The user authentication via PAP, CHAP, token-card or EAP. The attributes are in TLV form, easily extensible without modification installed base. The types of the packet can be ACCESS-:

- REQUEST: Contains access credentials
- REJECT: Access is denied
- CHALLENGE: Request additional info from the user
- ACCEPT: Access granted + network parameters given

Th RADIUS authentication has two pourpose, server reply authentication and no replay and masking the password. During the ACCESS-REQUEST step it is name Request Authenticator (16 byte randomly generated by the NAS), in the other steps is named Responde Authenticator and it is computed via keyed-digest:

$$md5(code||ID||length||RequestAuth||atrtibutes||secret)$$

The attributes of the RADIUS are:

- type=1 (Username) - Network Access Identifier
- type=2 (User password)
- type=3 (Chap password)
- type=60 (CHAP-Challenge)

The NAI has been defined by the RFC-2468 and is made of:

$$NAI = username[@realm]$$

the user name is the one used in the PPP authentication phase. The figure 28 shown the flow of the protocol.

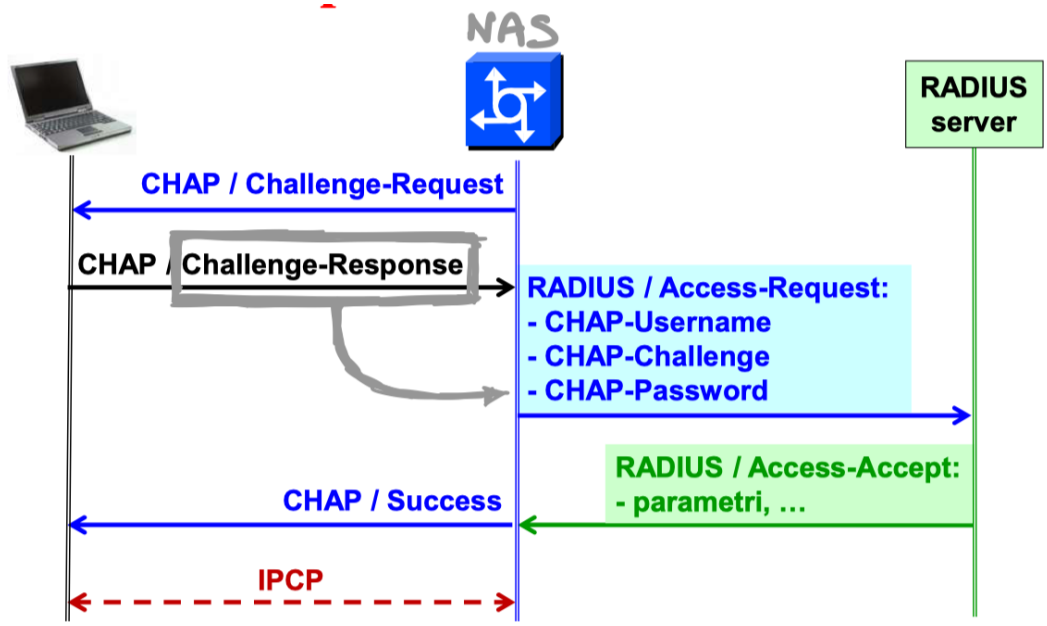

Figure 28: CHAP + RADIUS

DIAMETER was developed like the RADIUS evolution (twice the radius), it has a special emphasis on roaming between ISPs. It guarantee protection via IPsec or TLS, both compulsory for the server side.

IEEE 802.1x is a port-based Network Access Control: L2 architecture, usefull in a wired connection to block access, absolutely needed in wireless networks. The first implementations was born for Windows and Cisco wireless access-point. Is an authentication and key-management framework, it can derive keys for management, authentication, integrity and secrecy of packets, using standard algorithms for key derivation like TLS, SRP, etc... It can also

provide other security services.
The main advantages for 802.1x is that it exploits the application level for the actual implementation of the security:

- Direct dialogue between supplicant and AS (NIC-NAS "pass-trough device")
- No cange needed on NIC and NAS to implement new mechanisms
- Perfect integration in AAA

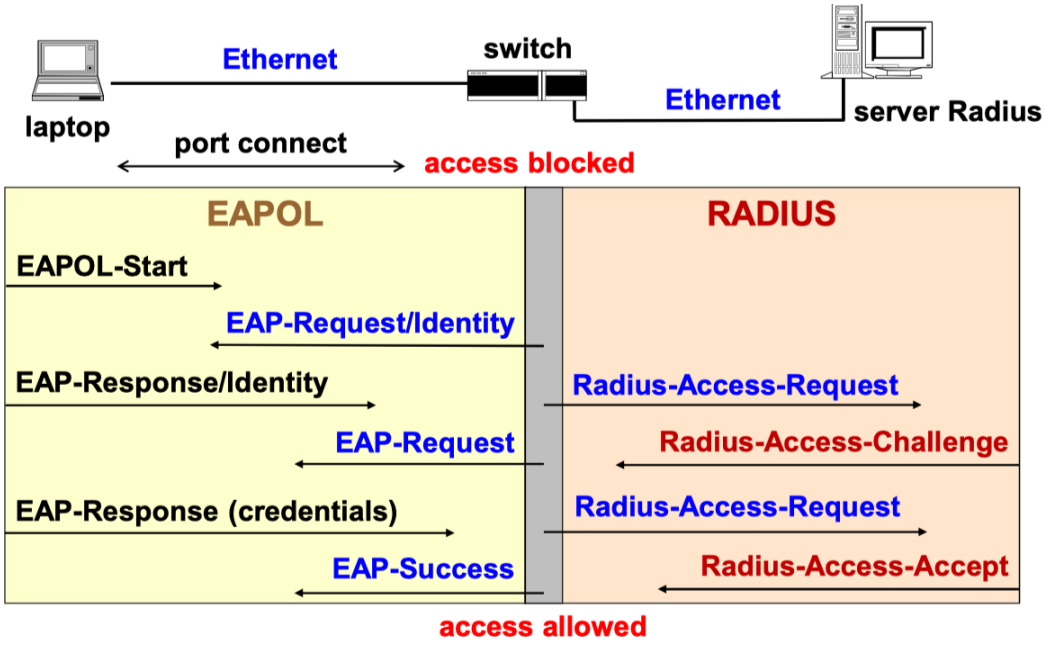

Figure 29: 802.1x Flow

Which is the best level for apply security? The upper we go in the stack, the more specific are the security functions (e.g. Identify user, commands, data) and independent from the underlying network... But we leave more room for DoS attacks. The lower level we go in the stack, the more quickly we can expel the intruders, but we have also fewer data to make the decision (e.g. MAC, IP). The dest way is to put a little bit of security in every step of our stack. At low level, for faster expulsion, high level for more precise expulsion.

3.3 DHCP

DHCP is non-authentication protocol (really bad, trivial activation of shadow server). The main kind of attacks are:

- DoS: Provides wrong configurations.

- MITM: Good configuration but /30 subnet, attacker like gateaway. With NAT also reply intercepted.

There are some solution to increase security but are not so much used, like DHCP snooping or HMAC-MD5 to authenticate messages (too complicated, required shared secret). In case a RADIUS server is available, is better to use it to gain network configurations.

3.4 Security LV3 - VPN

At LV3 security can be achived by two main solutions, securing all the networks, or by securing some paths.
What is a **VPN**? *Virtual Private Network* is a technique hardware and/or software to create a private network while using a shared (or anyway untrusted) channels and transmission devices. There are several techniques to create a VPN:

- Private Addressing
- Protected Routing (IP Tunnel)
- Cryptography protection of the network packets (secure IP Tunnel)

The first, is not too much secure, the network to be part of the VPN use non-public addresses so that they are unreachable from other networks. This solution can be easily defeated if somebody discovers the addresses, it can sniff the packets during transmission and it has access to the communication devices.
The other solution, via Tunnel, use the router to encapsulate whole L3 packets as a payload inside another packet: IP-IP, IP-MPLS, other. THe router must perform access control to the VPN by ACL (Access Control List). This protection can be defeated by anybody that manages a router or can sniff packets during transmission. In figure 30 a schema of the process:

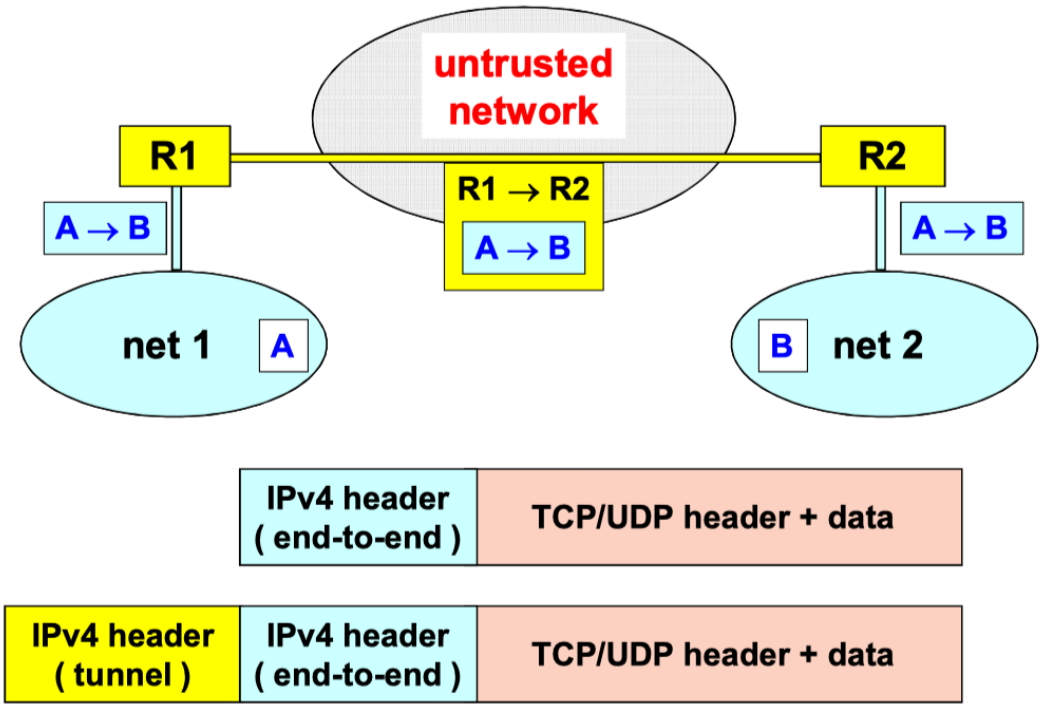

Figure 30: VPN via IP Tunnel

If packet has size equal to the MTU, then encapsulation will only possible with fragmentation, this means performance loss, that can't be grater than 50%.
The best solution is the VPN via secure IP tunnel, before the encapsulation, the packets are protected with:

- MAC (integrity + authentication)
- Encryption (confidentiality)
- Numbering (to avoid replay)

If the cryptographic algorithms are strong, then the only possible attack is to stop the communications. This solution is a.k.a as S-VPN (Secure VPN), figure 31.

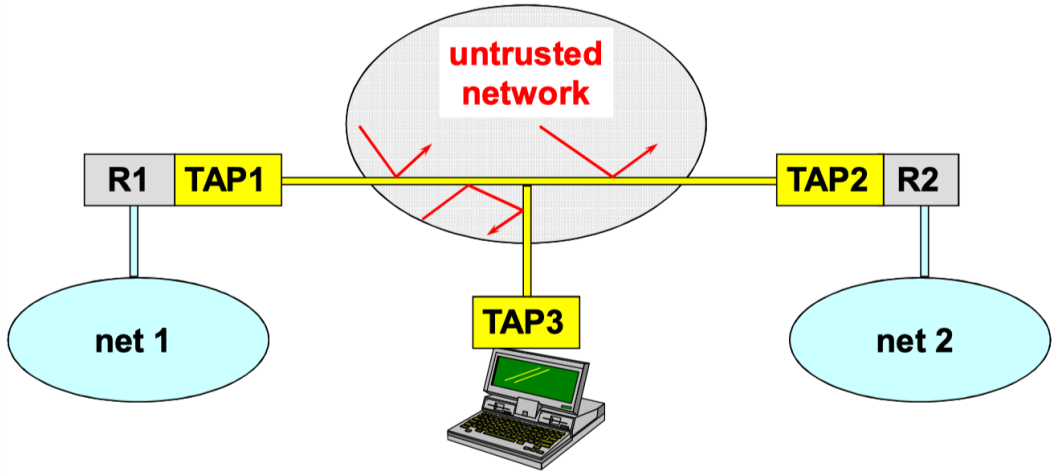

Figure 31: Secure VPN schema

3.4.1 IPsec

One of the most used VPN solution is **IPsec**, defined by IETF, is an architecture for L3 security in IPv4/IPv6, to create S-VPN over untrusted networks, or to create end-to-end secure packet flows. It defines to specific packet types:

- **AH** (*Authentication Header*): Integrity, Authentication, no replay
- **ESP** (*Encapsulating Security Payload*): Confidentiality

It make use of IKE for keys exchange.
The **Security Association** (SA) is an unidirectional logic connection between two IPsec systems, two SA are needed to get complete protection of a bidirectional packet flows. Each SA is indipendent and it is associated with different security services, they can be also the same.
IPsec make use of some sort of DB for managing connections:

- SAD (SA Database): Not a real DB, is the list of active SA and their characteristics
- SPD (Security Policy Database): List of security policy to apply to the different packet flows, configured a-priori or by an automatic system.

The **transport mode** of IPsec is used for end-to-end security, that is used by hosts, not gateways. The advantage is that is computationally light, the cons is that there isn't protection for the header variable fields. The **tunnel mode** instead, is used to create a VPN by gateways, the advantages are the protection of header variable, the problem is that is computationally heavy.
The AH, that ensure data integrity and sender authentication, the first version was supporting keyed-MD5 and optionally of keyed-SHA-1, now in ensure also

(partial) protection from replay attack and it use HMAC-MD5-96 and HMAC-SHA-1-96. Figure 32 show the flow of the architecture.

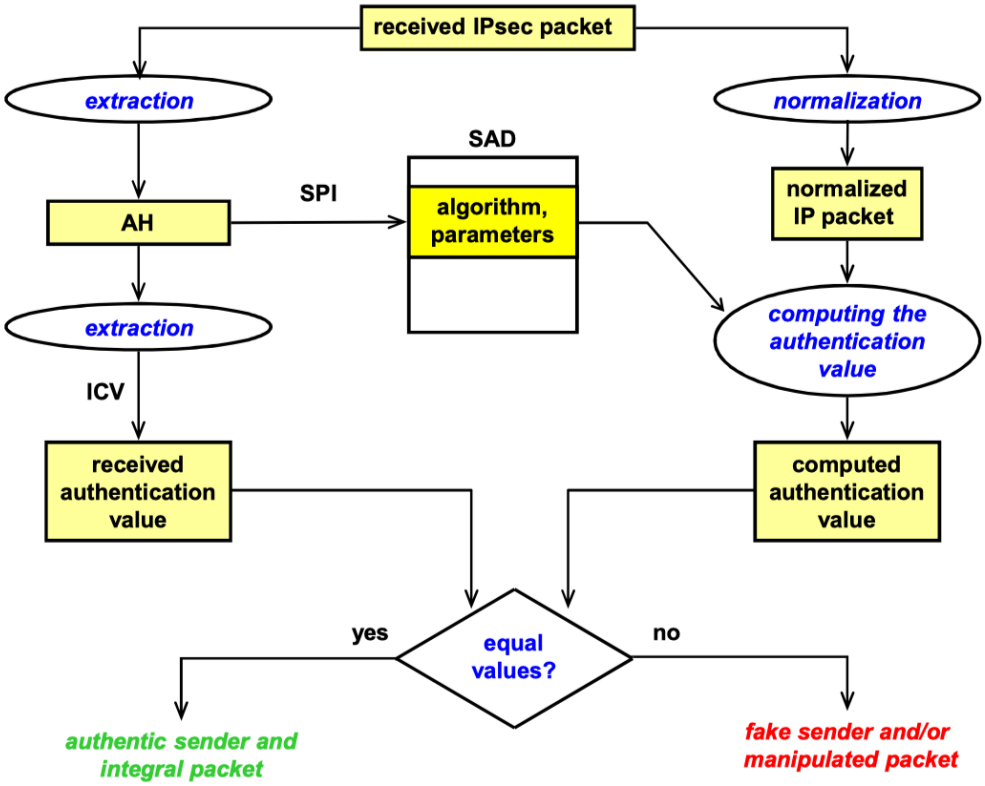

Figure 32: IPsec flows

The normalization of AH it reset the TTL and the Hop Limit Field. If the packet contains a ROuting Header, then:

- Set the destination field to the address of the final destination
- Set the content of the routing header to the value that it will have at destination
- Set the Address Index field at the value that it will have at destination

ESP (*Encapsulating Security Payload*) is a solution to encapsulating IP packets. With the latest version is possible to provide confidentiality and authentication. Using it in trasport mode have some conseguence:

- + Payload is hidden (including info needed for QoS, filtering or intrusion detection)
- - The header remains in clear

Using the tunnel mode something is changing:

- + Hides both payload and (original) header
- - Larger packet size

The last implementation use the ESP-DES-CBC format.

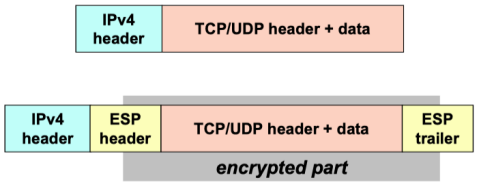

Figure 33: ESP: transport mode

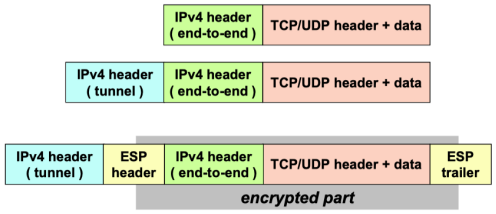

Figure 34: ESP: tunnel mode

The IPsec implementation details are:

- VPN-A: ESP/3DES-CBC/HMAC-SHA1
- VPN-B: ESP/AES-128-CBC/AES-XCBC-MAC-96

The NULL algorithms for ESP:

- For authentication or privacy (not simultaneosly)
- Protection vs. performcae trade-off

Sequence Number:

- Partial protection from replay
- Minimum window of 32 packets (64 suggested)

The real **replay protection** follow a sequence of actions:

1. After SA is established, sender initilizes sequence number to 0
2. For each packet the Seq.Numb is incrementend
3. At the reach of the maximum sequence number, the new SA should be negotatiated

The IPsec v3 use: AH is optional and ESP is mandatory, support for single multicast source, uses extended sequence number (64 bits), support AEAD and add some clarifications about SA and SPI (for faster lookup). Multiple combinations of VPN can be exploited together in order to use of the different benefits between the different modes:

- trasport-mode between to hosts
- tunnel-mode between to gateways
- 1st and 2nd together
- Host and gateways with tunnel mode

Key management is one of the most important component of IPsec, is in charge of provides, to both parties, the symmetric keys used for packet authentication and/or encryption. The distribution can be achived in two ways:

- OOB
- Automatic in-band

The protocol used by IPsec is the **ISAKMP** (*Internet Security Association and Key Management Protocol*) it negotiate, set-up, modify and delete SA, the exchange is not fixed, the one used is OAKLEY. The whole phase is managed by **IKE**, *Internet Key Exchange*, it create SA to protect the ISAKMP exchange, this created SA is used to protect the negotiation of the SA needed by IPsec traffic, the same SA can be used several times to negotiate other IPsec SA.
The IKE flow is divided in two phases:

1. Negotiation of a bidirectional ISAKMP SA: *main or aggressive mode*
2. Negotiation of the IPsec SA: *quick mode*

The different modes of operation are:

- Main mode: 6 msg, protect parties identities
- Aggressive mode: 3 msg
- Quick mode: 3 msg, negotiation only of the IPsec SA
- (new) Group mode: 2 msg, already used for update SA parameters

The IKE process allows different authentication methods:

- Digital Signature: non-repudiation of the IKE Negotiation
- Public Key Encryption: identity protection in the aggressive mode
- Revised Public Key Encryption: less expensive, only 2 public-key operations
- Pre-shared Key: party ID may only be its IP address

System requirement for IPsec appliance:

- Router: Powerful CPU or Crypto AES-NI accelerator
- Firewall: Powerful CPU
- VPN Concentrator: (special-pourpose appliance terminator of IPsec tunnel)

Of course the IPsec tunnel encapsulation require a reduce of the throughput due to 2 main reasons:

- Larger Packet Size: AH (+24 bytes), ESP-DES-CBC (¿= 32 bytes)
- Larger number of packets: SA activation and update

Usually the decrease is not too much important, with a proper hardware, there are some exception like in Point-To-Point link that used L2 compression that now becomes useless or counterproductive when applied to ESP packets.

3.4.2 IP insecurity

The intrinsic insecurity of IP protocol drive insecurity to all the services transported by it, the main problem are the not authentication of addresses and the unprotected packets (integrity, authm confidentiality and replay).

ICMP is the *Internet Control and Management Protocol* the is vital for the network management, the main problem is that is used for many attacks because it has no authentication. Some example of attacks are:

- Smurfing: Using broadcast addresses to lot of replay to a single host
- Fraggle: Similar to smurfing but with UDP packets
- ARP Poisoning
- TCP SYN Flooding: Multiple open connection TCP

3.4.3 DNS

The goal of the DNS is to translate from name to IP and from IP to name. Is a again a vital service, the query are performed over UDP@53, is not secured at all.
One of the main attacks performed exploiting DNS leak is the shadows server by sniffing to intercept the queries and spoofing to generate a fake answers. Other possible attacks are the DNS cache poisoning.
For a lot of reasons the **DNSsec** is needed to avoid attacks like the Kaminsky one (figure 35), is now available from some public like Cloudflare DNS.

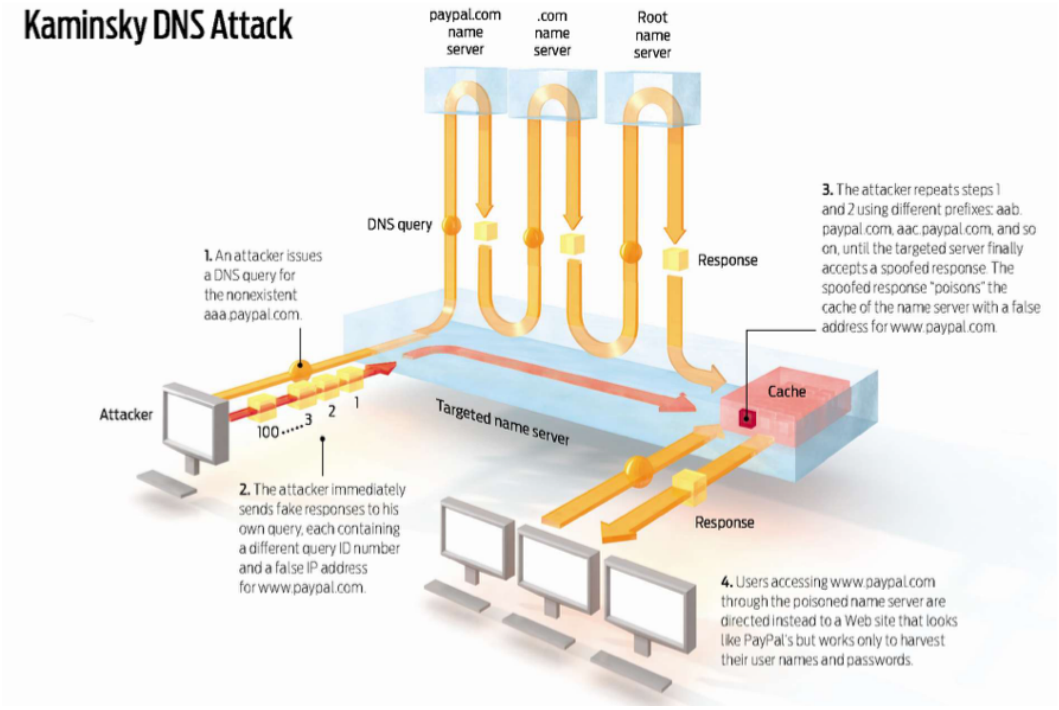

Figure 35: Kaminsky Attack

This new solution peform: Digital Signature of DNS records. It must face over some issues like not signed DNS response, no root CA, bigger record size and scarse experimental results.

Routing security another low critical appliance are the router, normally they have low security management for accessing like telnet or SNMP, there is also low security in the exchange of routing tables.

4 X.509

The third version of X.509 was developed in June 1996, the goal is to groups together in a unique document the modifications required to extend the definition of certificate and CRL. Two types of fundamental extensions are available:

- **Public**: That is defined by the standard and consequently made public to anybody
- **Private**: Unique for a centain user community

X.509 was useless before the idea of the extensions. The extensions can be defined critical and non critical:

- In the verification process the certificates that contain an unrecognized critical extension MUST be rejected
- A non critical extension MAY be ignored if it is unrecognized

The different processing is entirely the responsibility of the party that performs the verification: the **Relying Party (RP)**.
The public extension classes defined by the v3 of X.509 include:

- Key and Policy information: Author, subject, key usage, etc...
- Certificate subject and certificate issuer attributes
- Certificate path constraints: Basic, name and policy
- CRL distribution points

The **key usage** identifies the application domain for which the public key can be used (nonRepudiation, keyEnchipherment, dataEncipherment, keyAgreement, digital signature, cRLSign). It can be critical or not critical, if critical it can be used only for the scopes for which the corresponding option is defined.
The **Certificate Subject and Certificate Issuer Attributes** can have:

- Subject Alternative Name (SAN): Different formalisms to identify the owner (Mail, IP, URL)
- Issuer Alternative Name
- Subject Directory Attributes

The Basic constraints is used to indicates if the subject of the certificate can act as a CA. If true, also the depth of the certification tree can be defined. It can be critical or non critical. The name constraint field is used only by CA, it define the space of names that can be ceritified by the CA and the possibility of subTree. The last possibilities, the CRL distribution point, identifies the distribution point of the CRL to be used in validating a certificate, it can be a directory entry, mail or URL and it can be critical or not.
The **Private Extensions** that is extensions common to a certain user community.

4.1 CRL

Is the certificate revocation list, they are issued periodically and maintained by the certificate issuers. Of course they are digitally signed:

- By the CA that issued the certificates
- By a revocation authority delegated

A set of extensions have be developed for the CRLv2, the **crlEntryExtensions** and the **crlExtensions**. The timeline of a revocation is shown in the following figure:

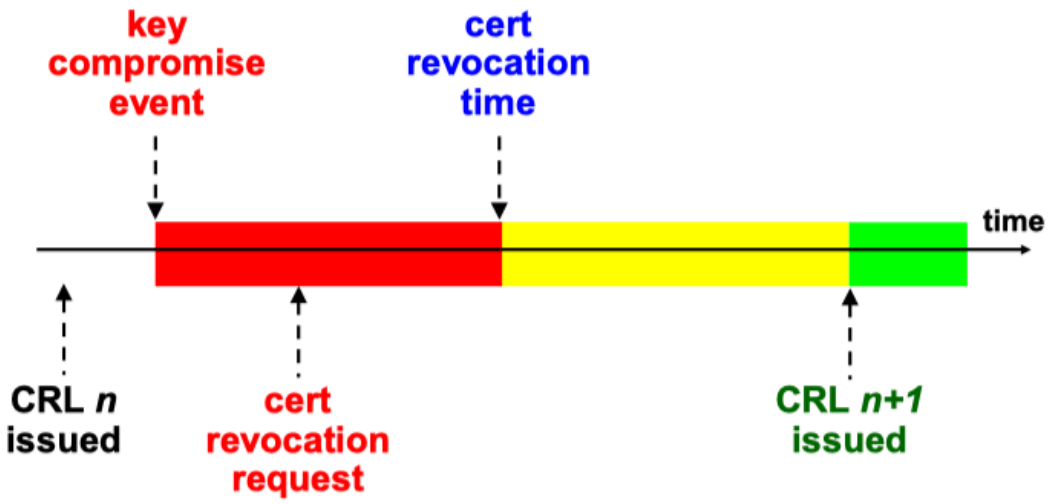

Figure 36: Revocation Timeline

The timeline is divided in three parts:

- RED: Still valid certificate (Revocation requested)
- YELLOW: Cert still valid for who not already informed (dangerous)
- GREEN: Safe zone

When a key is issued is also important to verify the revocation time policy, they can varying from minutes to days. A solution to improve the efficency of this protocol is to use OCSP.
The **OCSP** *Online Certificate Status Protocol* is a standard to verify online if a certificate is valid: Good, revoked or unknown. The response is signed by the server. The OCSP server certificate cannot be verified with OCSP itself. The response of the OCSP server can be precomputed, this reduce the load over the server, but makes possible replay attacks. Is also possible to obtain information not from CRL. The responder can be of 2 types:

- **Trusted**: Response signed with a pair key;cert independent of the CA for which it is responding. (Company responder or TTP paid by users)
- **Delegated**: The OCSP server signs the responses with a pair which is different based on the CA for which it is responding. (Paid by CA)

4.2 Time stamping

Is a really important feature. Is a proof of creation of data before a ceratin point in time, the requested is managed by a Time-Stamping Authority. The flow is the following:

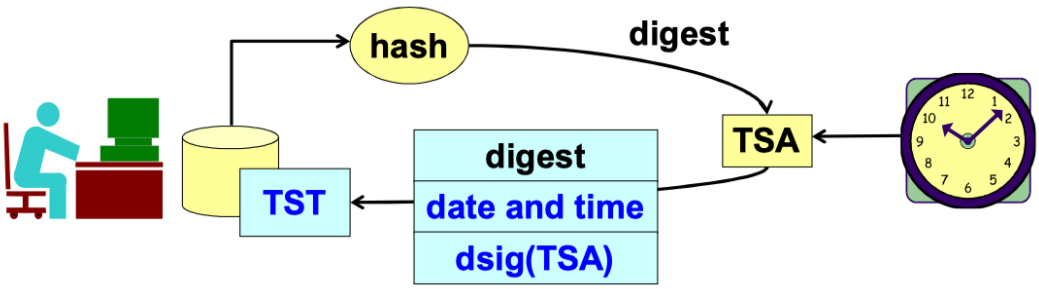

Figure 37: Time-Stamping procedure