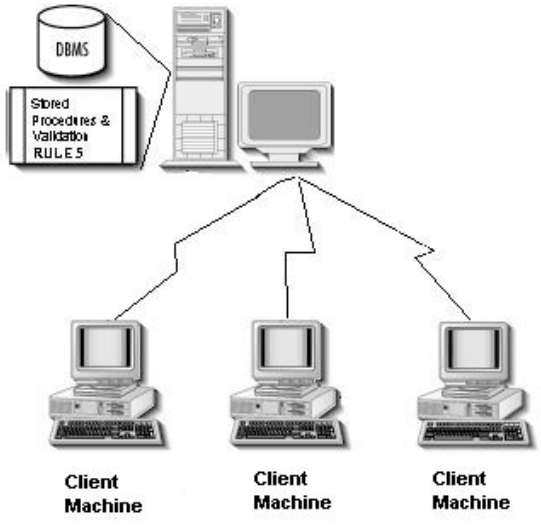
Computer Networks
Chapter 8 - Network Security

Marino Miculan

Scenario: information is distributed

- Authorized people can send, retrieve, modify information from a distance
- Physical restriction not possible anymore
- Security threats not only on the computer where information is stored, but also on the channels used for transmissions



es. Cisco aveva dei router con dentro chip dell FBI che sniffavano i dati di certa gente

How to deal with the "security problem" in general?

- General strategy for security
  1. Identify **assets** to protect (↳ qual è l'oggetto che dobbiamo proteggere? identifichiamolo)
  2. Setting **security goals** (↳ obiettivi da rispettare sugli ASSET)
  3. Establishing **security policy** (↳ politiche/regole per garantire i goal)
  4. Identify **threats** (↳ minaccia)
  5. Develop **security services** to implement **controls/ countermeasures** and **disaster/recovery plan** (↳ servizi di sicurezza)

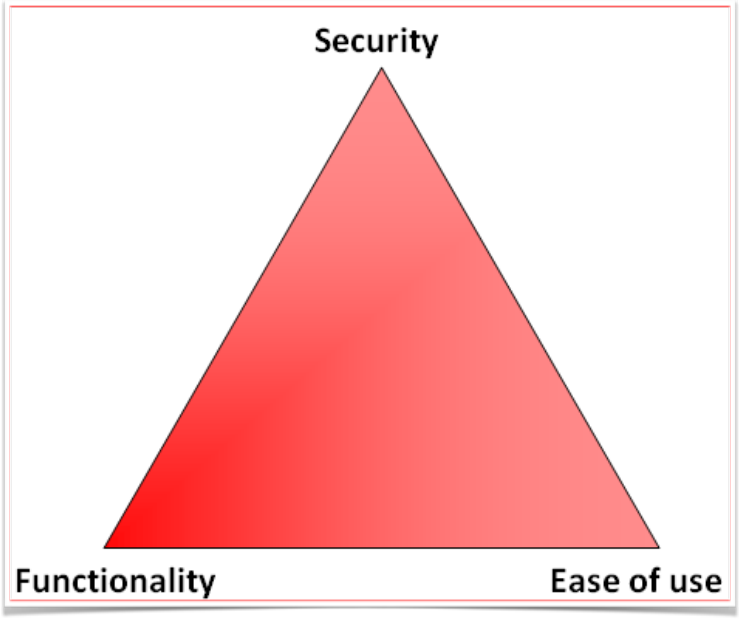
Security is always a cost, but insecurity costs more

- Security is obtained by implementing **services** which come at a cost
  - economical
  - computational resource
  - human resources for deployment and management
  - more complex program design and implementation
  - practical hassles to users
- But lack of security may cost much more
  - loss of vital assets (informations, data, etc)
  - legal and penal consequences

es. ANTIVIRUS si paga: - soldi
- memoria per avviarlo + cpu per usarlo
- traffico di rete per gli aggiornamento

Security is always a trade-off…

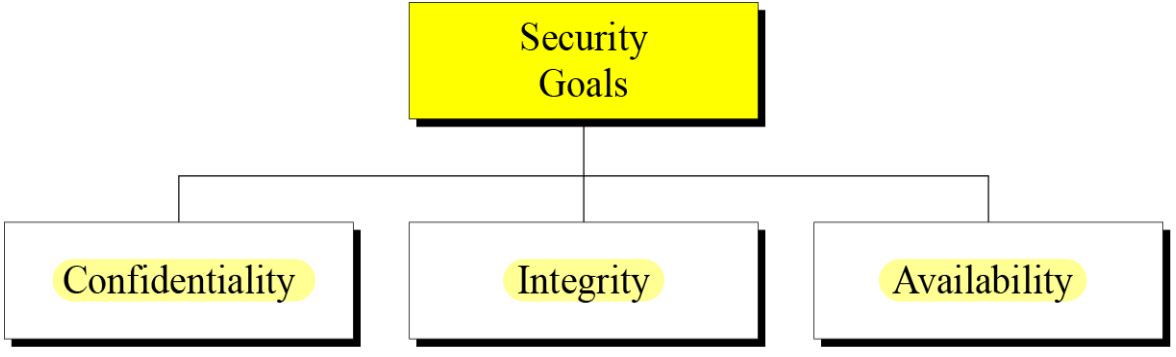
- …w.r.t. functionality and usability
  - We cannot maximise all three aspects at once
  - enhancing an aspect decreases the others
  - which ones are more important and which one can be sacrificed, depends on the situation



sistema HARDENIZZATO: sistema che ha solo le funzionalità essenziali, infatti meno funzioni fa più è sicuro

Goals of Information Security

- Many models
- Quite common: the CIA triad



Sono 3 aspetti indipendenti

Security Goals

- **Confidentiality**: we need to protect our confidential information. An organization needs to guard against those malicious actions that endanger the confidentiality of its information. ⟶ tra i router i pacchetti viaggiano in chiaro → va presa una misura di precauzione
- **Integrity**: Information needs to be changed constantly. Changes need to be done only by authorized entities and through authorized mechanisms.
- **Availability**: The information created and stored by an organization needs to be available to authorized entities. quando me hanno ca necessita' Information needs to be constantly changed, => it must be accessible to authorized entities.

Security threat, attack, service, mechanism

- **Security Threat**: A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security goals and cause harm.
- **Security Attack**: An assault on system security that derives from an intelligent threat, i.e., any action that compromises the security goals of information of an organization
- **Security Service**: a service for enhancing the security of the system and information transfers. Intended to counter security attacks, using security mechanisms.
- **Security Mechanism**: a mechanism that is designed to detect, prevent, or recover from a security attack.

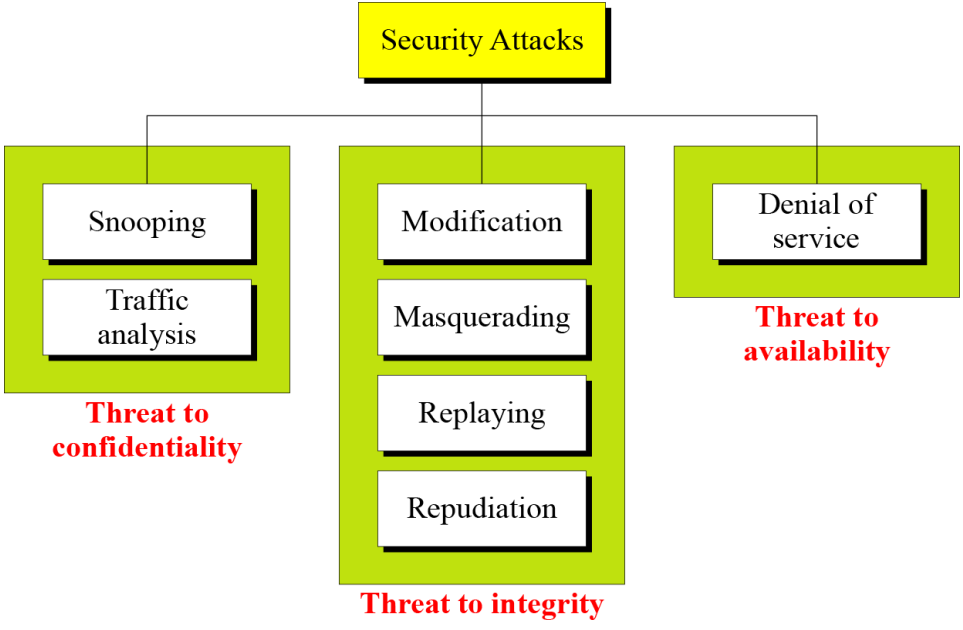
MINACCIA
es. modificare/sniffare il traffico del router. Potrebbe succedere ma non è detto che succeda
ATTACCO: verificarsi di un'azione, metto in atto la minaccia
↳ singole funzionalità con cui posso costruire servizi di sicurezza
↳ es. usare un LOG o BACKUP
↳ es. per capire se qualcuno sta tentando di attaccare la mia password uso il meccanismo di detect: tengo un counter di quante volte è stata inserita sbagliata

es. della password:
- THREAT: voglio indovinare/trovare la password
- ATTACK: provo ad inserire la password
- SERVICE: riconoscere il brute-force
- MECCANISMO: tengo il count delle password sbagliate

Security attack, service, mechanism: an example

- In my computer there are the text of next exams (**assets**)
- Some people are interested in accessing them (**threat**)
- Sometimes I walk away leaving the computer alone (**vulnerability**)
- While I am away, an unauthorized person may sit and use my computer (**attack**)
- Countermeasure: implement an ***authentication*** **service**, which asks and guarantees the identity of the users of the computer
- Some **mechanisms** for implementing the authentication service:
  - Username/password
  - Fingerprint
  - Smartcard
  - Voice tone
  - Retinal scan
  - Security guard

Attacks to security goals in network communications

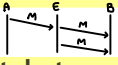


Attacks to Confidentiality

- **Snooping:** unauthorized access to data
  - Unauthorized access to computer (hacker, virus, password misuse, social engineering, etc)
  - Data interception during transmissions (sniffing)
    - Use encipherment!
- **Traffic analysis:** obtaining some other type of information by monitoring online traffic
  - There are many ways to divulge partial information, even if data is encrypted (e.g. who do you talk to)

Attacks to Integrity

- **Modification** means that the attacker intercepts the message and changes it.
  - Sometimes the attacker simply deletes or delays the message
  - Usually follows a snooping attack
- **Masquerading or spoofing** happens when the attacker impersonates somebody else
  - Attacker may pretend to be the sender entity, or the receiver entity, or both ("man-in-the-middle")
- **Replaying** means the attacker obtains a copy of a message sent by a user and later tries to replay it (without modifications)
- **Repudiation**: the sender of the message might later deny that she has sent the message; the receiver of the message might later deny that he has received the message.
  - notice that this attack is performed by one of the two legitimate parties, and not by a third entity



Attacks to Availability

- **Denial of service (DoS)** is a very common attack. It may slow down or totally interrupt the service of a system.
  - Generate bogus requests to generate heavy load
  - Delete server's responses to force client the server is not responding
  - Delete client's requests, forcing it to repeat requests
  - Can be implemented by a coordinated network of attackers (distributed DoS, DDOS)
  - Countermeasure
    - make attacker's load no less than target's
    - Divert illicit requests, if possible

↳ se performato da più parti contemporaneamente

il DoS è facile attuarlo con TCP: durante l'Handshaking basta continuare a mandare SYN (da CLIENT a SRV) e non rispondere più.
(risolto da SCTP)

Passive vs Active Attacks

↳ l'attacco NON ALTERA il funzionamento del sistema ⟶ non si possono RILEVARE

- **Passive** attacks: learn or make use of information from the systems without affecting system resources
- **Active** attacks: alter system resources or their operations

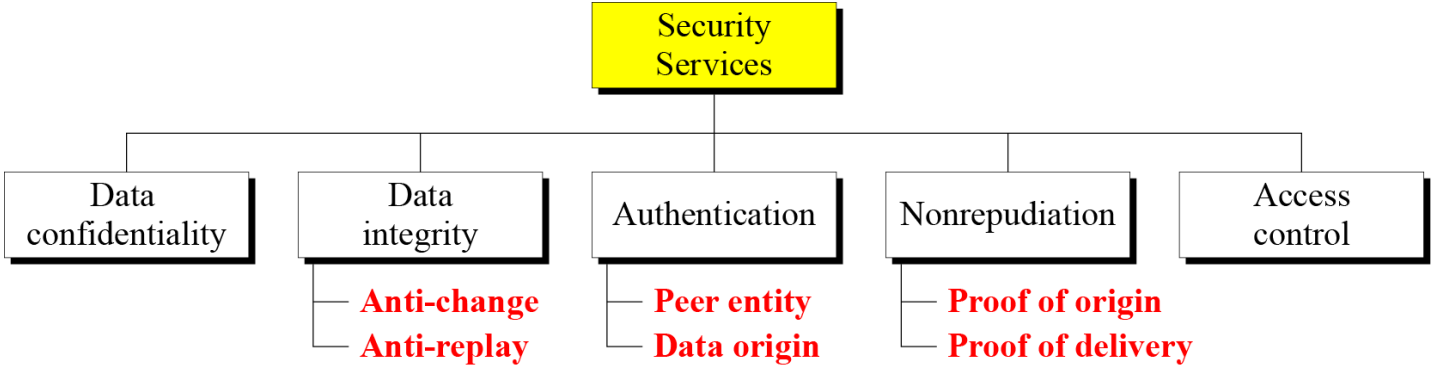
Attacks	*Passive/Active*	*Threatening* (minacciamo)
Snooping Traffic analysis	Passive	Confidentiality
Modification Masquerading Replaying Repudiation	Active	Integrity
Denial of service	Active	Availability

Approaches to passive and active attacks

- Passive attacks do not change data nor affect the system
  - Difficult to detect
  - Emphasis is on prevention, rather on detection
- Active attacks change data or affect the system
  - Difficult to prevent completely
  - Easier to detect (and apply suitable counter-measures)

gli ATTACCHI ATTIVI all'integrità devono convincere il ricevente che non sia successo niente, non deve accorgersene, semmò l'attacco fallisce

Security Services (according to ITU X.800)



- (Availability - assurance that a resource is available for the intended uses - is seen as a property, not a service!)

Security Services

- **Authentication**: assurance that the communicating entity is the one claimed
  - Data origin authentication - easier
  - Peer entity authentication - at connection initiation and during connection
- **Access control**: prevention of the unauthorized use (read, write, modify, execute, etc) of a resource
  - very broad meaning, but Denial of Service does not fall in this category
- **Data Confidentiality:** protection of data and meta-data (such as traffic flow) from unauthorized disclosure
- **Data Integrity**: assurance that data received is as sent by an authorized entity

Security Services (cont.)

- **Non-Repudiation** - protection against denial by one of the parties in a communication
  - **Origin**: proof that the message was sent by the specified party
  - **Destination**: proof that the message was received by the specified party
- **Availability** - assurance that a resource is available for the intended uses
  - Defined to be a property of other services, not a service on its own

Security Mechanism

- basic feature designed to detect, prevent, or recover from a security attack
- no single mechanism that will support all services required
- often replicates functions normally associated with physical documents
  - such as signatures, dates;
  - protection from disclosure, tampering, or destruction;
  - notarized or witnessed;
  - recorded or licensed
  - ...

Security Mechanisms (X.800): specific security mechanisms

- May be incorporated into the appropriate protocol layer in order to provide some service
- **Encipherment**: hiding or covering data
  - can provide confidentiality, but also other services
  - Cryptography and steganography
- **Data integrity**: detect data changes
  - e.g. checksum values, hash
- **Digital signatures**: guarantees receiver about message origin

Security Mechanisms (X.800): specific security mechanisms

- **Authentication exchange**: the parties prove their identity to each other (es. con parola d'ordine)
- **Traffic Padding**: insert bogus data into real data traffic, to avoid traffic analysis
- **Routing Control**: selecting and changing traffic routes, in order to avoid interceptions
- **Notarization**: a third trusted party controls the communications (e.g. to prevent repudiation)
- **Access control**: methods to prove that an entity has access rights to resources

Relationship between services and mechanisms

Service \ Mechanism	Enciph-erment	Digital signature	Access control	Data integrity	Authenti-cation exchange	Traffic padding	Routing control	Notari-zation
Peer entity authentication	Y	Y			Y			
Data origin authentication	Y	Y						
Access control			Y					
Confidentiality	Y						Y	
Traffic flow confidentiality	Y					Y	Y	
Data integrity	Y	Y		Y				
Non-repudiation		Y		Y				Y
Availability				Y	Y			

riconoscimento

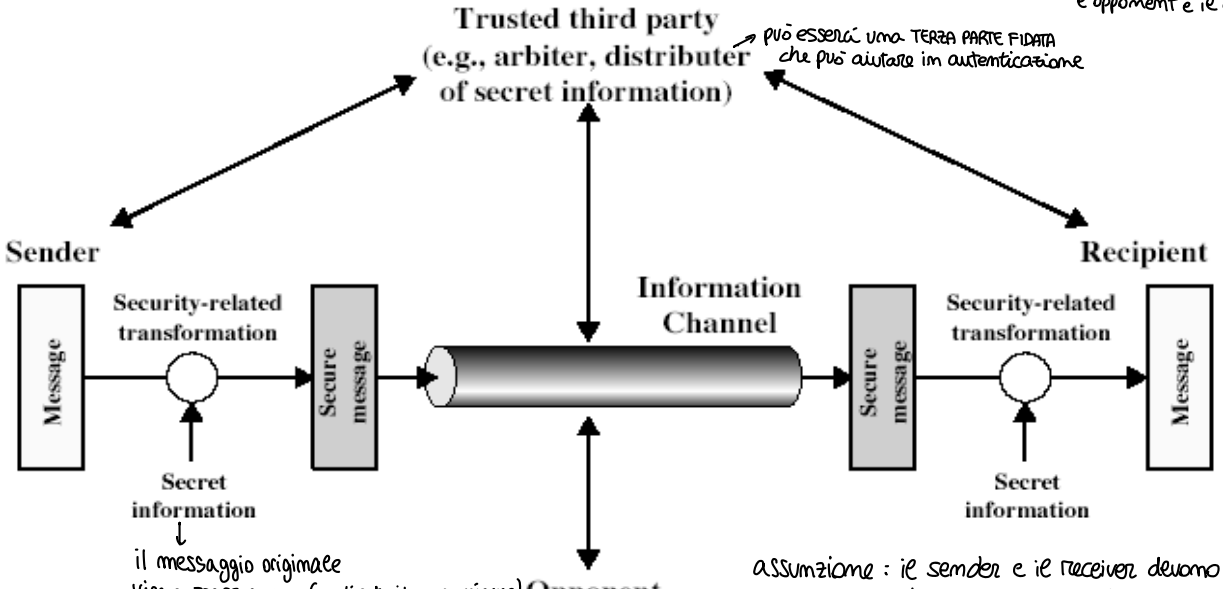
- Not all mechanisms can be used for implementing a given service
- Mechanisms discussed before are only theoretical recipes to implement security
- Actual implementation of security services needs some techniques.

Models for network security

- A model is an *abstraction* for considering the threats and *design* security services by deploying security mechanisms and deciding strategies
- Two main models: ↱ 2 modelli per progettare servizi di sicurezza
  - The **model of insecure channel** ↱ sicurezza di rete, for reasoning about security of data transmitted on an insecure network
  - The **model of network access** ↱ sicurezza di perimetro, for reasoning about threats of our information system
- We will see only the first one

Model for Network Security (or the Insecure channel) (Dolev-Yao)

canale insicuro, è il modello della rete Internet. Chiunque potrebbe attaccarci nel mezzo. → questo può fare tutto in mezzo al canale, tant'è che si dice che l'oppoment è il canale



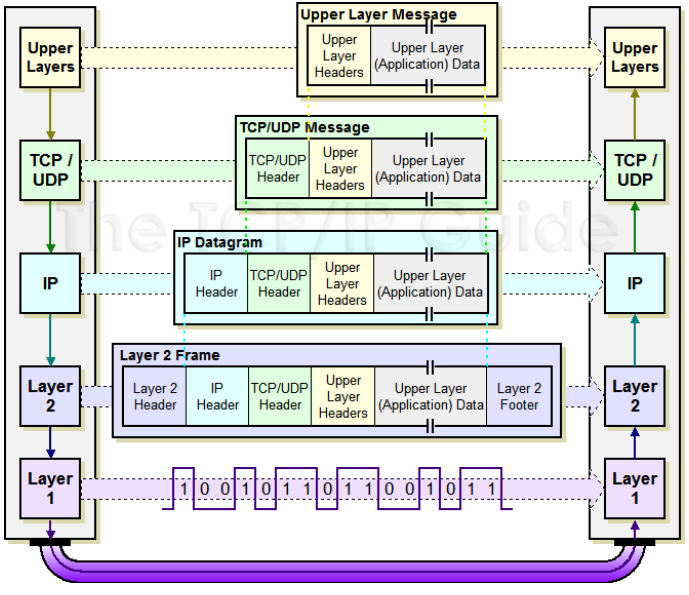
Model for Network Security

- using this model requires us to:
  - design a suitable algorithm for the security transformation
  - generate the secret information (keys) used by the algorithm
  - develop methods to distribute and share the secret information
  - specify a protocol enabling the principals to use the transformation and secret information for a security service

Where to implement security services?

- Each level of the OSI or TCP/IP model implements a (virtual) channel
- Security services can be implemented on (almost) any of these channels
- Results vary

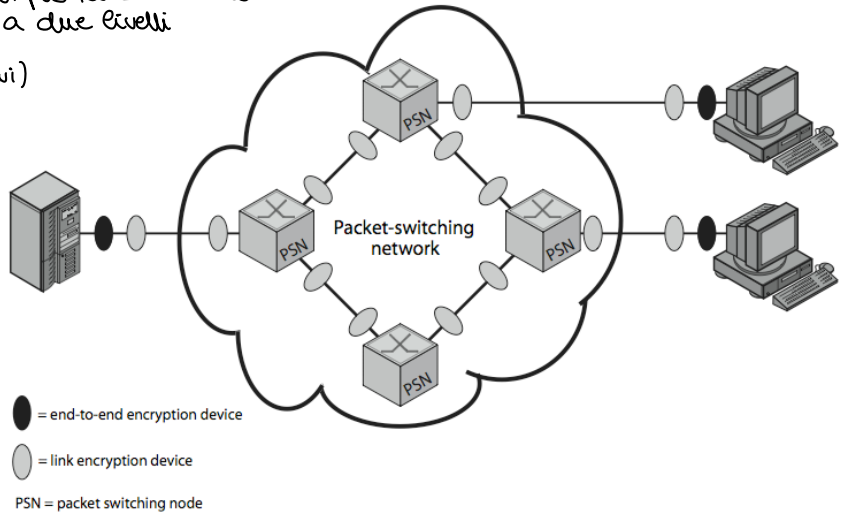
qualsiasi di questi canali può essere oggetto di attacchi



Placement of Security

- have two major placement alternatives
- **link encryption** : metto al sicuro il singolo link (mezzo che collega due dispositivi)
  - encryption occurs independently on every link → es. WiFi con la password è un sistema di sicurezza a LIVELLO DI LINK
  - implies must decrypt traffic between links
  - requires many devices, but paired keys
- **end-to-end encryption** (a liv. 3-4-6-7)
  - encryption occurs between original source and final destination
  - need devices at each end with shared keys → es. SSH, PGP

si può fare la sicurezza a due livelli



la trasformazione (a livello LINK) vale sul collegamento (linee) ma quando arriva sul router il pacchetto è in chiaro
per proteggere anche i router dobbiamo usare e'end to end

Placement of Security

- when using end-to-end encryption must leave headers in clear
  - so network can correctly route information
- hence although contents protected, traffic pattern flows are not
- ideally want both at once
  - end-to-end protects data contents over entire path and provides authentication
  - link protects traffic flows from monitoring

Placement of Security

- can place security function at various layers in OSI Reference Model
  - link encryption occurs at layers 1 (physical) or 2 (data link)
  - end-to-end can occur at layers 3 (network, per system), 4 (transport, per process), 6 (presentation), 7 (application)
- as move higher **less information is secured** but **it is more secure** though more complex with more entities and keys
  - Number of keys needed in a network is $O(n^2)$, where n is the number of parties
  - At the network level, a key for each pair of hosts
  - At the application level, a key for each pair of applications

↳ più implementiamo la sicurezza a liv. alto ↳ copriamo meno informazioni ma in maniera robusta

↳ in ogni livello c'è una coppia (se ci sono 2 partecipanti), in generale n chiavi per livello, con n host

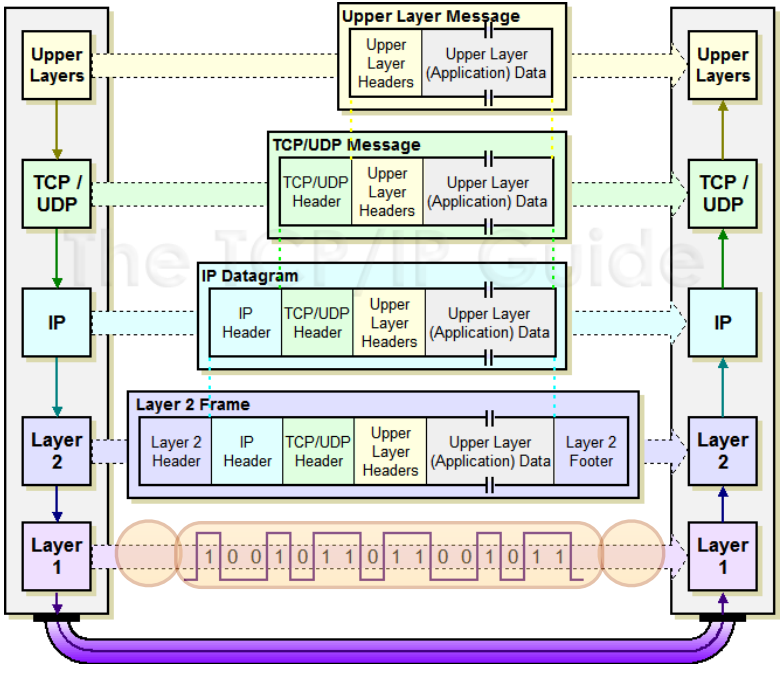
Placement of Security

- At lower levels: (↳ più è a basso livello più è trasparente per l'utente)
  - more information are protected by the service
  - more transparent for the user/programmer
  - more complex and expensive (sometimes impossible) to implement effectively
  - vulnerable in bridges (↳ nei ROUTER i pacchetti sono in chiaro)
- At higher levels:
  - less information are protected, but better
  - more intrusive for the user/programmer
  - needs more keys

Security at physical layer

- Link-wide secure communications
- Implemented by sensors to detect physical intrusions on the medium (seals, infrared barriers, pressure sensors…)
- Managed by telcos
- Covers *everything*
- Expensive: each link requires its own security
- Not always possible (e.g. wireless links)

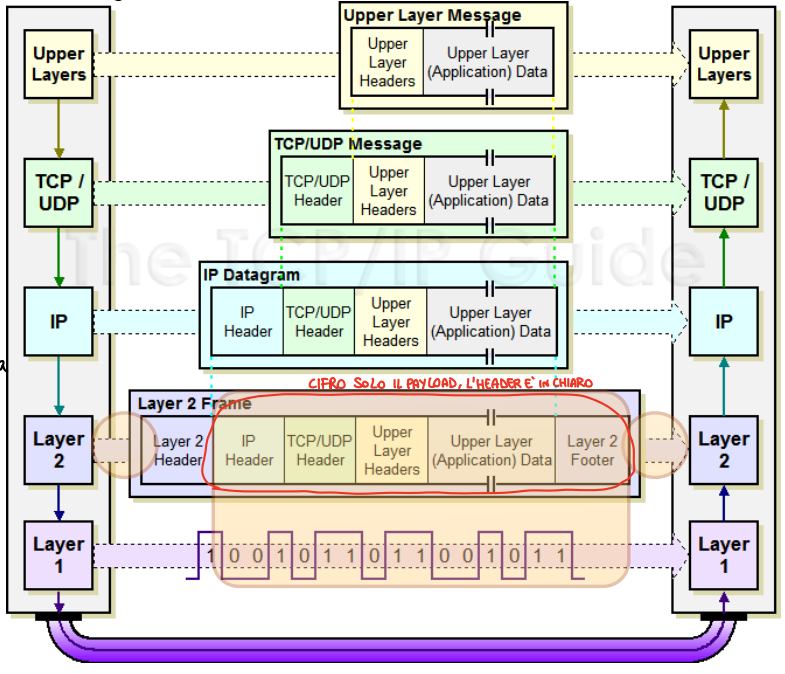
limito l'accesso al mezzo fisico (es. inserisco i cavi dentro dei tubi a pressione che hanno dei sensori, se provi ad aprire il tubo esce l'aria ed i sensori lo avvertono ed interrompono la comunicaz.)



Security at datalink layer

All' interno dell' interfaccia di rete, es. scheda Ethernet

- Link-wide secure communications
- Implemented in network cards (e.g. WEP, WPA)
- Managed by sysadmin
- Covers application data, transport, network header → copre il payload ma non l'header di liv. 2 semmò non arriva dall'altra parte
- Vulnerable in network bridges (routers)

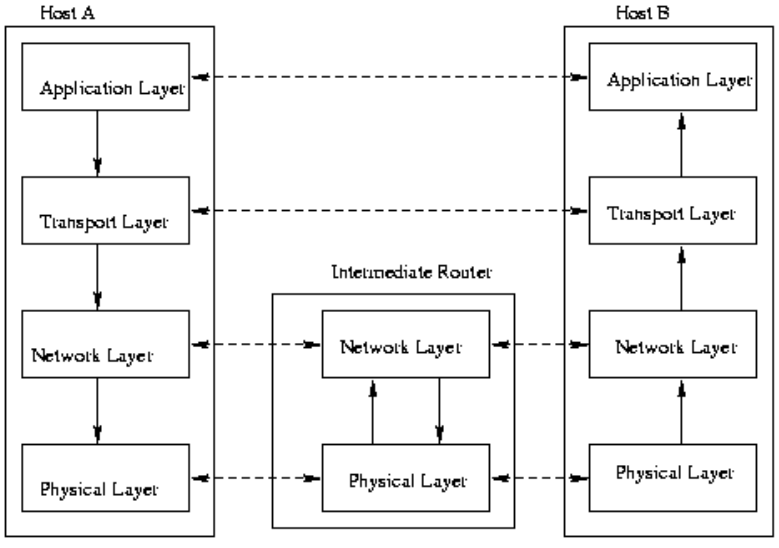


es. della vulnerabilità di Alexa ...

capisce se c'è qualcuno in casa o no in base alla frequenza dei pacchetti inviati

Security at datalink layer

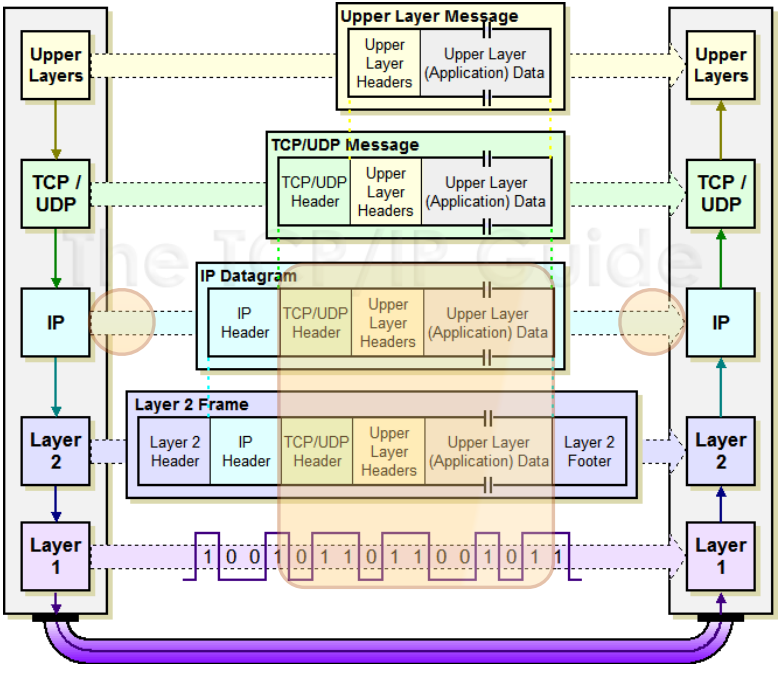
- Datalink security is vulnerable if traffic has to cross routers: these have to decrypt the messages in order to get routing informations, which can be thus intercepted by intruders



Security at network layer

- System-wide secure communications (not ad hoc for applications)
- Implemented in kernel
- Managed by sysadmin; transparent for developers
- Covers application data and transport header → così possono passare per i router senza dover essere decifrati, lascio in chiaro soltanto gli indirizzi
- Example: IPsec, a IP integration which can implement both authentication and confidentiality
- VPNs implemented here

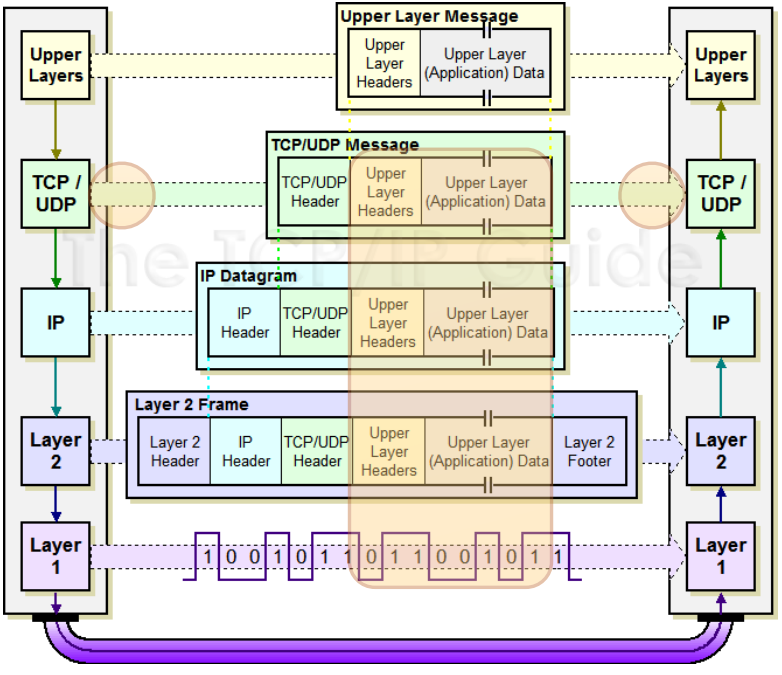
i provider vedono tutti i siti che visitiamo (vedono ip destinatario) ma non vedono il contenuto del traffico



Security at presentation/transport layer

- End-to-end secure communications
- Implemented as libraries or system's APIs
- Programmers need to use explicitly the service, but not to program it
- Covers application data and header (if any)
- Example: TLS/SSL, used for "securing" many existing connection-oriented protocols (HTTP->HTTPS, POP->POPS, IMAP->IMAPS...)

↳ PROTOCOLLI UGUALI che però USANO delle socket che cifrano tutto tranne gli indirizzi (quindi non vedi il payload)

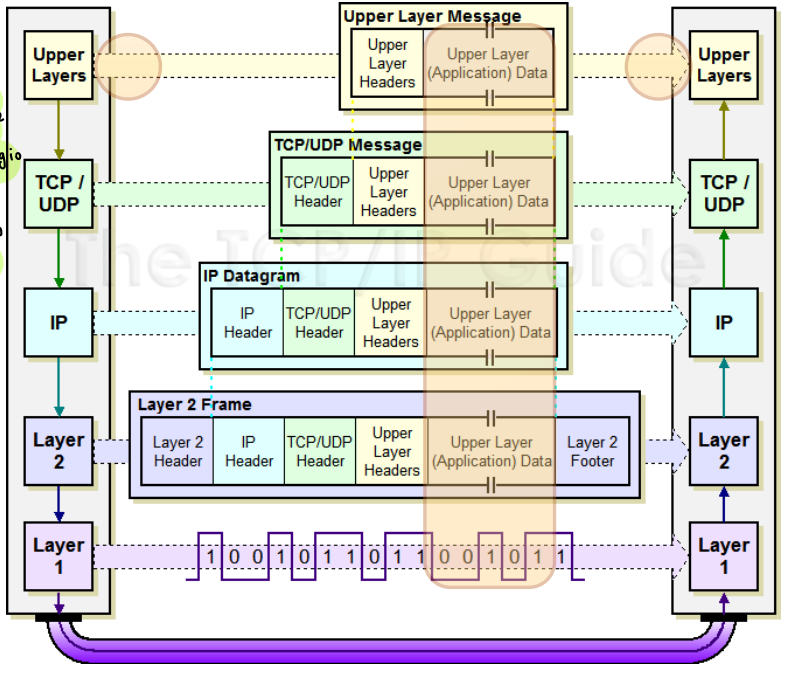


Security at application layer

- Implemented directly in applications
- Covers only application data
- Many "ad hoc" security protocols:
  - SSH for remote connections
  - PGP, S/MIME, PEC for email
  - SET/EMV for credit-card transactions
  - Kerberos, OAuth2 for authentication
  - Signal, MTProto2 for chats, etc.

quindi l'header del livello 7 è in chiaro, es. in una mail il mittente e il destinatario sono visibili ma il messaggio no.

Telegram/Whatsapp uguale, si sanno mittente e destinatario data, ora e dimensione ma non il messaggio



Security services & protocols vs ISO OSI architecture

Level		Services	Protocols/applications
7	Application	End-to-end services	PGP, S/MIME, SAML, SSH, EMV, SET, ...
6	Presentation		XML sicuro
5	Session	Authentication encryption traffic analysis…	SSL/TLS, Secure UDP
4	Transport		
3	Network	Authentication, encryption	IPSec
2	Datalink	Authentication for link access, confidentiality	WEP, WPA, 802.1X
1	Physical		

Encryption

ma non è l'unico meccanismo di sicurezza

- Encryption is a fundamental security mechanism, used in many services
- Two main kinds of encryption
  - Symmetric, or “private key”, encryption
  - Asymmetric, or “public key”, encryption
- Not alternative, but two different tools with different applications

Some Basic Terminology

- **plaintext** - original message
- **ciphertext** - coded message (cifrato che è ≠ ciphrato)
- **cipher** - algorithm for transforming plaintext to ciphertext (cifrario)
- **key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to ciphertext (cifrare)
- **decipher (decrypt)** - recovering ciphertext from plaintext (decifrare)
- **cryptography** - study of encryption principles/methods (CRITTOGRAFO è quello che inventa gli algo)
- **cryptanalysis (codebreaking)** - study of principles/ methods of deciphering ciphertext *without* knowing key (cerca di scoprire il testo in chiaro o la chiave)
- **cryptology** - field of both cryptography and cryptanalysis (CRITTOLOGIA)

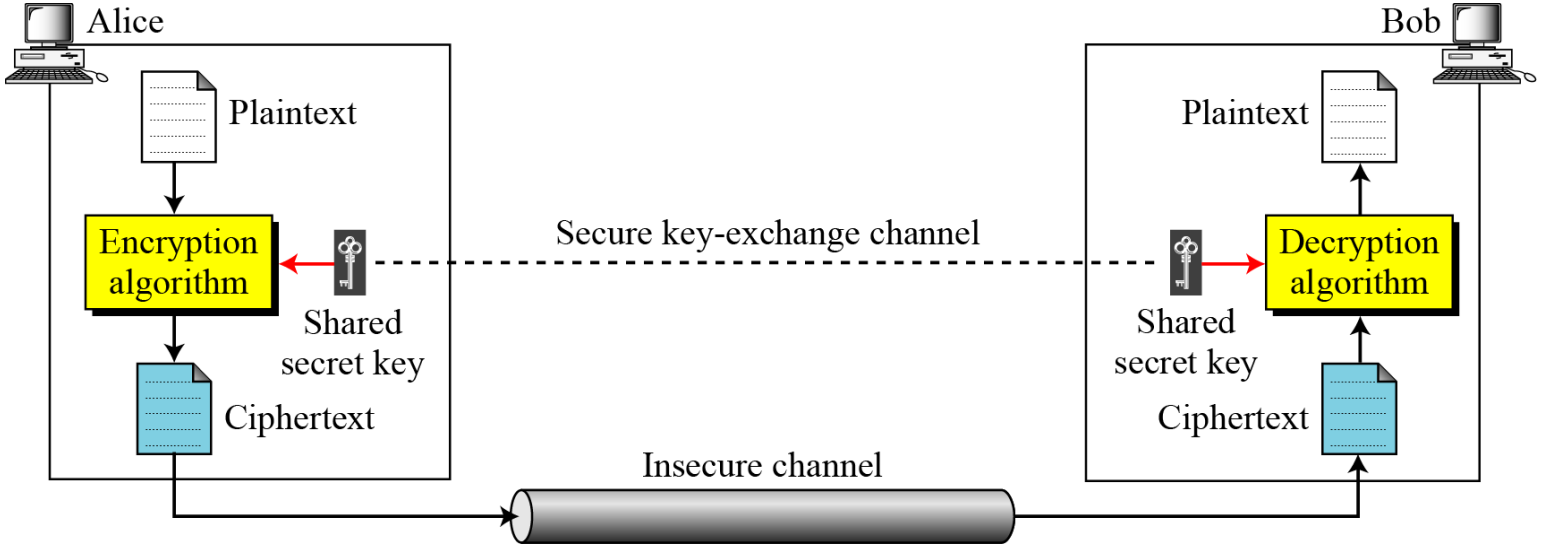
Symmetric Encryption

- or conventional / private-key / single-key
- sender and recipient share a common key, secret to the intruder
- all classical encryption algorithms are private-key
- was only type prior to invention of public-key in 1970's
- and by far most widely used
  - **not** substituted by asymmetric (I.e. public key) encryption

↱ sender e receiver hanno le stesse conoscenze

È più veloce ed efficiente e fa il lavoro sporco

Symmetric Cipher Model



Requirements

- two requirements for secure use of symmetric encryption:
  - a strong encryption algorithm: without knowing the key, it must be impossible to get the plaintext from the ciphertext.
  - a secret key known only to sender / receiver
- mathematically we have two functions, such that:

$Y = E_K(X)$ encryption (Y: ciphertext)

$X = D_K(Y)$ decryption (X: plaintext)

$D_K(E_K(X)) = X$

$D_K(E_H(X)) = error$ if $K \neq H$

il CRITTOGRAFO deve inventare D,E

l'attaccante può avere tutto tranne la CHIAVE (e il PLAIN TEXT), ha anche gli algoritmi

Requirements: Kerckhoff's principle

- We must assume encryption algorithm is known by the opponent
  - **Kerckhoff's principle** (*La cryptographie Militaire*, 1883): security of the message must depend only the secrecy of the key, not on the secrecy of the encryption algorithm → assumere che l'algoritmo sia pubblico
  - Aka *Shannon's maxim*:
    "The enemy knows the system."
- The only information the opponent is missing is the key; everything else is known → anche perché se l'attaccante ci dovesse sgammare la chiave, basta cambiarla (veloce) e non leggerà più niente
- (implies a secure channel to distribute key)

↳ se l'algo è segreto e l'attaccante lo scopre, cambiarlo è LENTO → tanto vale lasciarlo pubblico
se la CHIAVE " " " " è VELOCE

Requirements: Kerckhoff's principle

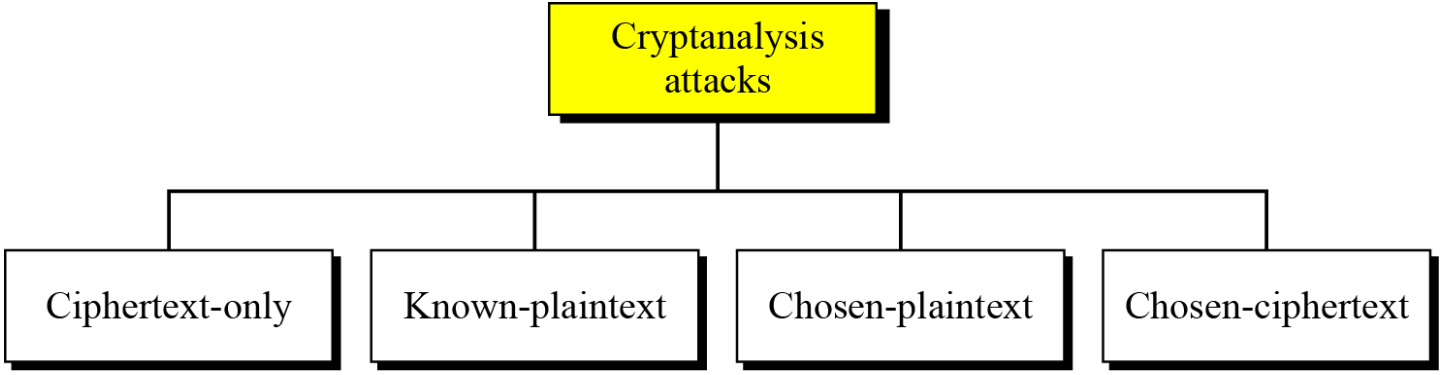
- One can keep secret the source code and the cipher (although arguable…), but this cannot be a requisite for the security

↳ se un SW ti dice che è sicuro perché nessuno sa il codice è fuffa, probabilmente ci ha inserito una backdoor. ⟶ gli algoritmi devono essere pubblici, conta solo la CHIAVE.

- Any security software design that doesn't assume the enemy possesses the source code is already untrustworthy
- **never trust closed source encryption algorithms and protocols**

Cryptanalysis

- **Cryptanalysis**: the process of attempting to discover the plaintext corresponding to a ciphertext, or better the key
- Many attacks, depending on what cryptoanalyst knows/can do



Cryptanalytic Attacks

- Increasing information for the cryptanalist:
- **ciphertext only** (es. mi metto sul router)
  - Cryptanalist only knows algorithm & ciphertext
- **known plaintext**
  - know/suspect one or more plaintext & corresponding ciphertext; try to recover the key

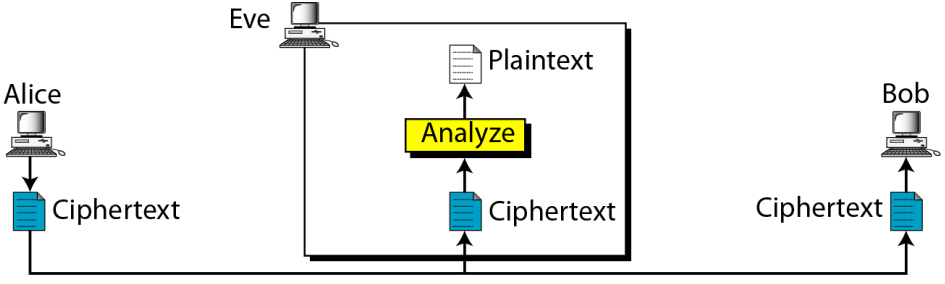
(es. l'intestazione (header) che è in chiaro)

- **chosen plaintext** → l'attaccante riesce a fare in modo che il messaggio da cifrare sia quello che vuole lui forzando il mittente ad inviare uno specifico messaggio con lo scopo di analizzare il risultato cifrato ed ottenere la chiave
  - select plaintext and obtain ciphertext
- **chosen ciphertext**
  - select ciphertext and obtain plaintext
- **chosen text**
  - select plaintext or ciphertext to en/decrypt
- Good algorithms are designed to withstand a known-plaintext attack, at least.

Ciphertext-Only Attack

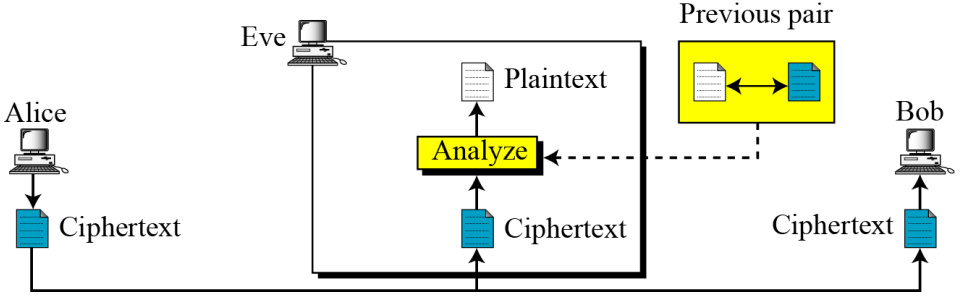
- Cryptanalist only knows algorithm & ciphertext
- Aim: discover plaintext and/or key

le chiavi appartengono ad un insieme finito → provo tutte le combinazioni possibili (BRUTE FORCE)



Known-Plaintext Attack

- Cryptanalist knows/suspects one or more plaintext & corresponding ciphertext
- Aim: discover new plaintexts and the key



- Good algorithms are designed to withstand a known-plaintext attack, at least.

Which security can we get from encryption?

- **unconditional security** → qualunque sia la potenza di calcolo non posso trovare la chiave → raggiungibile solo con un cifrario (non utilizzato)
  - no matter how much computer power or time is available, the cipher cannot be broken since the ciphertext provides insufficient information to uniquely determine the corresponding plaintext
- **computational security**
  - The cost of breaking the cipher exceeds the value of the encrypted information; or
  - given limited computing (↳ ci vuole troppo tempo) resources (e.g. time needed for calculations is too large), the cipher cannot be broken

Brute Force Search

devo rendere il costo di questo attacco irraggiungibile → ci vuole troppo tempo ed energie

- always possible: simply try all possible keys, until the correct one is found
  - requires to recognize when the key is wrong or correct, that is: $D_{K'}(E_K(M)) = \text{error}$ iff $K' \neq K$
  - assume either know / recognisable plaintext
- most basic attack, proportional to key size; becomes quickly unfeasible

Key Size (bits)	Number of Alternative Keys	Time required at 1 decryption/μs (con un PC MEDIO → 1 MILIONE di TENTATIVI al secondo)	**Time required at 10^6 decryptions/μs** (con un computer del governo → 10^{12} tentativi al secondo)
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\,\mu s = 35.8$ minutes	2.15 milliseconds
56 (es. DES lo ha)	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\,\mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\,\mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\,\mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 diff. chars (perms) (↳ permutazione delle lettere dell'alfabeto)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\,\mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

Age of the universe = 13.7×10^9 years ⟹ tutto è possibile, ma ci vuole molto tempo

Cryptanalytic attacks

- Thus, brute-force is always possible in theory, but not in practice: its computational cost is too high
- Other strategy: **Cryptanalytic attack**
  - look for mathematical weakness in the cypher, possibly for specific keys, texts
  - Can be useful to reduce the key space where to apply brute-force attacks
  - Some known and used ciphers do have cryptanalytic attacks, but not practical
    - e.g. reducing the key space from 2^{128} to 2^{126} is considered an attack, but in practice it does not weaken the cipher

→ analizzo il caso per escludere dei caratteri

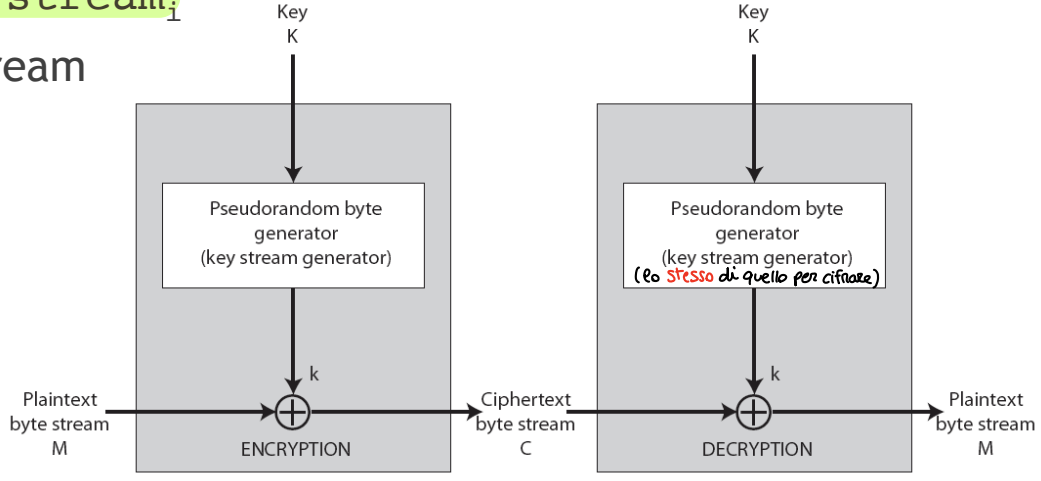
Stream Ciphers

due famiglie di cifrari simmetrici: - a blocco
- a flusso (considerano byte per byte)

- process message byte by byte (as a stream)
- have a pseudo random **keystream**
  ↳ flusso usato come CHIAVE
- combined (XOR) with plaintext bit by bit
  - $C_i = M_i \text{ XOR } Keystream_i$
- randomness of keystream completely destroys statistical properties in message

Messaggio PLAINTEXT M che entra come flusso e fa XOR con un flusso pseudo casuale

$Ci \oplus KSi = (M \oplus Ksi) \oplus ksi$



Stream Cipher Properties

↳ se abbiamo due messaggi cifrati con lo stesso keystream

- Must **never** reuse keystream, otherwise:
  - $C \text{ XOR } C' = (M \text{ XOR } KS) \text{ XOR } (M' \text{ XOR } KS) = M \text{ XOR } M'$
    (ciphertext 1, ciphertext 2, plaintext 1, keystream; C, C')
- some design considerations are:
  - must have long period with no repetitions
  - Keystream statistically random
  - depends on large enough key
  - large linear complexity
- properly designed, can be as secure as a block cipher with same size key, but usually simpler & faster
- A very common implementation is RC4
  - widely used (web SSL/TLS, wireless WEP, WPA)

il flusso NON DEVE RIPETERSI MAI (neanche in sessioni diverse)
↳ dopo un po' per forza si ripeterà (è deterministico)
quindi è un problema costruire algo tali che data una chiave generino stream lunghi senza ripetersi.
es. RC4 ripete la sequenza dopo circa 2^{100} passi (ha 100 bit di flusso)

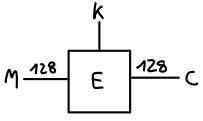
ammucco il "rumore" introdotto dal keystream

Con One-Time Pad uso un keystream random ed è efficace ma bisogna condividere in anticipo il keystream sul canale sicuro (ma dato che |keystream| = |messaggio|, tanto vale scambiarsi il messaggio)
Una volta usato va scartato (One Time Pad) altrimenti non sarebbe più random

Block Ciphers

non macinano a bit/byte alla volta, consumano a BLOCCHI

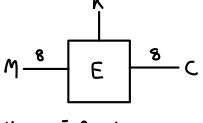
- block ciphers process messages in blocks, each of which is then en/decrypted
- like a substitution on very big characters
  - 64-bits or more (modern ciphers work with 128 bits)
  - The larger the block, the less effective statistical attacks are
- many current ciphers are block ciphers → i più utilizzati
- Well analysed, broader range of applications



$E_k : \{0,1\}^{128} \to \{0,1\}^{128}$ (è biettiva, invertibile)

non faccio bit a bit,
MAPPO 128 bit in altri 128 bit

es. Blocco a 8 bit
⇓
vuol dire una lettera alla volta.
↳ es. nella lingua italiana è facilmente rompibile con un'analisi statistica, infatti la 'E' è il carattere mediamente più presente e ad es. se viene mappato in 'z' (che è sicuramente meno usata) avrei tante occorrenze di 'z' e quindi dedurrei facilmente che si tratta di 'E'.



uso piuttosto 24 bit → 3 caratteri alla volta

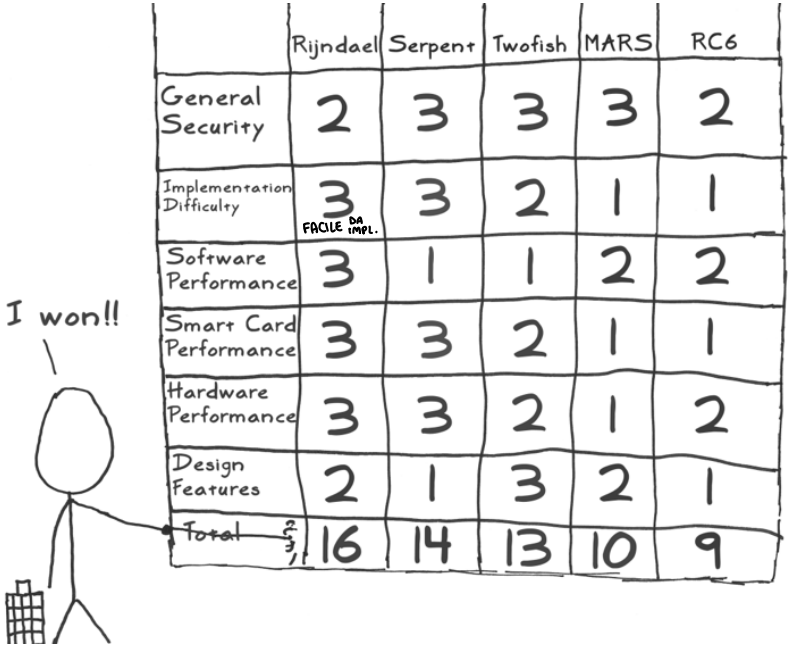
Block Cipher Principles

- block ciphers look like an extremely large substitution
- would need table of 2^{64} entries for a 64-bit block
  - Memory usage: $2^{64} \times 2^{6} = 2^{70}$ bytes — unfeasible
- Instead, create substitution from smaller building blocks
- Many classical symmetric block ciphers are based on a Feistel Cipher Structure (DES)
  ↳ a 64 bit con 56 bit di chiave → non era troppo sicuro
- More modern ciphers are different: AES

AES - Advanced Encryption Standard — Origins

- US NIST issued call for ciphers in 1997. Requirements:
  - private key symmetric block cipher
  - 128-bit data, 128/192/256-bit keys
  - stronger & faster than Triple-DES
  - active life of 20-30 years (+ archival use)
  - provide full specification & design details → era tutto pubblico
  - both C & Java implementations
- 15 candidates accepted Jun 98, 5 shortlisted Aug-99
- Rijndael was selected as the AES in Oct-2000 (= AES)
- issued as FIPS PUB 197 standard in Nov-2001
- Being used in many contexts by now, replacing 3DES

Bluray, Digitale terrestre, SSL, SSH usano AES

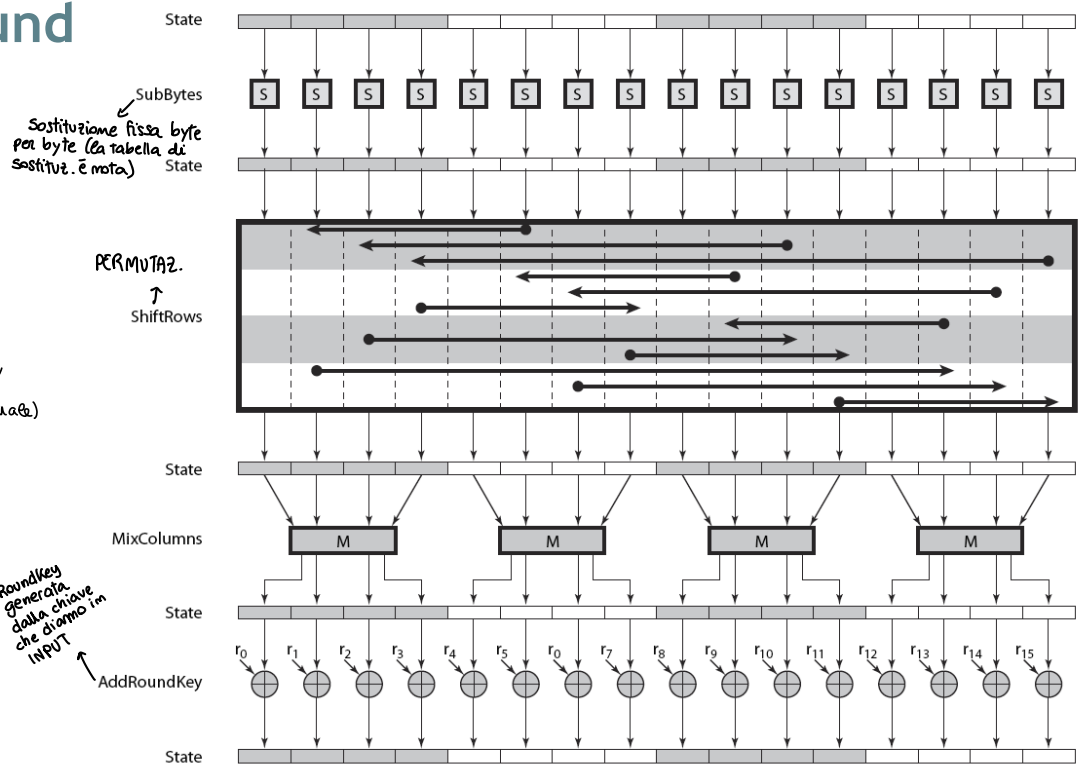


See the funny http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html

The AES Cipher - Rijndael

- designed by Rijmen and Daemen in Belgium
- has 128/192/256 bit keys, 128 bit data
- an **iterative** rather than Feistel cipher
  - processes data as block of 4 columns of 4 bytes
  - operates on entire data block in every round
- designed to be:
  - resistant against known attacks
  - speed and code compactness on many CPUs
  - design simplicity

AES Round


State
SubBytes
Sostituzione fissa byte per byte (la tabella di sostituz. è nota)
State
PERMUTAZ.
ShiftRows
la struttura è sempre la stessa, cambia solo la ROUND KEY ad ogni esecuzione (il resto è uguale)
State
MixColumns
M
M
M
M
State
Roundkey generata dalla chiave che diamo in INPUT
AddRoundKey
State

AES security

- Up to now, the only successful attacks are side channel attacks
- Allowed by US government for SECRET and TOP SECRET documents
- Deeply studied by many cryptographers by now
- Most recent attacks are still unfeasible (e.g. Biryukov & Khovratovich related key attack, has a complexity of $2^{99.5}$).

↓
riesci a ridurre la complessità da 2^{128} a $2^{99,5}$ quando usi chiavi molto simili

↳ ae posto che 2^{128} però è comunque irraggiungibile dal punto di vista pratico

AES può essere utilizzato in tutti i livelli (es. WPA2 usa AES, IPsec usa AES, SSL usa AES)

Modes of Operation of block ciphers

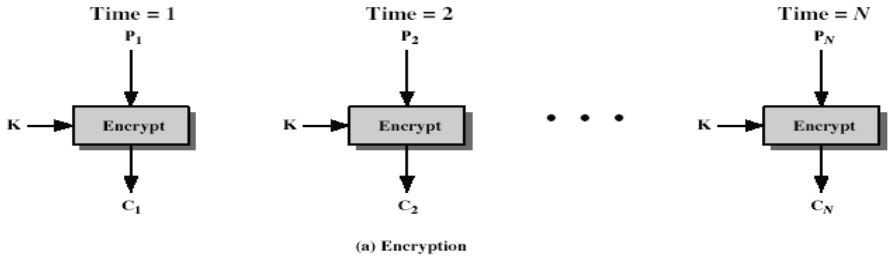
- block ciphers encrypt fixed size blocks
  - eg. DES encrypts 64-bit blocks with 56-bit key
    AES encrypts 128-bit blocks with 128 / 192 / 256 bit keys
- need some way to en/decrypt arbitrary amounts of data in practice
- **ANSI X3.106-1983 Modes of Use** (now FIPS 81) defines 4 possible modes
- subsequently 5 defined for AES & DES
  - ECB, CBC, CFB, OFB, CTR
- Many others have been proposed

MODI di utilizzare i cifrari a BLOCCHI/a flusso:

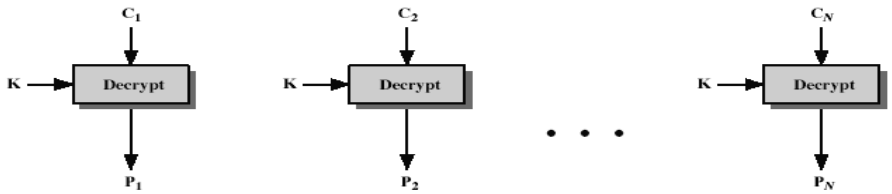
Electronic CodeBook (ECB)

→ non va bene perché se ho due plaintext uguali (blocchi) ottengo gli stessi chypertext e quindi posso capire se ci sono delle ripetizioni

- message is broken into independent blocks
- each block is a value which is substituted, encoded independently of the other blocks : $C_i = DES_{K1}(P_i)$



(a) Encryption



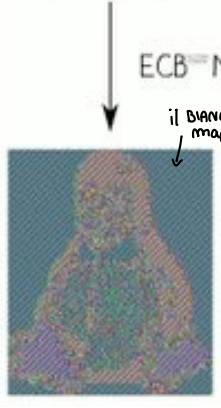
(b) Decryption

Advantages and Limitations of ECB

- message repetitions may show in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which code-book analysis problem
- weakness is due to the encrypted message blocks b independent
- main use is sending a few blocks of data (usually 1

PROBLEMA: a blocco uguale corrisponde chyper uguale
la struttura del file viene preservata nella cifratura

es. cifratura con ECB a 16 bit



Cipher Block Chaining (CBC)

- message is broken into blocks
- linked together in encryption operation
- each previous cipher blocks is chained with current plaintext block, hence name
- use Initial Vector (IV) to start process

$$C_i = E_K(P_i \text{ XOR } C_{i-1})$$

ENCRYPT

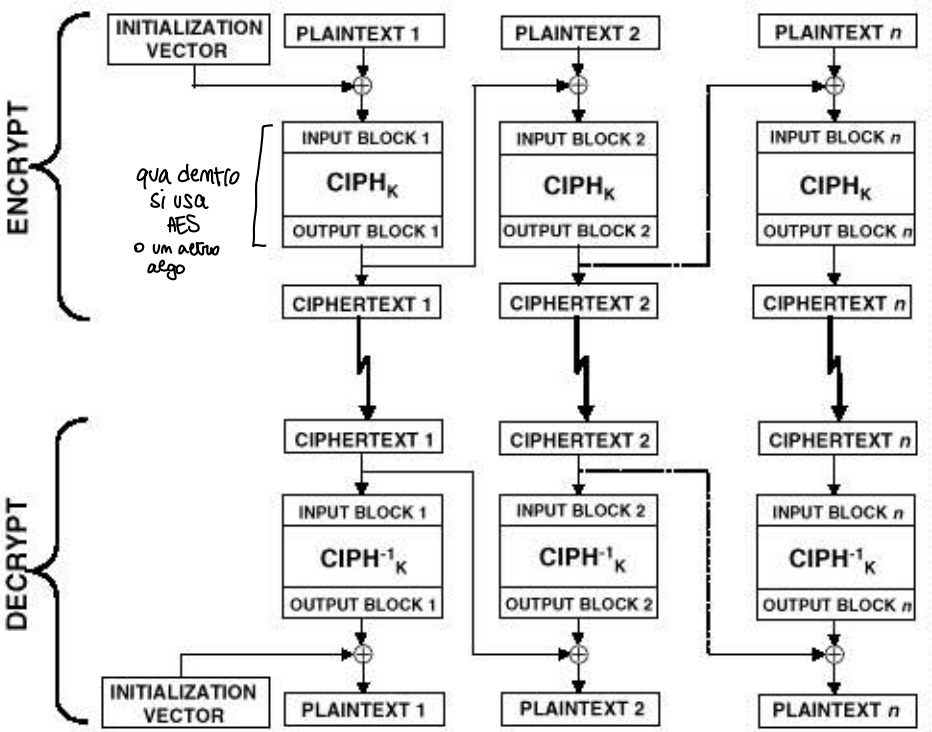
P_i: blocco in chiaro corrente

C_{i-1}: blocco precedente cifrato

$$C_0 = IV$$

- uses: bulk data encryption (e.g., email, files, web), authentication

Cipher Block Chaining (CBC)

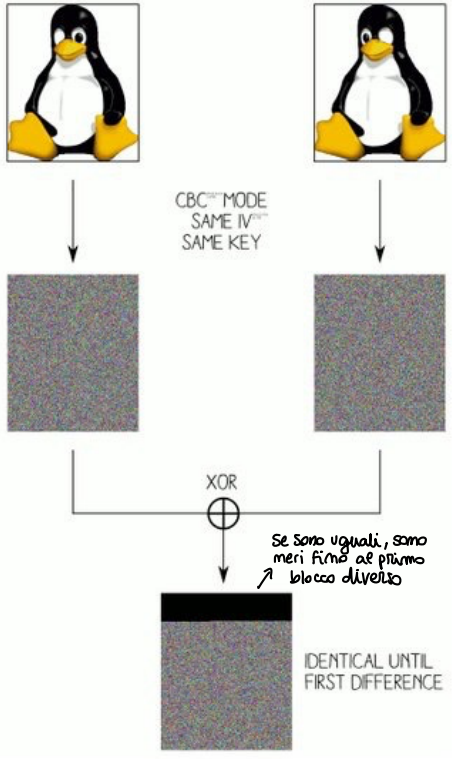


Advantages and Limitations of CBC

↳ tutti i BLOCCHI sono DIPENDENTI: ogni modifica in un blocco influisce su tutti i blocchi successivi

- a ciphertext block depends on **all** blocks before it
  - any change to a plaintext block affects all following ciphertext blocks
- need **Initialization Vector** (IV)
  - which must be known to sender & receiver
  - if sent in clear, attacker can change bits of first block, and change IV to compensate
  - hence IV must either be a fixed value (as in EFTPOS)
  - or must be sent encrypted in ECB mode before rest of message

- si tiene fisso
- viene inviato in ECB



Output FeedBack (OFB)

trasformo un cifrario a BLOCCHI in un cifrario a FLUSSO

- message is treated as a stream of bits
- output of cipher is added to message → la chiave qui è il flusso (IV (inizio) + chiave per crittografare con AES)
- output is then feed back (hence name)
- feedback is independent of message
- can be computed in advance

$$C_i = P_i \text{ XOR } O_i$$

(CYPHERTEXT = PLAINTEXT XOR OUTPUT (flusso))

$$O_i = E_K(O_{i-1})$$

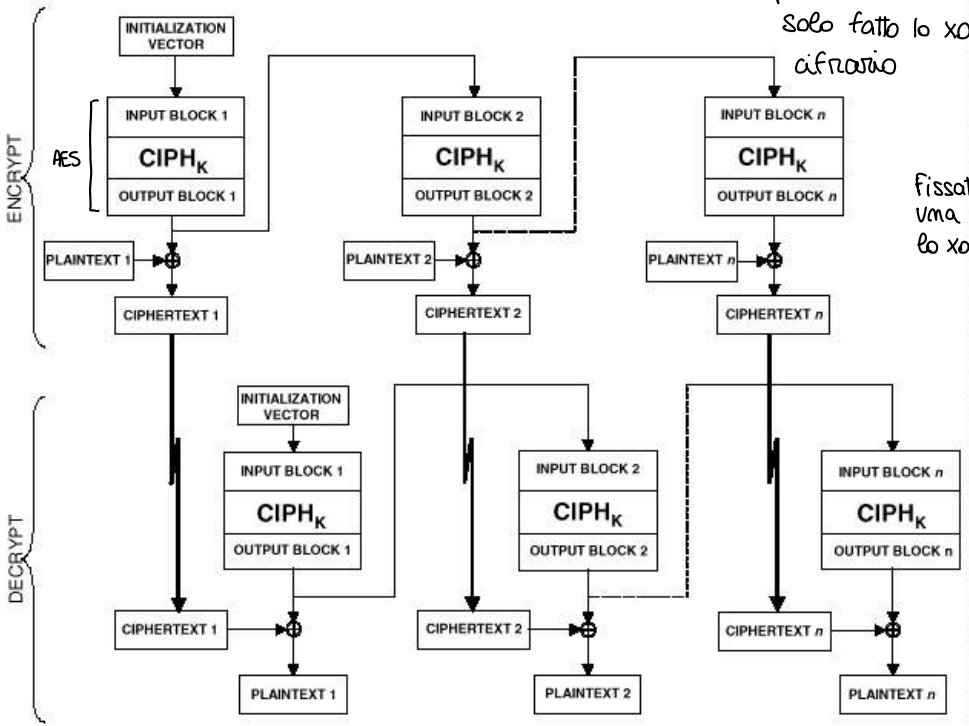
(OUTPUT)

$$O_0 = IV$$

- uses: stream encryption on noisy channels

Output FeedBack (OFB)

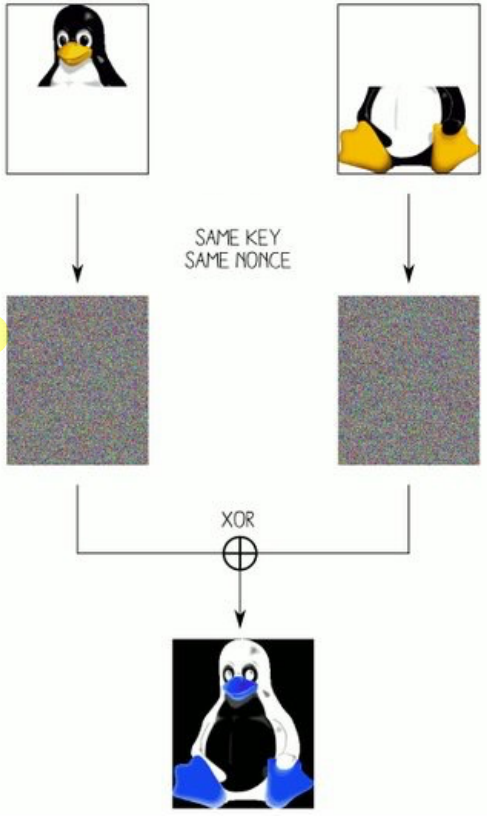
uso i dati del cifrario per fare la XOR con il plaintext. il plaintext non è dato in INPUT, viene solo fatto lo XOR con l'OUTPUT del cifrario



Fissato K e IV allora ritorna una sequenza con cui poi fare lo XOR

Advantages and Limitations of OFB

- bit errors do not propagate
- more vulnerable to message stream modifica
- a variation of a Vernam cipher
  - hence must **never** reuse the same "pseud sequence (key+IV)
- sender & receiver must remain in sync
- originally specified with m-bit feedback
- subsequent research has shown that only **ful** (ie CFB-64 or CFB-128) should ever be used



Counter (CTR)

- a "new" mode, though proposed early on
- similar to OFB but encrypts counter value rather than any feedback value → cifro un contatore e ottengo 16 byte pseudocasuali
- must have a different key & counter value for every plaintext block (never reused)

$$C_i = P_i \text{ XOR } O_i$$

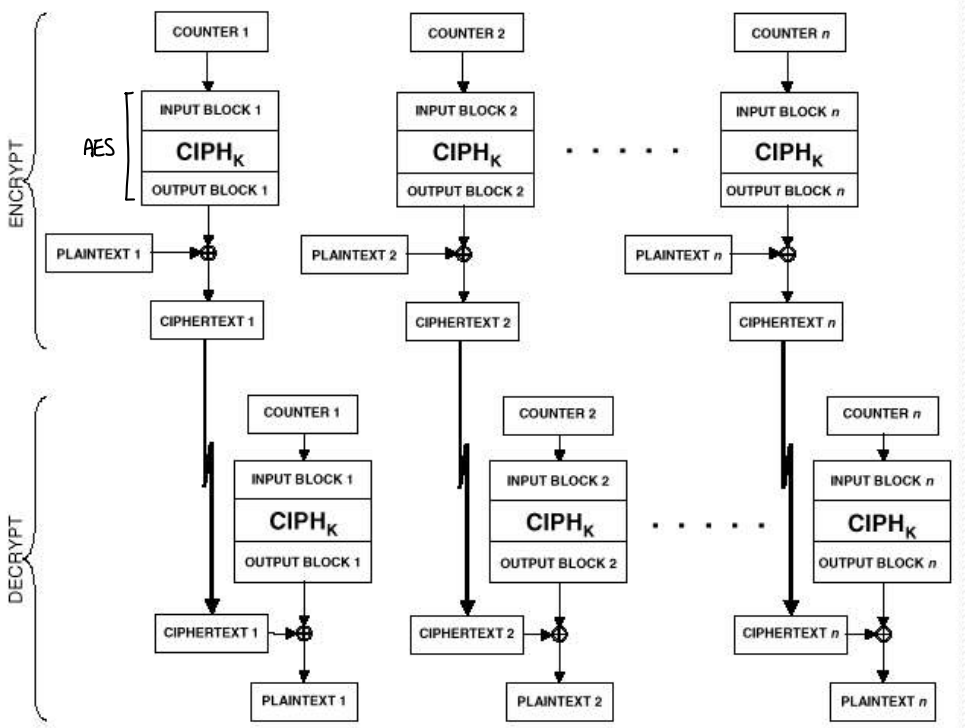
$$O_i = E_K(i)$$

↳ codifico il counter e poi faccio lo XOR blocco a blocco

- uses: high-speed network encryptions (ATM, IPsec)

Counter (CTR)

Ogni blocco è CIFRATO e DECIFRATO separatamente dagli altri



Advantages and Limitations of CTR

- efficiency
  - can do parallel encryptions in h/w or s/w
  - can preprocess in advance of need
  - good for bursty high speed links
- random access to encrypted data blocks (e.g., encrypted file systems)
- provable security (good as other modes)
- but must ensure never reuse key/counter values, otherwise could break (cf OFB)

a differenza di OFB non serve tenere in memoria il blocco precedente

ora ci concentriamo come garantire altri goal

Message Integrity and Authentication

↳ proteggere i messaggi da modifiche non autorizzate (o se qualcuno le modifica bisogna accorgersene)

- message integrity and authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
- will consider the security requirements
- then three alternative functions used:
  - message encryption
  - message authentication code (MAC)
  - hash functions

Security Requirements

Brief recall of security attacks:

- disclosure
- traffic analysis
- masquerade
- content modification
- sequence modification
- timing modification
- source repudiation
- destination repudiation

(passive) attacks to confidentiality

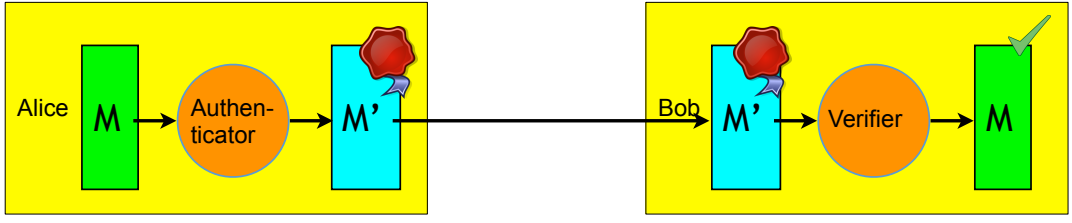
(active) attacks to authenticity of data, sender (eg., digital signatures), receiver

Authentication mechanisms

aggiunge informazione di autenticità

verifica autenticità e integrità del messaggio

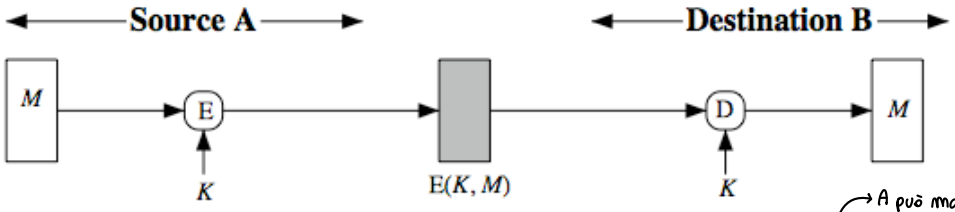
- Each mechanism is composed by an **authenticator** and a **verifier**, run separately



- Data produced by authenticators can be verified independently (in protocols)
- Kind of authenticators:
  - Message Encryption
  - Message Authentication Codes → vedo se chi me e'ha mandato è lui
  - Hash functions → vedo se il file è integro

Message Authentication by Symmetric Encryption

CIFRATURA



(a) Symmetric encryption: confidentiality and authentication

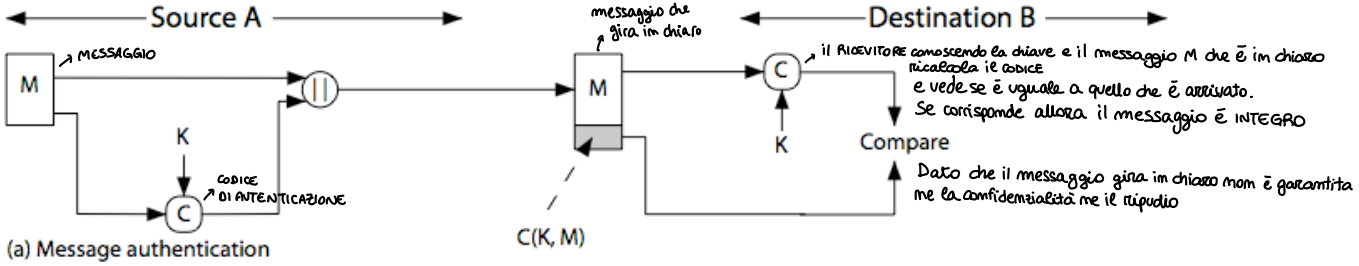
la cifratura è un meccanismo di autenticità ma non di firma (non impedisce il ripudio perché essendo simmetrica non c'è un terzo?)

→ A può mandare il messaggio e può dire poi che non ha mandato niente

- if symmetric encryption is used then:
  - receiver knows sender must have created it
    - since only sender and receiver know key used
  - receiver knows content cannot have been altered
    - if message has suitable structure, redundancy or a checksum to detect any changes
  - Does not avoid **repudiation** by A or **forgery** by B

Message Authentication Code (MAC)

→ creare un codice di AUTENTICAZIONE del MESSAGGIO



- generated by an algorithm that creates a small fixed-sized block
  - depending on both message and some key → l'algoritmo genera il CODICE in base alla KEY e al MESSAGGIO
  - like encryption though need not be reversible
- appended to message as a signature (↑ il CODICE)
- receiver performs same computation on message and checks it matches the MAC
- provides receiver the assurance that message is unaltered and comes from sender
- note that a MAC is **not** a digital signature
  - Both sender & receiver share key and could create it

MAC Properties

- a MAC is a cryptographic checksum

  $MAC = C_K(M)$ ↗ da un messaggio di lunghezza variabile, in base alla chiave, restituisce un messaggio di lunghezza fissa

  messaggio ↑ $f_{HASH}(M+K) = MAC$ ↓ chiave → e invio M+MAC

  - condenses a variable-length message M
  - using a secret key K
  - to a fixed-sized authenticator
- is a many-to-one function — ho un num FINITO di codici che posso assegnare con la HASH e ho ∞ messaggi possibili → collisioni (mappare ∞ messaggi su FINITI codici)
  - potentially many messages have same MAC
  - but finding these needs to be very difficult

Hash Functions

→ non miriamo alla segretezza (perché non sono invertibili)
dimostriamo che il pacchetto arriva integro

- condenses arbitrary message to fixed size

`h = H(M)`

- usually assume that the hash function is public and **not keyed**
  - cf. MAC which is keyed
- hash used to detect changes to message
- can use in various ways with message
- most often to create MAC and digital signature

Requirements for Hash Functions

1. can be applied to any sized message `M`
2. produces fixed-length output `h`
3. is easy to compute `h=H(M)` for any message `M`
4. and must be resistent in three ways:
   1. Preimage resistance: given h is infeasible to find x s.t. H(x)=h
   2. Second preimage resistance: given x is infeasible to find y s.t. H(y)=H(x).
   3. Collision resistance: It is infeasible to find any x,y s.t. H(y)=H(x).

invertirla deve essere difficile

→ dato l'output (fisso) di m bit quanti tentativi devo fare per trovare due valori t.c. $h(x)=h(y)$? 2^m

→ PARADOSSO del COMPLEANNO: se ho m bit di HASH per avere una $p(0,5) = 1,18 \cdot \sqrt{2^n} \cong 2^{n/2}$

(per i compleanni ho m = 365 giorni e ho un clash di compleanni già su circa $1,18 \cdot \sqrt{365} \simeq 23$ persone)

Secure Hash Algorithm

- SHA originally designed by NIST & NSA in 1993
- was revised in 1995 as SHA-1
- US standard for use with DSA signature scheme
  - standard is FIPS 180-1 1995, also Internet RFC3174
  - nb. the algorithm is SHA, the standard is SHS
- based on design of MD4 with key differences
- produces 160-bit hash values
- In 2005, some results on security of SHA-1 have raised concerns on its use in future applications
  - Clash can be found in 2^{69} operations instead of 2^{80}.

Revised Secure Hash Standard

- NIST issued revision FIPS 180-2 in 2002
- adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
- designed for compatibility with increased security provided by the AES cipher
- structure & detail is similar to SHA-1
- hence analysis should be similar
- but security levels are rather higher

BIT
↳ usato per la firma digitale
hash che produce blocchi di 512 bit indipendentemente dalla lunghezza delle stringhe in input

Use of Hash digests in Message Authentication

HASH per costruire il MAC $h(K, M) = MAC$

- A message digest does not authenticate the sender of the message.
- To provide message authentication, Alice needs to provide proof that it is Alice sending the message and not an impostor.
- The digest created by hash function is called a modification detection code (MDC). → verifica l'integrità del pacchetto
- What we need for message authentication is a message authentication code (MAC). → con la chiave verifica che il mittente sia quello

ho il messaggio "pippo" e la chiave "ciao" e per verificare l'integrità calcolo $h(ciaopippo) = x$ (append; x = MAC)
e invio (pippo, x). Il ricevente verifica calcolando $h(ciaopippo)$ e verifica se = x.
(anche il ricevente ha la chiave)

MDC lo ottengo se faccio h(M) = MDC ma rileva solo se ci sono modifiche al messaggio
MAC lo ottengo se faccio h(M,K) = MAC e mi garantisce l'integrita'

Hash Functions & MAC Security

- **brute-force attacks** exploiting
  - strong collision resistance hash have cost $2^{m/2}$
    - have proposal for h/w MD5 cracker (64 bit)
    - 128-bit hash looks vulnerable, 160-bits better
  - MACs with known message-MAC pairs
    - can either attack keyspace (cf key search) or MAC
    - at least 128-bit MAC is needed for security
- **cryptanalytic attacks** exploit structure
  - like block ciphers want brute-force attacks to be the best alternative

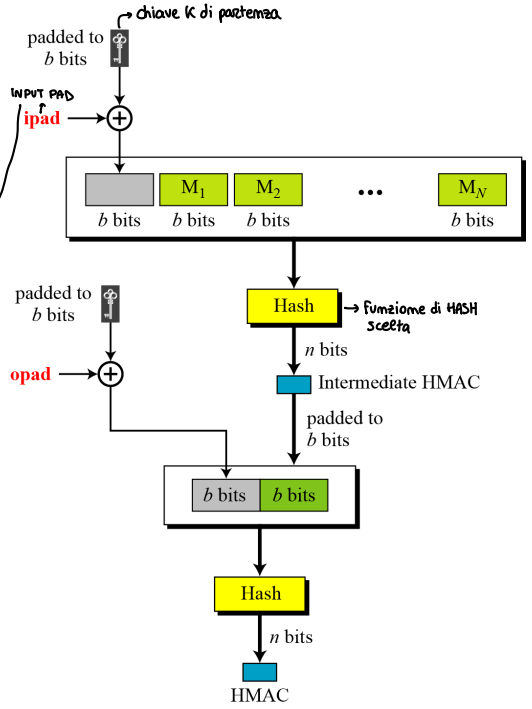
HMAC

Standard con cui si usano le funzioni di HASH per generare i MAC

- MAC based on hash functions - kind of keyed hash
- specified as Internet standard RFC2104
- uses hash function on the message:

$$HMAC_K = Hash[(K^+ \text{ XOR } opad) \,||\, Hash[(K^+ \text{ XOR } ipad)||M)]]$$

- where K^+ is the key padded out to size and opad=0x5C='\', ipad=0x36='6' are specified padding constants
- overhead is just 3 more hash calculations than the message needs alone
- any hash function can be used
  - eg. MD5, SHA-1, RIPEMD-160, Whirlpool



Una volta che garantisco l'INTEGRITÀ, se uno vuole avere anche l'AUTENTICITÀ cifra tutto quanto con un'altra chiave (es. cifro con AES)

→ inviare il pacchetto in CHIARO, ma non si può modificare (se lo modifichi il RICEVENTE se ne accorge)

→ ovvero inviare il pacchetto CIFRATO, non si può leggere

CHIAVE DI MODIFICA ≠ CHIAVE DI CIFRATURA

Se ci affido e basta. Non garantisco l'INTEGRITÀ, il RICEVENTE non si accorge di modifiche.

⇒ PER CONFIDENZIALITÀ → AES (cifratura)
PER INTEGRITÀ → HMAC (hash)

HMAC Security

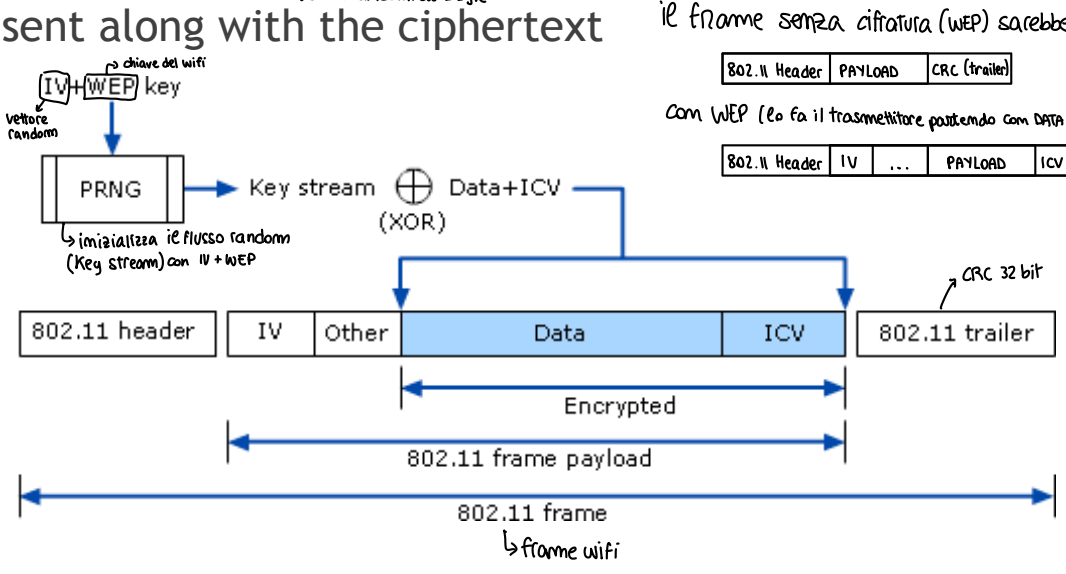
- proved security of HMAC relates to that of the underlying hash algorithm
- attacking HMAC requires either:
  - brute force attack on key used
  - birthday attack (but since keyed would need to observe $2^{n/2}$ messages generated with the same key)
- choose hash function used based on speed versus security constraints
  - MD5 is good enough, and faster than SHA-1

Example application of Symmetric key message authentication: Wired Equivalent Privacy (1999)

- 802.11 defines WEP, a symmetric key encryption schema between host and access point
  - se non so la psw non vedo il contenuto dei pacchetti, nella rete cablata invece i pacchetti viaggiano in chiaro ma per vederli devo collegarmi ad uno switch e smiffare
- Intended to offer same **confidentiality** of wired connections, and **restrict access** to infrastructure to authorized hosts only
  - se io attaccante provo ad inviare pacchetti ad una rete di cui non so la passw, l'AP lo scarta
- Only **authenticated** packets can access to the network
- Key (password wifi) is shared between host and AP
  - Can be (and usually is) the same for all hosts
  - Not specified how this key is exchanged between hosts and AP (usually, by external channels, e.g. orally/written)
  - Nor how long it is valid (often is unchanged for years)
    - la password del wifi raramente viene cambiata perché l'utente non rispetta le regole

WEP: encryption

- 802.11 payload+checksum (ICV) is encrypted using (a variant of) RC4, with a 24-bit IV (3 byte)
- IV is sent along with the ciphertext



vettore random di 3 byte

il frame senza cifratura (WEP) sarebbe

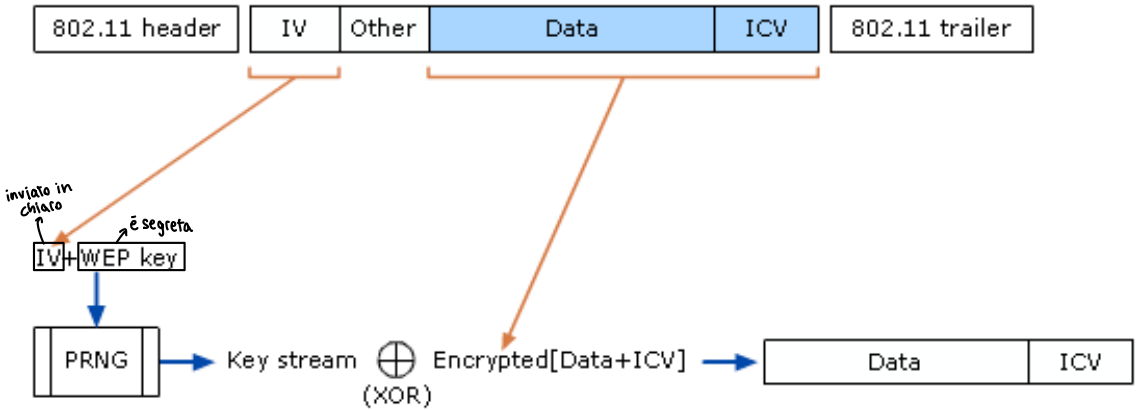
802.11 Header	PAYLOAD	CRC (trailer)

Con WEP (lo fa il trasmettitore partendo con DATA e CHIAVE WEP)

802.11 Header	IV	...	PAYLOAD	ICV (hash)	CRC (trailer)

WEP: decryption

- The receiving station initializes the PRNG with the IV just received, to obtain the keystream



fa per ogni singolo frame wifi: garantisce segretezza (perché c'è lo stream e i dati sono CIFRATI)
controllo di accesso (AP scarta i frame che non passano i test)

WEP: encryption/decryption

il problema è che IV sono 3 byte

- Encryption/decryption can be done by the WiFi hardware (hence IV is sent before the ciphertext)
- 802.11 does not specify how the IV must be generated and changed (but it is recommended to change it often, in order to avoid reuse attacks)
- In fact, many WEP cards, at each packet:
  - either auto-increment IV, starting from 0
  - or generate a pseudo-random IV, starting from some fixed or random seed

Quello che cifra i dati è il case stream generato da IV e chiave → PROBLEMI: la chiave è sempre la stessa per anni
l'IV lo leggiamo perché è in chiaro → posso ricostruire i pacchetti
se uno invece non conosce la chiave può fare un ATTACCO COMPLEANNO

Birthday attack to WEP (Borisov et al, 2001)

- Keystream is identified by the pair (key,IV)
- We have to find at least two packets encrypted with the same pair → per trovare due IV uguali basta che aspetti (dato che lo stream ha sequenze finite), anche se ho due cyphertext diversi, ma con stesso stream, se faccio lo XOR trovo le PLAINTEXT (in XOR) — PAYLOAD cifrati
- Usually key is unchanged for long time (months, years)
- If IV is auto-incremented, it repeats at most after $2^{24} \simeq 16 \times 10^6$ packets, but lower IV are more common (due to wireless card restarts)
- If IV is pseudorandom, by "birthday paradox":
  - $n(0,5) \simeq 1,18*(2^{24/2}) = 1,18*2^{12} = 4833$
  - $p(10000) \simeq 0,95$

↳ 10 000 pacchetti li ottengo in circa 30 secondi

Birthday attack to WEP (Borisov et al, 2001)

- 1 Ethernet frame carries up-to 1500 bytes
  - 1 MB data = at least 700 frames
  - 10000 frame = up-to 15MB
- 802.11g can transmit up-to 14000 packets/s
- After enough sniffing, we observe two (or more) packets whose payloads C1, C2 are keyed with the same keystream KS.
- Then: $C1 \oplus C2 = (P1 \oplus KS) \oplus (P2 \oplus KS) = P1 \oplus P2$
- With some analysis (e.g. statistic, known text, frequency) we can recover the plain texts P1 and P2
  - We got a passive attack to data confidentiality!

Birthday attack to WEP (Borisov et al, 2001)

- Also, $C1 \oplus P1 = P1 \oplus KS \oplus P1 = KS$
- recover a piece of keystream which can be used for encrypting & transmitting packets no longer than P1
- With enough packets, we can recover enough keystream chunks to transmit whatever we want → riusciamo a far credere agli AP che i nostri pacchetti appartengono alla rete
  - Active attack to the access of the resource (the AP)
  - Since livetime of IV is not specified, access point must accept any IV, even always the same

Cryptographic attack to WEP (Fluhrer et al, 2001)

- For some IVs, RC4 is weak
  - in WEP, weak IV are of the form (a+3,255,x)
- an attacker knowing the m-th byte of the keystream can derive the m+1-th byte
  - In WEP, the first data byte often comes from the Subnetwork Access Protocol
- it is possible to (statistically) derive in sequence each byte of the key from the keystream chunks and IV
- By repeating enough the birthday attack we can recover enough chunks and IV to recover the whole key

Key Reinstallation Attack (KRACK, 2017)

- Recently, Vanhoef and Piessens found an attack to the 4-way handshake (the bug is in the protocol!)
  - Takes advantage of a retransmission of messages, which is allowed for dealing with messages loss
  - allows an attacker to reinstall a new session key between a client and the access point
    - In the case of Android 6.0, the key can be forced to be 0
  - In CCMP, an adversary can replay and decrypt (but not forge) encrypted packets
  - In TKIP (and GCMP), an adversary can replay, decrypt and forge encrypted packets!
- Patches and workarounds are under way (but many systems are still vulnerable)

Public-Key Cryptography

Finora abbiamo fatto la crittografia SIMMETRICA per CONFIDENZIALITA' e AUTENTICAZIONE (integrità)
ma non riusciamo a garantire il NON RIPUDIO perché è SIMMETRICA (non sappiamo distinguere un utente dall'altro)

- traditional private/single key cryptography uses one key, shared by both sender and receiver
- if this key is disclosed communications are compromised
- Good for ensuring confidentiality (passive attacks)
- also is **symmetric**, parties are equal
- hence does not protect sender from receiver forging a message & claiming is sent by sender (active attack)

Public-Key Cryptography

- probably most significant advance in the 3000 year history of cryptography
- uses **two** keys – a public & a private key
- **asymmetric** since parties are **not** equal
- Actual implementations use clever application of number theoretic concepts to function (easy on one way, hard in the other)
- complements **rather than** replaces private key crypto

Compleementa la simmetrica
↳ fa le lavoro sporco

Why Public-Key Cryptography?

- developed to address two key issues:
  - **key distribution** – how to have secure communications in general without having to trust a KDC with your key
    ↳ KEY DISTRIBUTION CHANNEL
  - **digital signatures** – how to verify a message comes intact from the claimed sender → il NON RIPUDIO non lo posso verificare con la simmetrica
- public invention due to Whitfield Diffie & Martin Hellman at Stanford Uni in 1976
  - known earlier in classified community, e.g. UK CESG (James Ellis, 1970) and NSA (mid-60's)

Public-Key Cryptography

avere una chiave con PARTE PUBBLICA | PARTE PRIVATA insolubili
↓ mota a tutti
↓ mota a chi crea la chiave
ASSIMMETRIA (lui sa il segreto e gli altri no)

- **public-key/two-key/asymmetric** cryptography involves the use of **two** keys:
  - a **public-key** PU_A, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key** PR_A, known only to the creator of the key, used to **decrypt messages**, and **sign** (create) **signatures**
- is **asymmetric** because those who encrypt messages or verify signatures **cannot** decrypt messages or create signatures

Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide **secrecy**)
    - The sender encrypts a message with the recipient's public key
  - **digital signatures** (provide **authentication** and **non-repudiation**)
    - The sender "signs"a message with its private key
  - **key exchange** (of session keys)
    - Use PKC for implementing secure channel for exchanging session keys
- some algorithms are suitable for all uses, others are specific to one

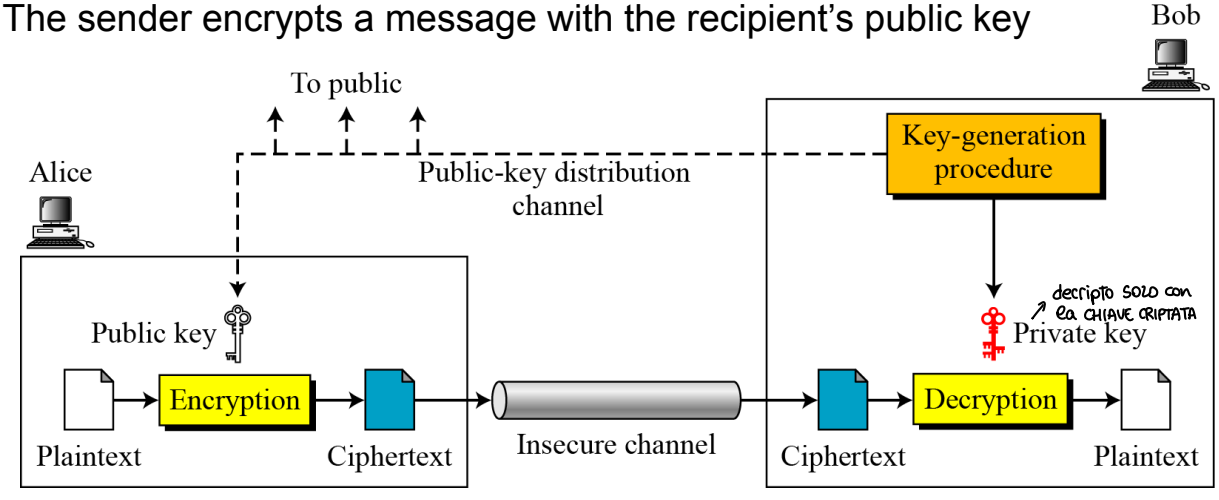
cifra con la chiave pubblica del destinatario

↳ il NON RIPUDIO lo ottengo FIRMANDO con la mia chiave privata

Public-Key Cryptography

ovviamente dalla chiave pubblica non si può risalire alla chiave privata

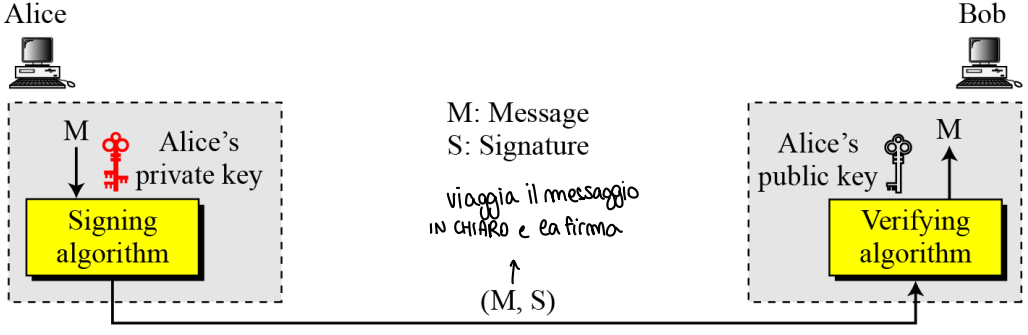
- **encryption/decryption** (provide secrecy)
  - The sender encrypts a message with the recipient's public key



Public-Key Digital Signature

garantisce il ripudio (non conta qui la segretezza)

- **digital signatures** (provide **authentication** and **non-repudiation**)
  - The sender "signs"a message with its private key



ora ALICE non può ripudiare il messaggio

solo ALICE può generare la sua firma, perché solo lei ha la sua chiave privata per generarla

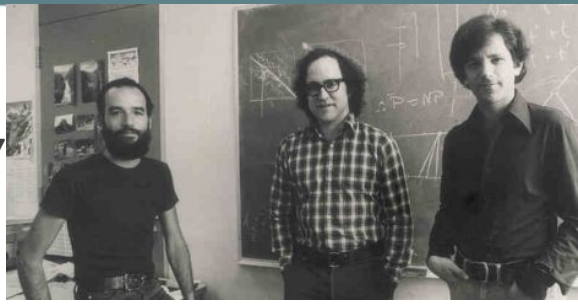
Public-Key Characteristics

gli schemi a chiave pubblica usano funzioni ONE WAY: semplici da calcolare ma per cui è molto difficile calcolarne l'inversa

- Public-Key algorithms rely on two keys (or better, two parts of the same key) where:
  - it is computationally infeasible to find private key knowing only algorithm & public key
  - it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
    - public key schemes utilise one-way functions: problems that are easy (P type) one way but hard (NP type) the other way
      ↳ per calcolare f
      ↳ per calcolare f^{-1}
  - either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms)

le f vengono chiamate TRAP DOOR se dato un parametro EXTRA, calcolare f^{-1} diventa facile

RSA



- by Rivest, Shamir & Adleman of MIT in 1977
- best known & widely used public-key scheme
- based on exponentiation in a finite (Galois) field over integers modulo a prime
  - exponentiation takes $O((\log n)^3)$ operations (easy)
- uses large integers (eg. 1024-2048 bits)
- security due to cost of factoring large numbers
  - factorization takes $O(e^{\log n \log \log n})$ operations (hard)

Euler Totient Function ø(n)

funzione di Eulero, TOZIENTE

- when doing arithmetic modulo n
- **complete set of residues** is: `0..n-1`
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
  - reduced residues have multiplicative inverses
- number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

definita per ogni intero positivo n, è il NUMERO di interi tra 1 e n che sono COPRIMI con n. es. $\phi(8) = 4$, ovvero $\{1,3,5,7\}$

↳ non hanno fattori in comune con n

mom hanno fattori in comune con n

PRIMO
$p \rightarrow \phi(p) = p-1$

$p \cdot q \rightarrow \phi(p \cdot q) = (p-1)(q-1)$
PRIMI

$7 \cdot 3 = 1$ perché devo considerare mod 10

→ per ciascun numero dentro al set esiste un solo numero che se moltiplicato per esso fa 1. es. $3^{-1} = 7$

RSA Key Setup

- each user generates a public/private key pair by:
- selecting two large primes at random - `p, q`
- computing their system modulus `n=p*q`
  - note `ø(n)=(p-1)(q-1)` (↳ TOZIENTE)
- selecting at random the encryption key `e` → scelgo un numero "e" che deve essere primo rispetto a $\phi(n)$
  - where `1<e<ø(n), gcd(e,ø(n))=1`
- solve following equation to find decryption key `d`
  - `e*d=1 mod ø(n) and 0≤d≤n`
- publish their public encryption key: `PU={e,n}`
- keep secret private decryption key: `PR={d,n}`

RSA Use

- to encrypt a message M the sender:
  - obtains **public key** of recipient PU={e,n}
  - computes: $C = M^e \bmod n$, where $0 \leq M < n$
- to decrypt the ciphertext C the owner:
  - uses their private key PR={d,n}
  - computes: $M = C^d \bmod n$
- note that the message M must be smaller than the modulus n (block if needed)
- large exponents can be computed efficiently with square-and-multiply algorithm

Why RSA Works

- because of Euler's Theorem:
  - $a^{\phi(n)} \bmod n = 1$ where $\gcd(a,n)=1$
- in RSA we have:
  - $n=p*q$
  - $\phi(n)=(p-1)(q-1)$
  - carefully chose e & d to be inverses $\bmod \phi(n)$
  - hence $ed=1+k*\phi(n)$ for some k
- hence:

$$C^d = M^{e*d} = M^{1+k*\phi(n)} = M^1 (M^{\phi(n)})^k = M^1 (1)^k = M^1 = M \bmod n$$

- (this argument is incorrect if M is not coprime with n, but even in this case the result holds for Fermat's little theorem - for a more complete proof, see https://www.di-mgt.com.au/rsa_theory.pdf)

RSA Key Generation

- users of RSA must:
  - determine two primes at random - p, q
  - select either e or d and compute the other
    - usually e=65537
- primes p,q must not be easily derived from modulus n=p*q
  - means n must be sufficiently large
  - typically try a Mersenne (2^p-1) or Fermat ($2^{2^n}+1$) number and use AKS or Miller-Rabin primality test
- exponents e, d are inverses, so use Inverse algorithm to compute the other

1. Select primes: $p=17$ & $q=11$
2. Compute $n = p*q = 17*11=187$
3. Compute $\phi(n)=(p-1)(q-1)=16*10=160$
4. Select e: $\gcd(e,160)=1$; choose $e=7$
5. Determine d: $d*e=1 \bmod 160$ and $d < 160$ Value is d=23 since $23*7=161= 1*160+1$
6. Publish public key PU={7,187}
7. Keep secret private key PR={23,187}

RSA Example - En/Decryption

- sample RSA encryption/decryption is:
- given message $M = 88$ (nb. $88<187$)
- encryption:
  $C = 88^7 \bmod 187 = 11$
- decryption:
  $M = 11^{23} \bmod 187 = 88$

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>1024 bits)
  - A 64-bit private key scheme has roughly similar security to a 512-bit RSA
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, but is made hard enough to be impractical to break
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

chiave privata SIMMETRICA

RSA è lento a CIFRARE, lo si usa per cifrare la chiave da mandare in GIRO e poi usiamo es. AES per cifrare

RSA Security

- possible approaches to attacking RSA are:
  - brute force key search
    - infeasible given size of numbers
  - mathematical attacks
    - based on difficulty of computing ø(n), by factoring modulus n
  - timing attacks (on execution of decryption)
  - chosen ciphertext attacks (given properties of RSA)

Other public-key cryptosystems

- ElGamal: based on *discrete logarithm*
  - For p large prime number, e_1 a primitive root (i.e., a generator of the reduced residues):
    - given d, to compute $e_2 = e_1^d \bmod p$ is easy
    - given e_2, to compute $d = \log_{e1} e_2 \bmod p$ is hard
- *Elliptic curves*: based on the solutions in a finite field of an equation of the form

$$y^2 = x^3 + ax^2 + b$$

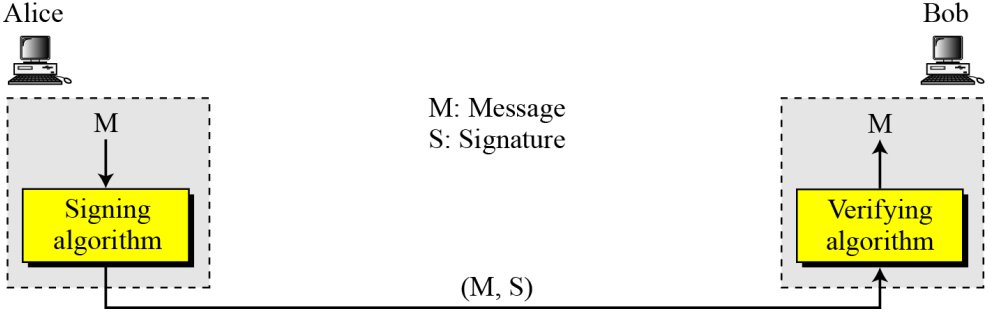
  - This set forms a group, where multiplication is easy but discrete logarithm is hard
  - Discrete logarithm is much more difficult than factorisation, hence shorter keys are sufficient (a 256-bit elliptic curve public key is equivalent to a 3072-bit RSA public key)
  - Implemented in the Digital Signature Algorithm, Elliptic Curve Digital Signature Algorithm (ECDSA), etc.

FINO a QUI

Digital Signatures

posso avere:
- ascolta audio un po' indietro

- digital signatures provide the ability to:
  - **verify author, date & time of signature**
  - **authenticate** message contents
  - be verifiable by **third parties** to resolve disputes
- include authentication function with additional capabilities



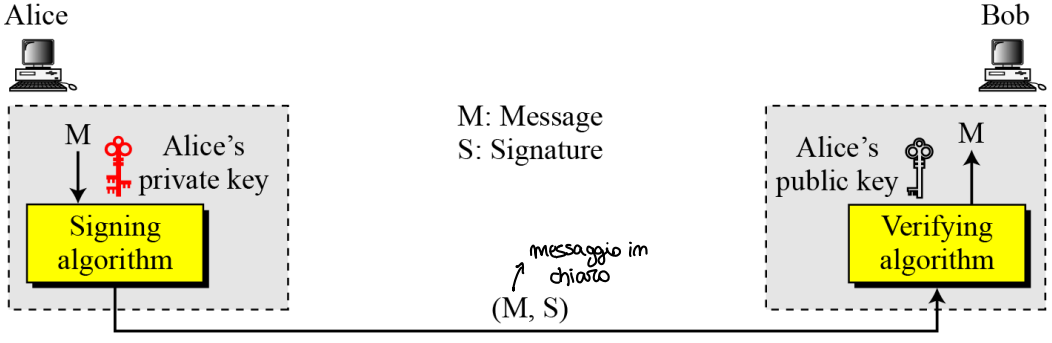
Digital Signature Properties

- must depend on the message signed
- must use information **unique to sender**
  - to prevent both forgery and denial
- must be relatively **easy to produce**
- must be relatively **easy to recognize & verify**
- be **computationally infeasible to forge** FALSIFICARE
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be **practical to save digital signature** in storage

Signature by asymmetric key

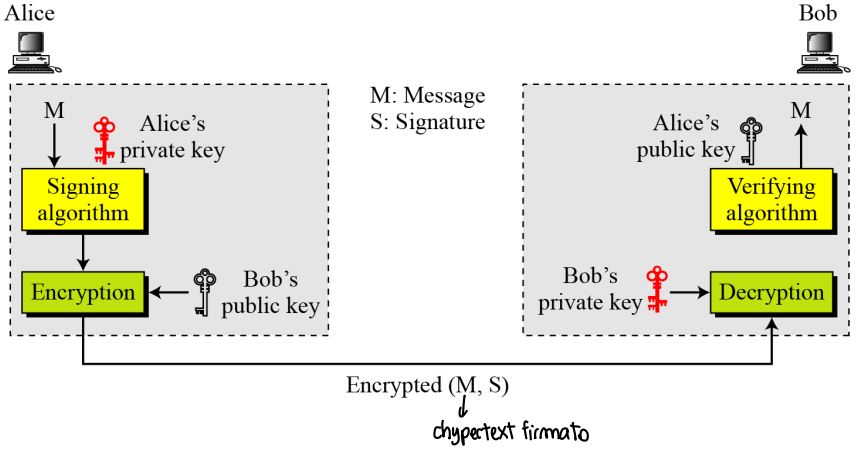
- A digital signature needs a public-key system.
- The signer signs with her private key; the verifier verifies with the signer's public key.

la FIRMA garantisce l'INTEGRITÀ (non serve aggiungerci un hashing)



Digital signatures do not provide confidentiality

- If needed, use a second layer (using e.g. public keys)

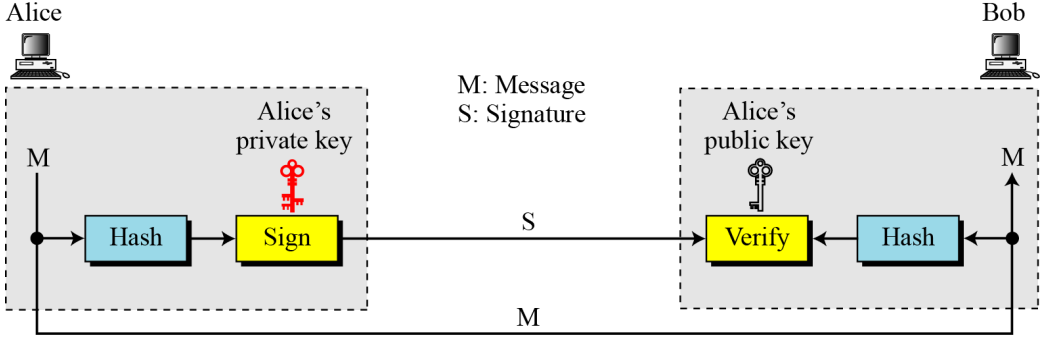


Signing the digest

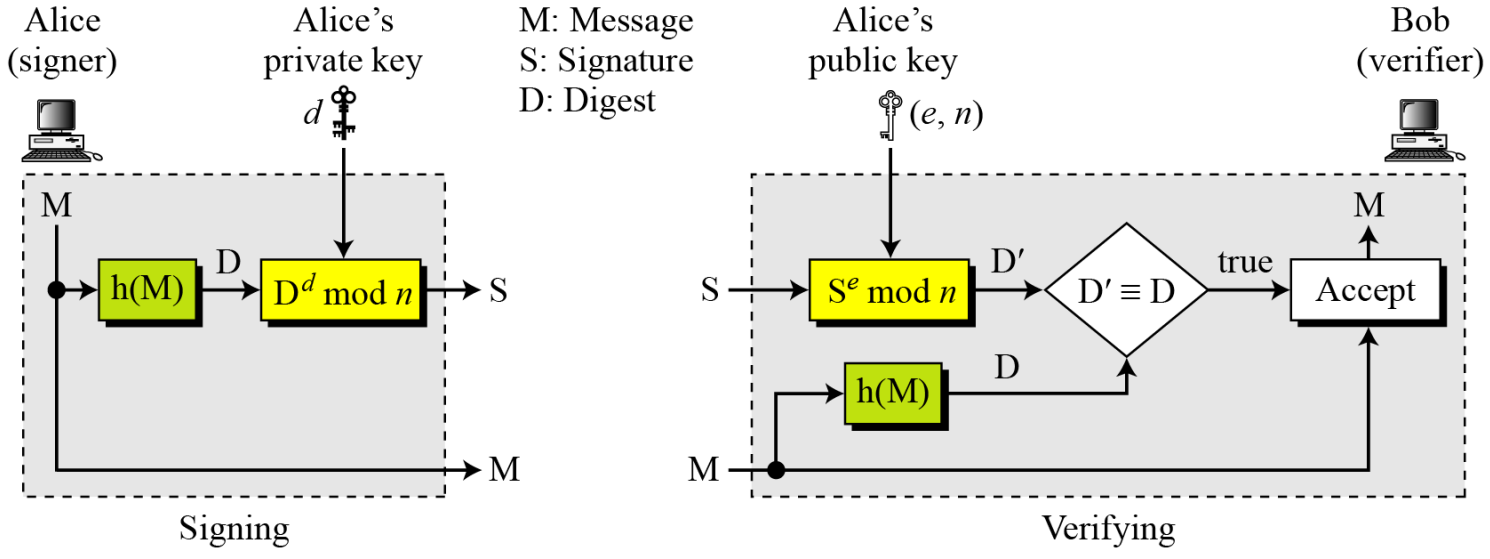
Schema attuale, la FIRMA DIGITALE non la faccio su M, ma sull'hash di M

- Needed because asymmetric cryptosystems are slow
- Secure if hash functions are secure enough

non può firmare anche il TIMESTAMP perché ALICE potrebbe falsificarlo. Ci vuole un terzo



Example: RSA digital signature scheme on message digest



Distribution of Public Keys: Public-Key Certificates

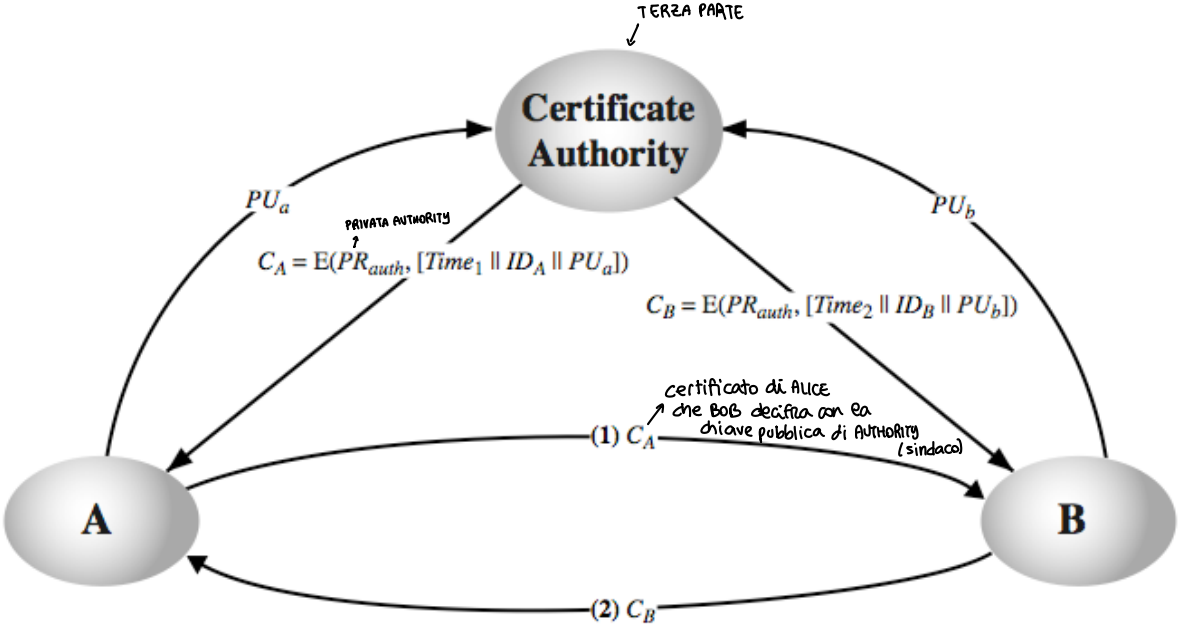
↳ come distribuisco le chiavi pubbliche? come fanno quelli che cifrano con la chiave pubblica a sapere che quella sia effettivamente del destinatario?

- a **certificate** binds **identity** to **public key**
  - usually with other info such as period of validity, rights of use etc

↳ FILE che collega identità a chiave pubblica, es. ALICE Pu_{ALICE} → una terza parte deve firmare questo file (in modo che non venga modificato, integrità)

- with all contents signed by a trusted Public-Key or **Certificate Authority** (CA)
- can be verified by anyone who knows the public-key authorities public-key
- Certificates are only generated by the CA, but not kept in the CA (to avoid tampering and bottlenecks)

Public-Key Certificates



Public-Key Certificates

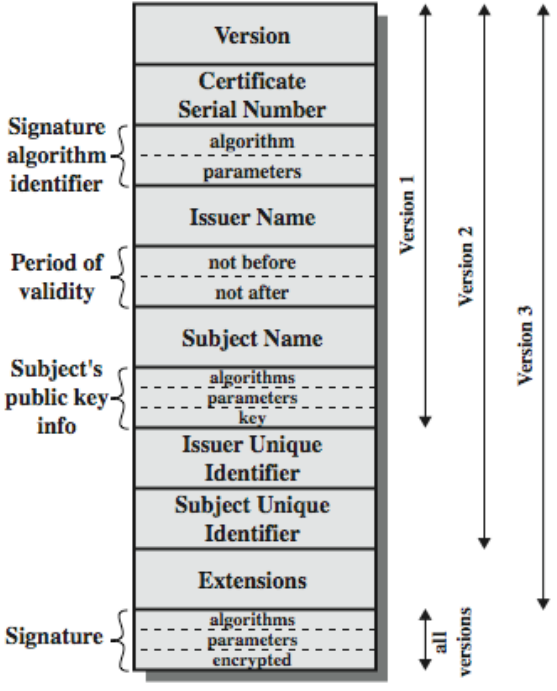
- Each principal requires **once** the CA to generate a certificate, which is kept locally.
  - Timestamps needed to implement out-dating of old certificates
  - CA does not need to keep copies of the certificates
- At any time, A sends its certificate to B; B can decrypt the certificate using CA's public key
  - this ensures that certificate comes from CA
  - if the certificate is not outdated, the PU_A is valid
- If PR_A is taken by an intruder, A generates a new pair and asks the CA a new certificate
  - Until timeout, intruder may be authenticated as A
  - Problem reduced with revocation lists (kept in CA).

X.509 Authentication Service

- part of ITU-T X.500 directory service standards
  - distributed servers maintaining user info database
  - First in 1988, many revisions; X.509v3 in 2002 (RFC 3280)
- defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended
- X.509 certificates are widely used

X.509 Certificates

- issued by a Certification Authority (CA), containing:
  - version (1, 2, or 3)
  - serial number (unique within CA) identifying certificate
  - signature algorithm identifier
  - issuer X.500 name (CA)
  - period of validity (from - to dates)
  - subject X.500 name (name of owner)
  - subject public-key info (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)
- notation CA<<A>> denotes “certificate for A signed by CA”



(a) X.509 Certificate

Certificate Extensions

- key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
  - increasing levels of checks & hence trust

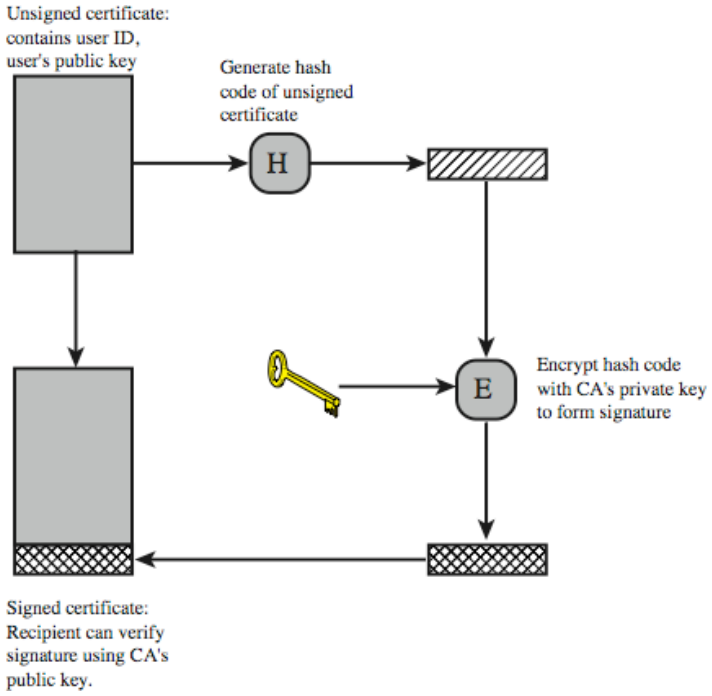
Class	Identity Checks	Usage
1	name/email check	web browsing/email
2	+ enroll/addr check	email, subs, s/w validate
3	+ ID documents	e-banking/service access

- certificate subject and issuer attributes
  - support alternative names (e.g. emails), in alternative formats for certificate subject and/or issuer
- certificate path constraints
  - allow constraints on use of certificates by other CA's

Obtaining a Certificate

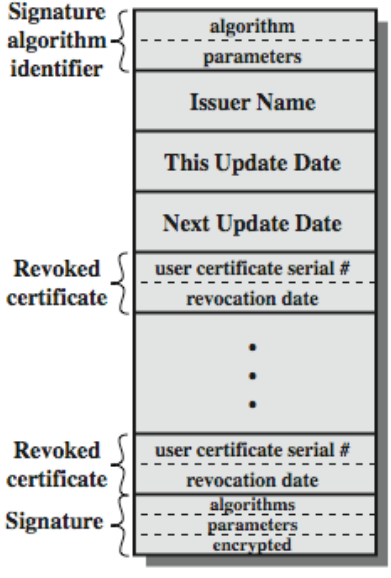
dentro al chip della tessera sanitaria c'è un generatore di chiavi RSA

- any user with access to CA can get any certificate from it
- The user
  - generates a private/public key
  - Sends the public key to the CA in a "certificate request"
  - The CA signs the request and sends back the certificate
- only the CA can generate a certificate
- because cannot be forged, certificates can be placed in a public directory
- can be renewed before it expires



Certificate Revocation

- certificates have a period of validity
- may need to revoke before expiry, eg:
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CA's maintain list of revoked certificates
  - can be published in Certificate Revocation List
- users should check certificates with CA's CRL, or verify the validity of a certificate by asking the CA via OCSP (Online Certificate Status Protocol)



(b) Certificate Revocation List

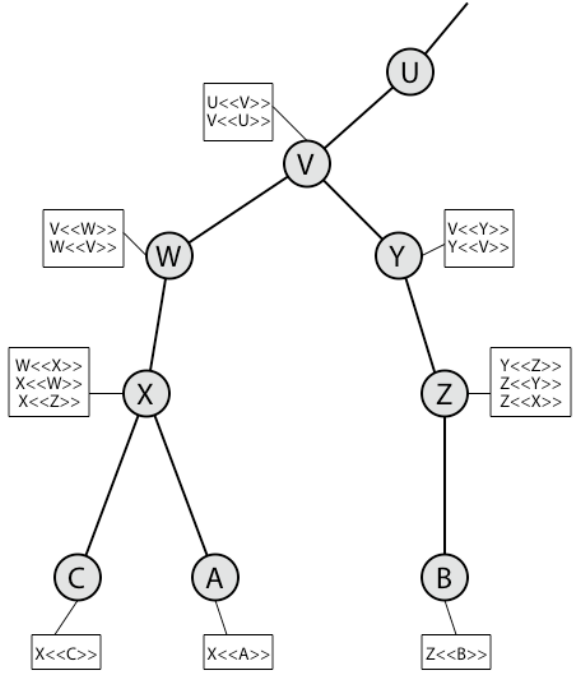
CA Hierarchy

- if both users share a common CA then they are assumed to know its public key
- otherwise CA's must form a hierarchy, for scalability issues
- use certificates linking members of hierarchy to validate other CA's
- each CA has certificates for clients (forward) and parent (backward)
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy

CA Hierarchy Use

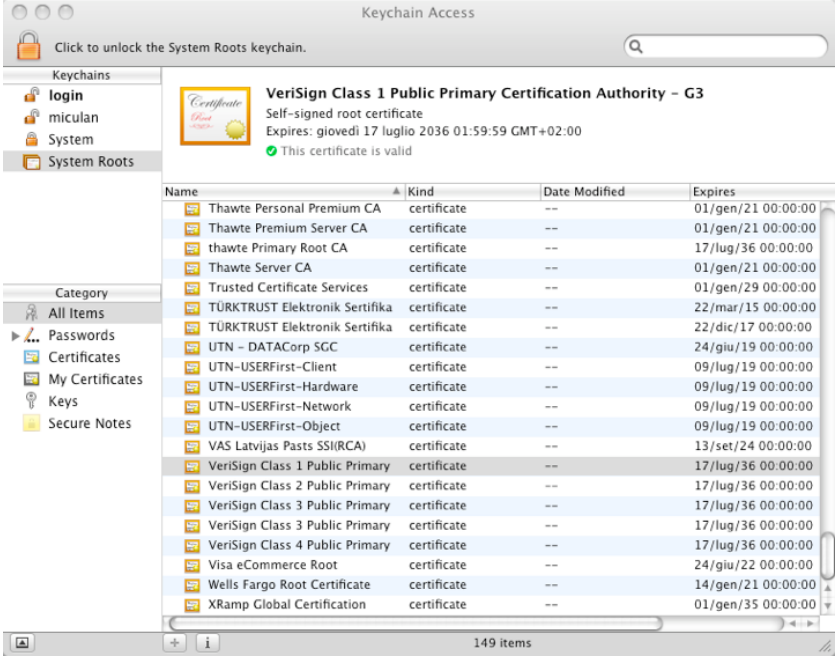
- e.g. user C, knowing only the public key of V, can obtain a verified copy of B's public key:

1. B sends to C a chain of certificates, U<<V>>, V<<Y>>, Y<<Z>>, Z<<B>>.
2. C validates V<<Y>> using the public key of V. (U<<V>> is not needed)
3. C extracts the pub key of Y from V<<Y>>.
4. C validates Y<<Z>> using the pub key of Y.
5. C extracts the pub key of Z from Y<<Z>>.
6. C validates Z<<B>> using the pub key of Z.
7. C extracts the pub key of B from Z<<B>>



Root Certification Authorities

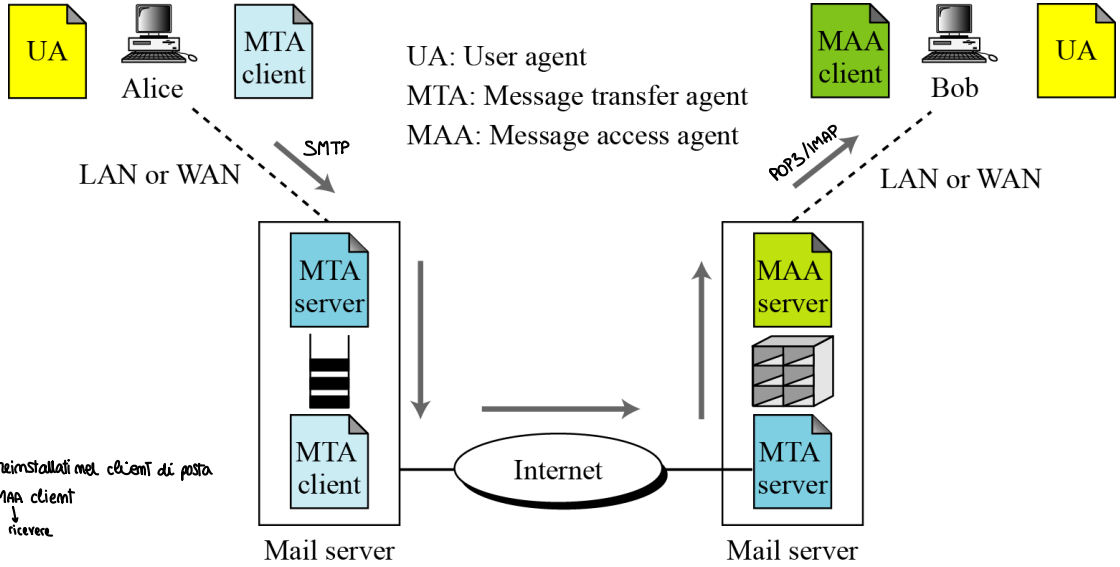
- Root CA are **self-certified**: they sign their own certificate
- These certificates must be distributed in secure way
- Usually come with the Operating System or the applications
  - you must trust them if you trust your OS or app



Email Security

Email è una rete overlay con molti switch di livello 7 (srv) che ci lavorano e smistano il traffico (e-mail) → rete a commutaz. di pacchetto

- email is one of the most widely used and regarded network services



IMAP e SMTP sono preinstallati nel client di posta contiene sia MTA e MAA client
↓ inviare ↓ ricevere

Email Security

- in normal email, message contents are not secure
  - may be inspected and modified either in transit, or by suitably privileged users on intermediate or destination system
  - easy to impersonate any user
  - Much like holiday postcards
- Possible email Security Enhancements
  - **confidentiality**: protection from disclosure
  - **authentication**: of sender of message
  - **message integrity**: protection from modification
  - **non-repudiation of origin**: protection from denial by sender
  - **non-repudiation of destination**: protection from denial of receiver

Basic structure of email security

- Sending an e-mail is a one-time activity
  - There is no “session”, no mutual exchange
- In e-mail security:
  - the sender of the message needs to include the identifiers of the algorithms used in the message.
  - Some public-key algorithms must be used
  - the encryption/decryption can be done using a symmetric-key algorithm, but the secret key to decrypt the message is encrypted with the public key of the receiver and is sent with the message.

PGP

- **Pretty Good Privacy (PGP)**
  - *de facto* standard for secure email
  - developed by Phil Zimmermann in the 90's
    - selected best available crypto algs to use
    - integrated into a single program
  - on Unix, PC, Macintosh and other systems
  - originally free, now also have commercial versions

PGP Operation - Authentication

1. sender creates message
2. use SHA-1 to generate 160-bit hash of message
3. signed hash with RSA using sender's private key, and is attached to message → messaggio in chiaro mandato con la firma
4. receiver uses RSA with sender's public key to decrypt and recover hash code
   ↳ destinatario la conosce già
5. receiver verifies received message using hash of it and compares with decrypted hash code

PGP Operation - Confidentiality

1. sender generates message and 128-bit random number as session key for it
2. encrypt message using CAST-128 / IDEA / 3DES in CBC mode with session key
   ↳ AES non era ancora stato inventato
3. session key encrypted using RSA with recipient's public key, & attached to msg → chiave di sessione passata assieme al messaggio
   - Recent versions support also ElGamal
4. receiver uses RSA with private key to decrypt and recover session key
5. session key is used to decrypt message

facendo così, per i SAV è impossibile leggere il contenuto delle mail

PGP Operation – Confidentiality & Authentication

- can use both services on same message
  - create signature & attach to message
  - encrypt both message & signature
  - attach RSA/ElGamal encrypted session key

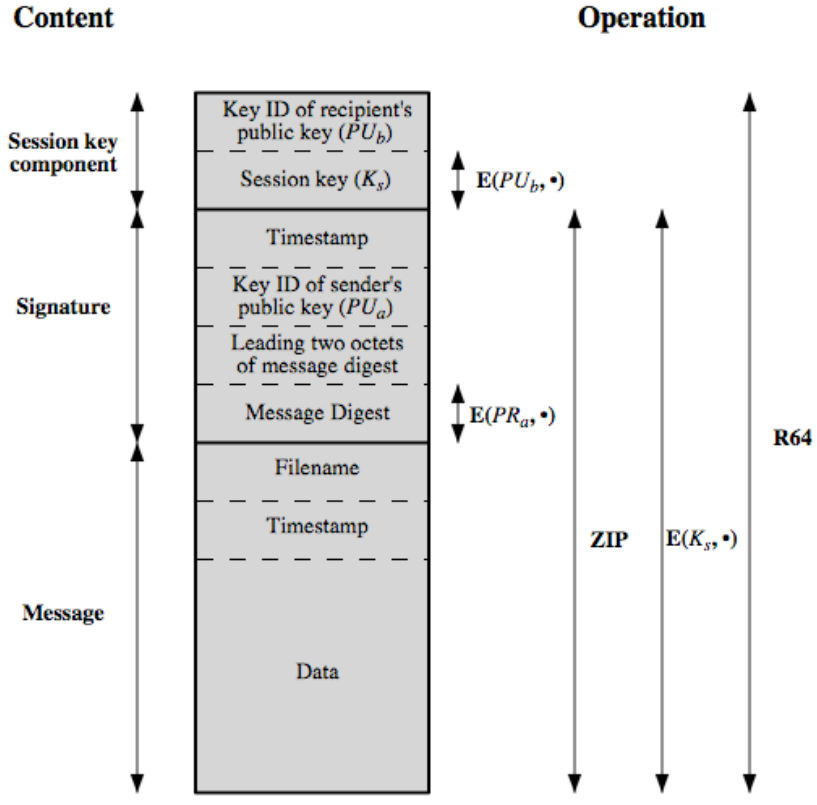
PGP Operation – Compression and Email Compatibility

- by default PGP compresses (using ZIP) message after signing but before encrypting (so can store uncompressed message & signature for later verification)
- for compatibility with email format (which is text only), this raw binary data must be encoded into printable ASCII characters
  - PGP uses radix-64 algorithm: each byte carries only 6 significant bits (i.e. 64 printable chars)
    - maps 3 bytes to 4 printable chars
  - also appends a CRC
- PGP also segments messages if too big

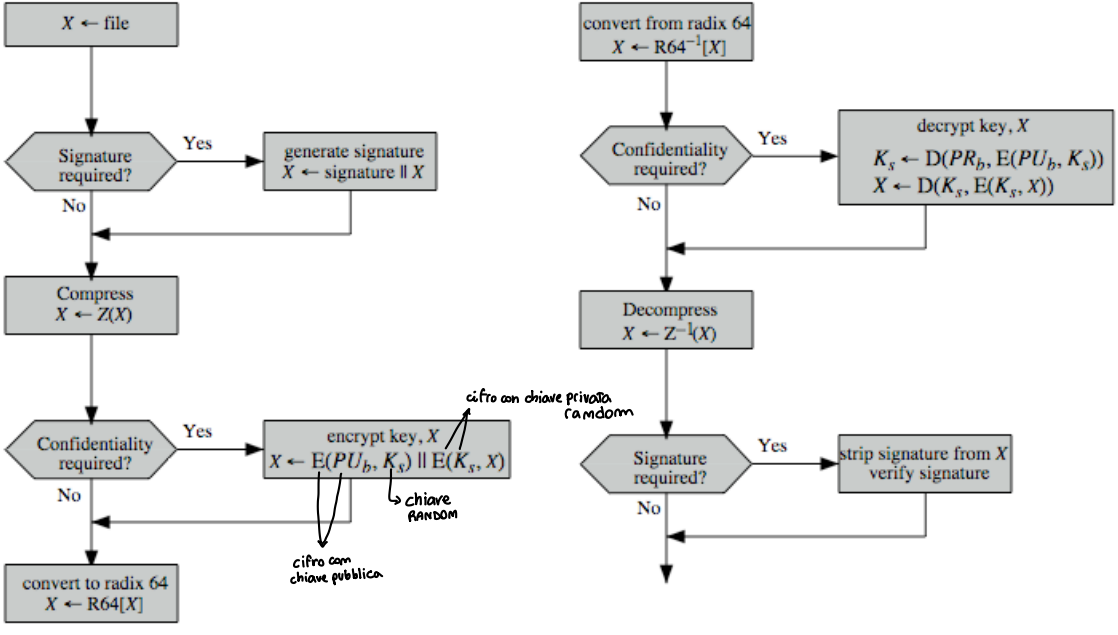
PGP Message Format

Notation:

$E(PU_b, \bullet)$ = encryption with user b's public key
$E(PR_a, \bullet)$ = encryption with user a's private key
$E(K_s, \bullet)$ = encryption with session key
ZIP = Zip compression function
R64 = Radix-64 conversion function



PGP Operation – Summary



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

PGP Session Keys

- need a “session” key for each message
  - of varying sizes: 56-bit DES, 128-bit CAST or IDEA, 168-bit Triple-DES
- generated using ANSI X12.17 mode
- uses random inputs taken from previous uses and from keystroke timing of user

PGP Key Management

es. quando ti si presenta qualcuno tu ti fidi dell'identità che ti dice, non gli chiedi la carta d'identità

- in PGP every user is own "certification authority"
  - can sign keys for users they know directly
- forms a **"web of trust"**
  - trust keys have signed
  - can trust keys others have signed if have a chain of signatures to them
  - a distributed network, instead of hierarchical like X.509
- key ring includes trust indicators
- users can also revoke their keys (e.g. when compromised, or just old)

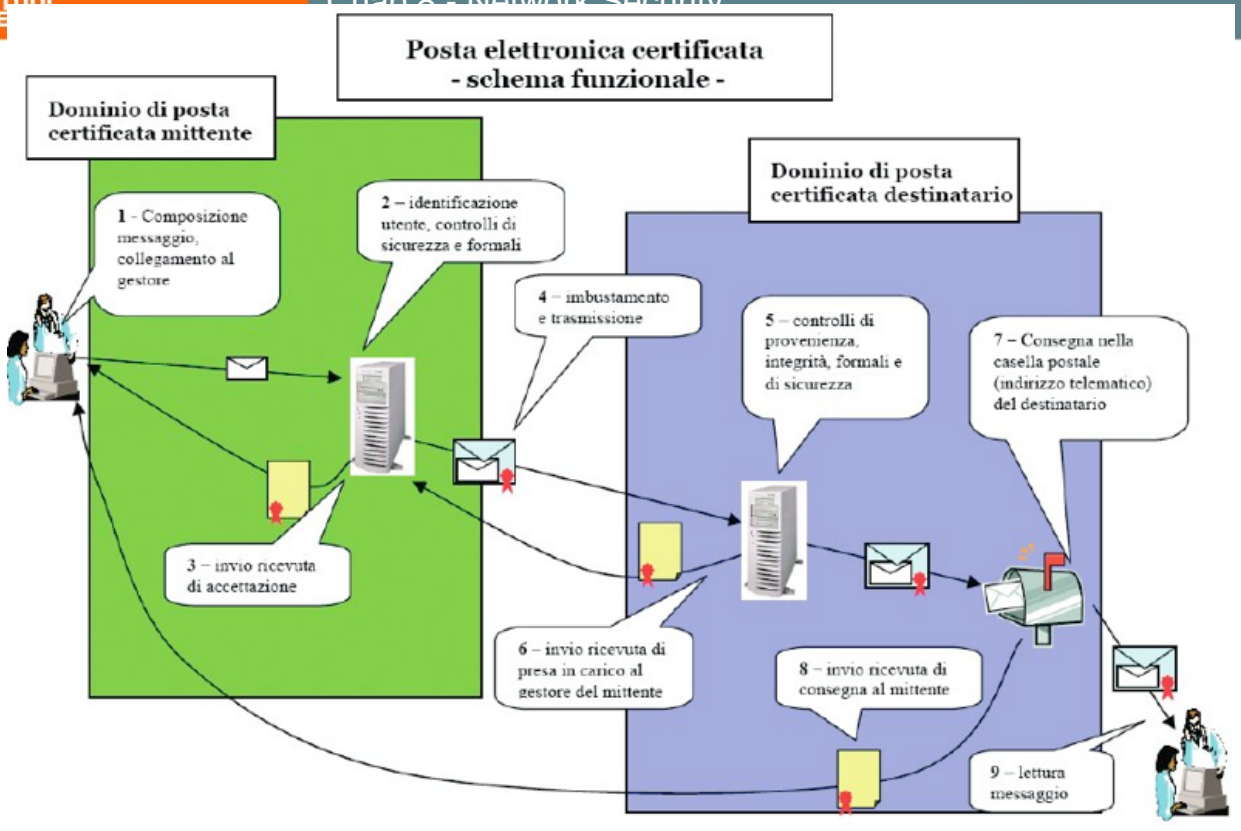
S/MIME (Secure/Multipurpose Internet Mail Extensions)

- original Internet RFC822 email was text only
- MIME provided support for varying content types and multi-part messages with encoding of binary data to textual form
- S/MIME added security enhancements
  - Similar working principles of PGP, but different implementations originally developed by RSA, then standardized by IETF
  - *de jure* standard for secure email (RFC 3851 and 5751)
- Nowadays, most mail agents support S/MIME
  - eg MS Outlook, Mozilla, Thunderbird, Mac Mail…

Posta Elettronica Certificata (PEC)

→ per garantire il non ripudio del ricevente

- security enhancement to S/MIME email
- Adds **non-repudiation of recipient** and **certified sending timestamp** (by the server)
- similar to *registered mail* (posta raccomandata)
- when the message is delivered in recipient's inbox, the sender receives a signed receipt
  - this does not guarantee that the recipient has actually read the email, but she cannot claim that has not received it
- Has legal value, if the servers are officially recognized (there is a public list)


Posta elettronica certificata
- schema funzionale -
Dominio di posta certificata mittente
Dominio di posta certificata destinatario
1 - Composizione messaggio, collegamento al gestore
2 – identificazione utente, controlli di sicurezza e formali
3 – invio ricevuta di accettazione
4 – imbustamento e trasmissione
5 – controlli di provenienza, integrità, formali e di sicurezza
6 – invio ricevuta di presa in carico al gestore del mittente
7 – Consegna nella casella postale (indirizzo telematico) del destinatario
8 – invio ricevuta di consegna al mittente
9 – lettura messaggio

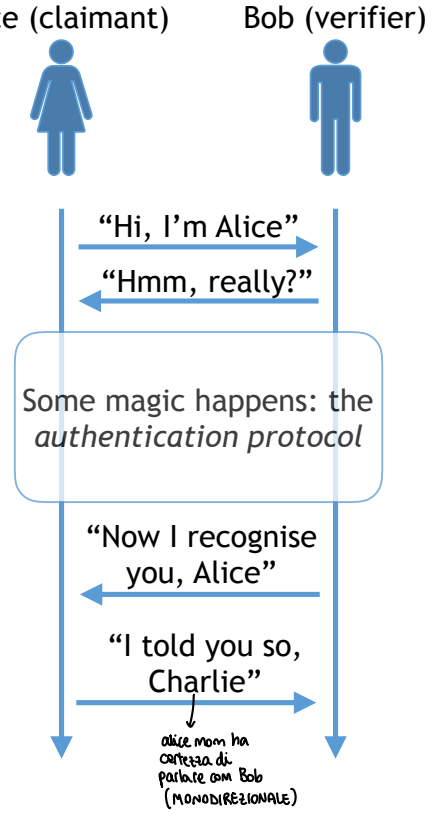
Key Management and Entity Authentication

- **Entity authentication** is a technique designed to let one party prove the identity of another party.
  - An entity can be a person, a process, a client, a server, etc.
  - The entity whose identity needs to be proved is the **claimant**
- The party that tries to prove the identity of the claimant is called the **verifier**.
- have a range of approaches based on the use of public-key encryption
  - Especially for session key distribution
- **need to ensure have correct public keys for other parties**
  - This may be not the case (e.g. outdated/revocated keys)
- various protocols exist using timestamps or nonces (and again, many published protocols found flawed)

Message vs Entity Authentication

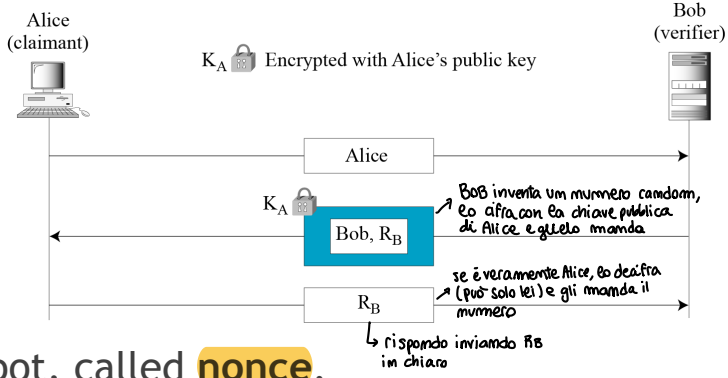
AUTENTICITÀ dei messaggi vs AUTENTICAZIONE di utenti
↳ statico ↳ dinamico

- It is important not to confuse *message* authentication and *entity* authentication
  - Message authentication authenticates one message at once, and might not happen in real time
  - Entity authentication authenticates the claimant for the entire duration of a **session**. protocolli di livello 5
- In this case we say that "the claimant (A) is authenticated for the verifier (B)"
- This usually corresponds to the establishment of a shared secret
- Entity authentication changes the internal knowledge of participants: there is a *before* and an *after*.



Authentication with asymmetric-key encryption

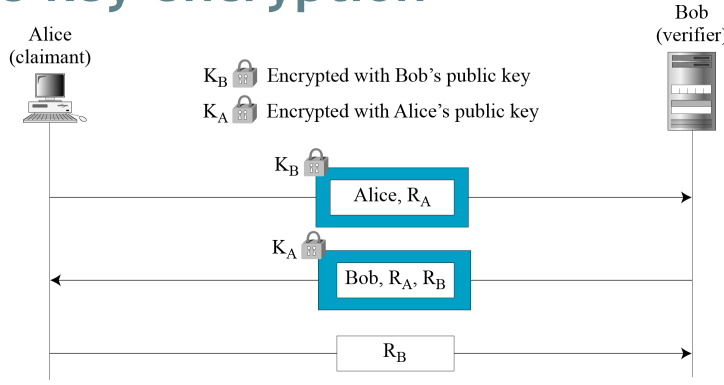
- If we have asymmetric encryption, we can identify a participant by its secret key
- **One-directional authentication protocol**:
  1. A → B: A
  2. B → A: $E\_K_A(B,R_B)$
  3. A → B: R_B
- R_B is a random number generated on the spot, called **nonce**.
- Steps 2-3 are a **challenge-response**: B places a challenge that only the real A can solve.
- Notice that the challenge is posed by the verifier and answered by the claimant
- This does not authenticate B for A.



Authentication with asymmetric-key encryption

- **Bi-directional authentication protocol**:
  1. A → B: $E\_K_B(A, R_A)$
  2. B → A: $E\_K_A(B, R_B, R_A)$
  3. A → B: R_B
- R_A, R_B are both nonces.
- Steps 1-2 are a challenge-response for B
  - This authenticates B for A
- Steps 2-3 are a challenge-response for A, which authenticates A for B.
- Here both parties need to have a public-private key

la challenge per Bob è aprire il messaggio



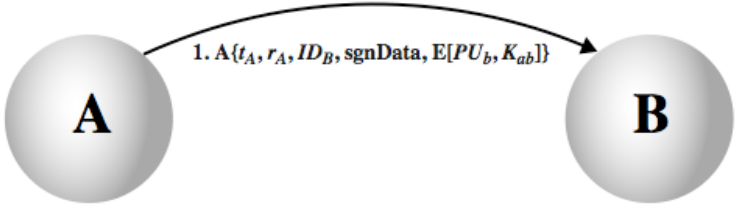
Authentication Procedures in X.509

- X.509 standard defines 3 authentication procedures:
  - One-Way Authentication: for unidirectional messages
  - Two-Way Authentication: interactive session, based on timestamps (require some synchronisation)
  - Three-Way Authentication: interactive sessions, based on nonces, without timestamps
- all use public-key signatures
- Message format (not all fields are always useful):

A{timestamp, nonce, dest, signed data, secret data}

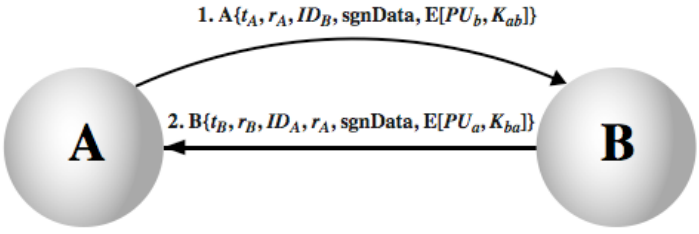
dati firmati da A

X.509: One-Way Authentication



- one message (A → B) used to establish
  - the identity of A and that message is from A
  - message was intended for B
  - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A (A{...})
- may include additional info for B (eg session key)

X.509: Two-Way Authentication

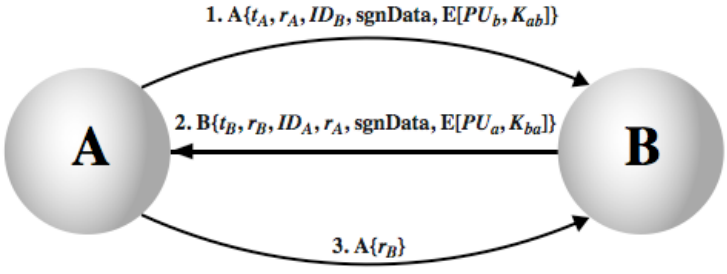


- 2 messages (A → B, B → A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply
- reply includes original nonce from A, also timestamp and nonce from B
- may include additional info for A (e.g. session key)

X.509: Use of timestamps

- Replay attack: An attacker can re-use an old message from A, to induce B to use an old session key K_{ab}
- For avoiding this, timestamp t_A is checked on B:
  - The message is accepted if
    $$|t_B - t_A| < \Delta$$
    where t_B is B's local time when it receives the message from A
- Issues about the use of timestamps:
  - How to choose Δ?
    - If too large, we are more at risk of replay attacks
    - If too small, we can reject many valid messages due to network delays
  - How to keep synchronised A and B's clocks?
    - Hardware clock are often very imprecise
    - There are synchronisation protocols (NTP), but they are not perfect (up to ~5-10ms), and can be attacked as well.

X.509: Three-Way Authentication



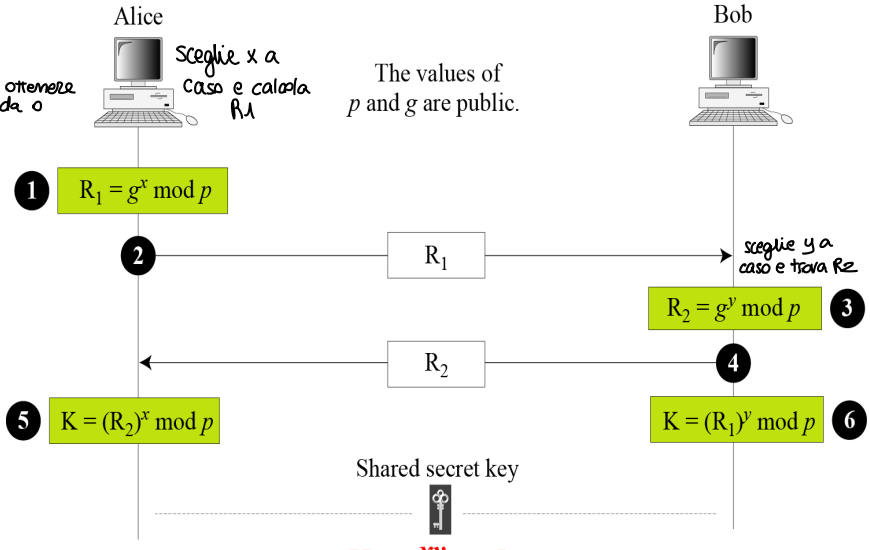
- 3 messages (A → B, B → A, A → B) which enables above authentication without synchronized clocks → faccio DOPPIA CHALLENGE
- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

Diffie-Hellman Key Agreement

- first “public”-key type scheme proposed by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - now know that Williamson (UK CESG) secretly proposed the concept in 1970
- is a practical method for creating a symmetric session key without the need of a third party (“key distribution center”)
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Exchange → è robusto agli attacchi passivi (attaccante osserva)

- all users agree on global parameters:
  - large prime integer or polynomial p
  - g being a primitive root mod p (moltiplicando più volte per se stesso il num riesco ad ottenere i numeri da 0 a p-1)
    - $\{g^i \bmod p \mid 0 \le i \le p-1\} = \{0 \ldots p-1\}$
- each user (eg. A) generates their key
  - chooses a secret key (number): $x < p$
  - compute their **public (or half) key**: $R_1 = g^x \bmod p$
- each user makes public that key R_1



$(R_2)^x = (g^y)^x = g^{yx} = (g^x)^y = (R_1)^y$

Diffie-Hellman Key Exchange

- shared session key for users A & B is K:

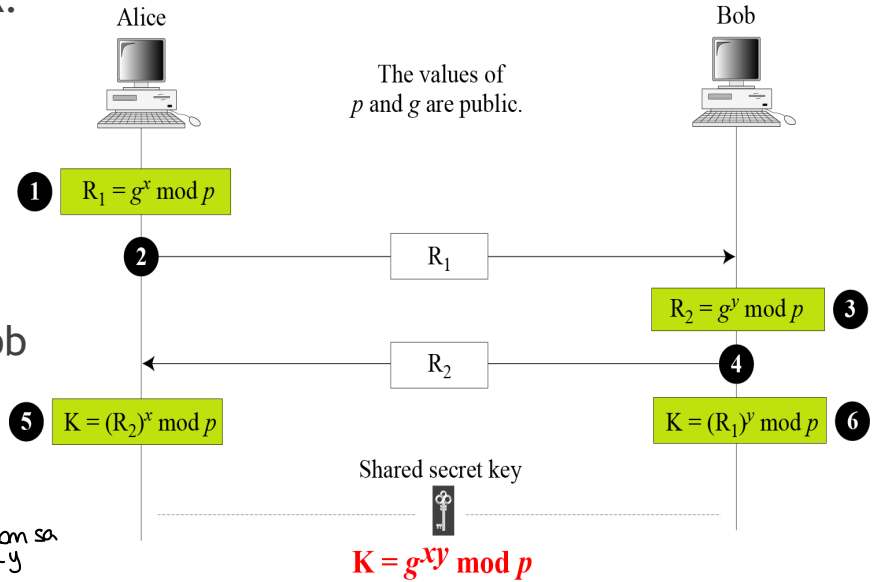
  $K = g^{x*y} \bmod p$

  $= R_1^{\,y} \bmod p$ (which **B** can compute)

  $= R_2^{\,x} \bmod p$ (which **A** can compute)

- Now K can be used as session key in private-key encryption scheme or other schemata between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs x or y, must solve discrete log - hard → $d\log_g(R2) = y$ (log. discreto: quel esponente di g che ottiene R2)

→ l'attaccante non sa me x me y



Diffie-Hellman Key Exchange

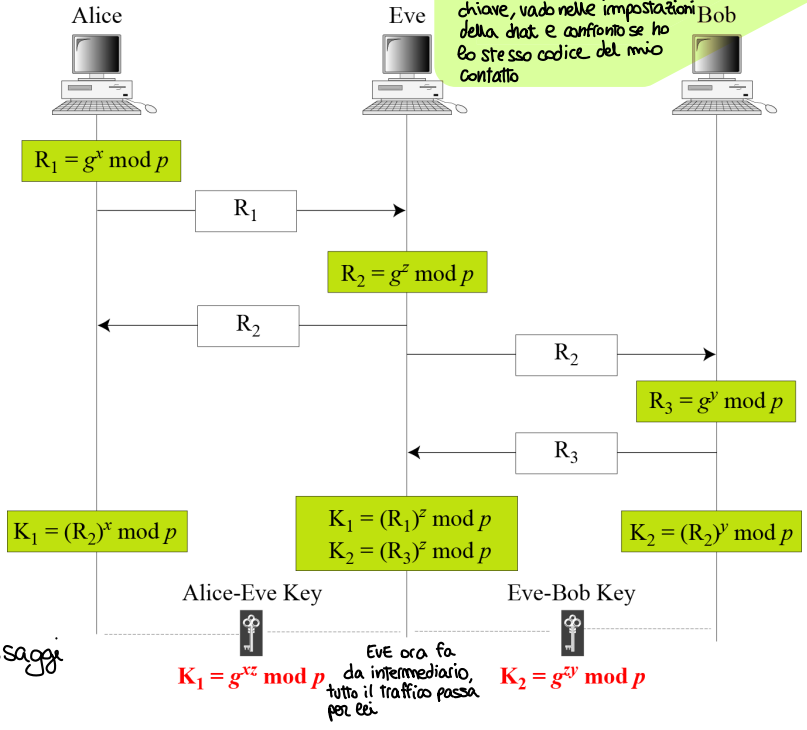
- Example: users Alice & Bob who wish to swap keys:

1. agree on prime $p=353$ and $a=3$
2. select random secret keys:
   - A chooses $x_A=97$, B chooses $x_B=233$
3. compute respective public keys:
   - $R_A=3^{97} \mod 353 = 40$ (Alice)
   - $R_B=3^{233} \mod 353 = 248$ (Bob)
4. compute shared session key as:
   - $K_{AB}= R_B^{x_A} \mod 353 = 248^{97} = 160$ (Alice)
   - $K_{AB}= R_A^{x_B} \mod 353 = 40^{233} = 160$ (Bob)

Man-in-the-middle attack to Diffie-Hellman

può succedere anche con le chat whatsapp e Telegram (i SRV possono fare da MAN IN THE MIDDLE)
↳ per vedere se la chat con una persona è sicura, ovvero entrambi hanno la stessa chiave, vado nelle impostazioni della chat e confronto se ho lo stesso codice del mio contatto

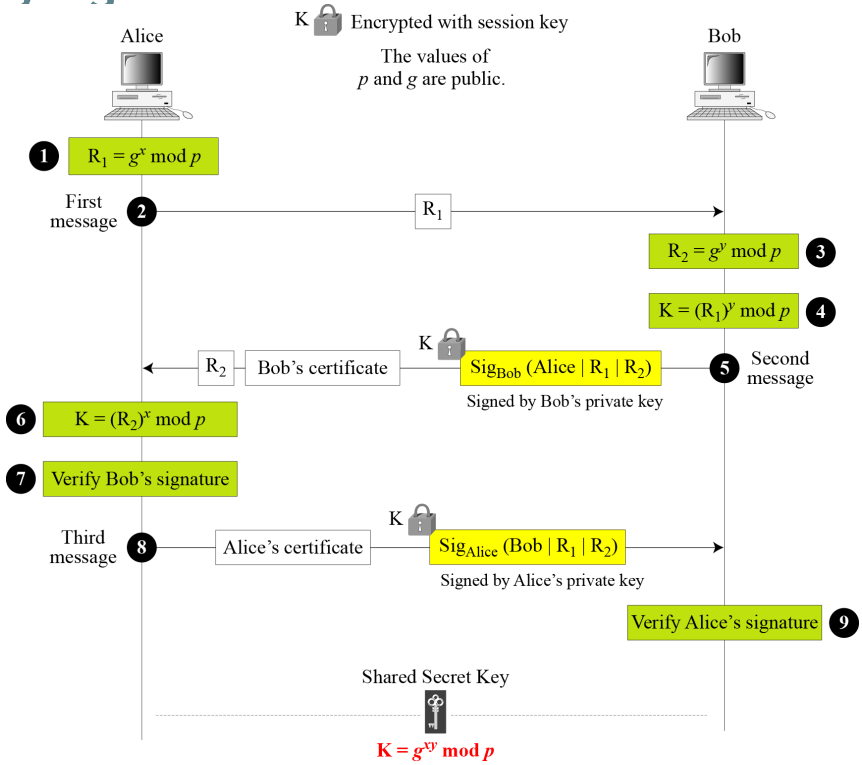
- Plain DH is vulnerable to a **man-in-the-middle** attack, because messages are not authenticated
  - Eve acts as a middle man between Alice and Bob, running two DH exchange in parallel: one with Alice, another with Bob
  - Then she can intercept, decrypt, read/change and re-encrypt all messages
- Notice that this is an active attack - if the attacker can make only passive attacks, it is not a problem



lo schema così non garantisce l'autenticità dei messaggi
↳ gira tutto in chiaro

Safe station-to-station key agreement

- Solution is that DH public keys must be authentication
- Usually keys are signed using public key certificates
  - Here asymmetric encryption and digital signatures come handy
- Similar solutions used in SSL and IPSec



Web Security

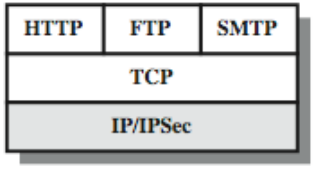
- Web widely used by business, government, individuals
- but Internet & Web are vulnerable
- have a variety of threats
  - integrity
  - confidentiality
  - denial of service
  - authentication
- need added security mechanisms

Approaches to web security

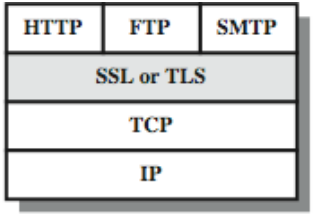
a) Network level is system-wide, but heavy to implement (and must change TCP/IP stack)

b) Transport level is general enough, and can be done in user space

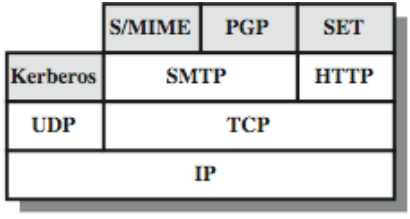
c) Application specific security mechanisms



(a) Network Level



(b) Transport Level



(c) Application Level

Security at transport layer: SSL (Secure Socket Layer) and (Transport Layer Security)

tutto il traffico HTTP gira in CHIARO, perché si appoggia su TCP e nei router è tutto in chiaro

- transport & session layer security service
  - offers integrity and confidentiality, also on traffic flow (somehow)
- uses TCP to provide a reliable end-to-end service
- SSL originally developed by Netscape
- version 3 designed with public input
- subsequently became Internet standard known as **TLS (Transport Layer Security)**

Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011
SSL 3.0	1996	Deprecated in 2015
TLS 1.0	1999	Deprecated in 2020
TLS 1.1	2006	Deprecated in 2020
TLS 1.2	2008	
TLS 1.3	2018	

non definite da uno STANDARD

definite da uno standard

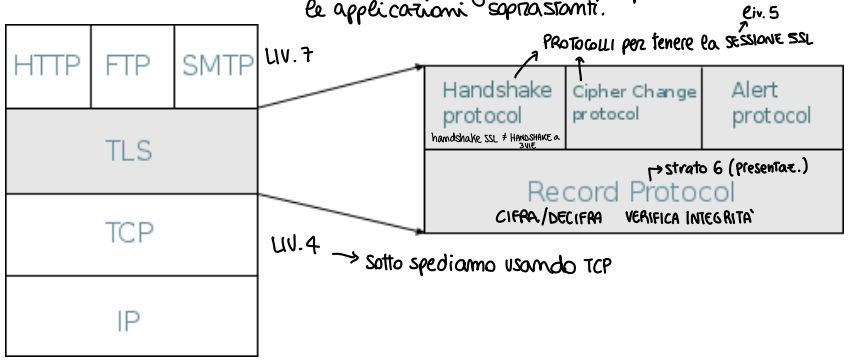
VULNERABILITÀ

è implementato in spazio utente, sono librerie

SSL ≠ TLS anche se spesso si nominano assieme

TLS Architecture

al posto che inventare HTTP con sicurezza, creo delle librerie generiche che possono usare tutte le applicazioni sopra stanti.



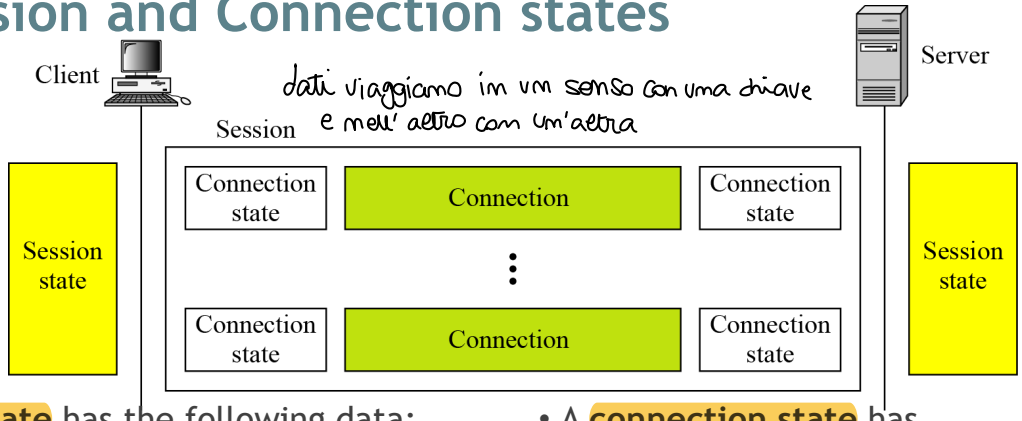
- TLS has two layers of protocols
- **Record Protocol** implements actual security transformation of application data (payload) into the insecure TCP channel, and back
  - Uses TCP sockets for transmission of its data
  - Used directly by applicative protocols (HTTP, IMAP, POP…)
  - Can be seen as a presentation protocol (layer 6)
  - Uses informations kept in the connection state and managed by other protocols
- **Handshake, Change Cyper Spec and Alert Protocols** are used for entity authentication, to set up and manage the informations needed for security services (secret keys)
  - Should be seen as a session protocols (layer 5) (although they use TLS Record)

es. HTTPS è HTTP che gira sulla socket TLS (che usa TCP)

TLS Architecture

- **TLS session**
  - an association between client & server applications
  - created by the Handshake Protocol
  - define a set of cryptographic parameters
  - may be shared by multiple TLS connections
- **TLS connection** → all'interno di una sessione puoi avere più connessioni
  - a transient, peer-to-peer, communication link
  - associated with 1 SSL session

TLS Session and Connection states



- A **session state** has the following data:
  - Session identifier
  - Peer certificate (optional): X.509v3 certificate
  - Compression method
  - Cipher spec: cipher and hash algorithms and data
  - **Master secret**: a 48 byte (384 bit) shared key
- A **connection state** has
  - Server and client random (SR, CR)
  - Six secrets, derived from the Master secret and SR, CR
    - Server write MAC secret, client write MAC secret
    - Server write key, Client write key
    - IV for CBC ciphers
  - Sequence number ($0 - 2^{64}-1$)

TLS session state

Parameter	*Description*
Session ID	A server-chosen 8-bit number defining a session.
Peer Certificate	A certificate of type X509.v3. This parameter may by empty (null).
Compression Method	The compression method.
Cipher Suite	The agreed-upon cipher suite.
Master Secret	The 48-byte secret.
Is resumable	A yes-no flag that allows new connections in an old session.

- The combination of key exchange, hash, and encryption algorithms defines a **cipher suite** for each TLS session.
- Example: SSL_DHE_RSA_WITH_DES_CBC_SHA means "we establish the master secret using ephemeral Diffie-Hellman authenticated with RSA; then the data is encrypted using DES mode CBC, and authenticated with SHA-1"

Diffie Hellman Effimero

SSL cipher suit list

Cipher suite	*Key Exchange*	*Encryption*	*Hash*
SSL_NULL_WITH_NULL_NULL	NULL	NULL	NULL
SSL_RSA_WITH_NULL_MD5	RSA	NULL	MD5
SSL_RSA_WITH_NULL_SHA	RSA	NULL	SHA-1
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4	MD5
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4	SHA-1
SSL_RSA_WITH_IDEA_CBC_SHA	RSA	IDEA	SHA-1
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES	SHA-1
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES	SHA-1
SSL_DH_anon_WITH_RC4_128_MD5	DH_anon	RC4	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH_anon	DES	SHA-1
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH_anon	3DES	SHA-1
SSL_DHE_RSA_WITH_DES_CBC_SHA	DHE_RSA	DES	SHA-1
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES	SHA-1
SSL_DHE_DSS_WITH_DES_CBC_SHA	DHE_DSS	DES	SHA-1
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE_DSS	3DES	SHA-1
SSL_DH_RSA_WITH_DES_CBC_SHA	DH_RSA	DES	SHA-1
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES	SHA-1
SSL_DH_DSS_WITH_DES_CBC_SHA	DH_DSS	DES	SHA-1
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES	SHA-1
SSL_FORTEZZA_DMS_WITH_NULL_SHA	Fortezza	NULL	SHA-1
SSL_FORTEZZA_DMS_WITH_FORTEZZA_CBC_SHA	Fortezza	Fortezza	SHA-1
SSL_FORTEZZA_DMS_WITH_RC4_128_SHA	Fortezza	RC4	SHA-1

TLS cipher suites

- TLS 1.0-1.2:

Key exchange/ agreement	Authentication	Data ciphers	Message authentication
RSA	RSA	RC4	Hash-based MD5
Diffie–Hellman	DSA	Triple DES	SHA hash function
ECDH	ECDSA	AES	
SRP		IDEA	
PSK		DES	
		Camellia	

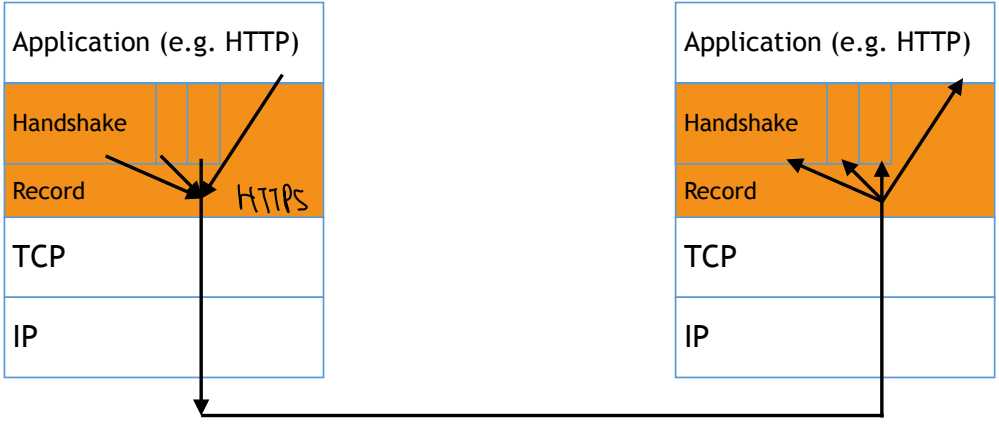
- Example: *TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256* means "we establish the master secret using elliptic curve Diffie-Hellman authenticated with RSA; then the data is encrypted using AES mode CBC with keys of 128 bits, and authenticated with SHA-256"
- In TLS 1.3, many legacy algorithms have been dropped in an effort to make the protocol more secure

Record Protocol Services

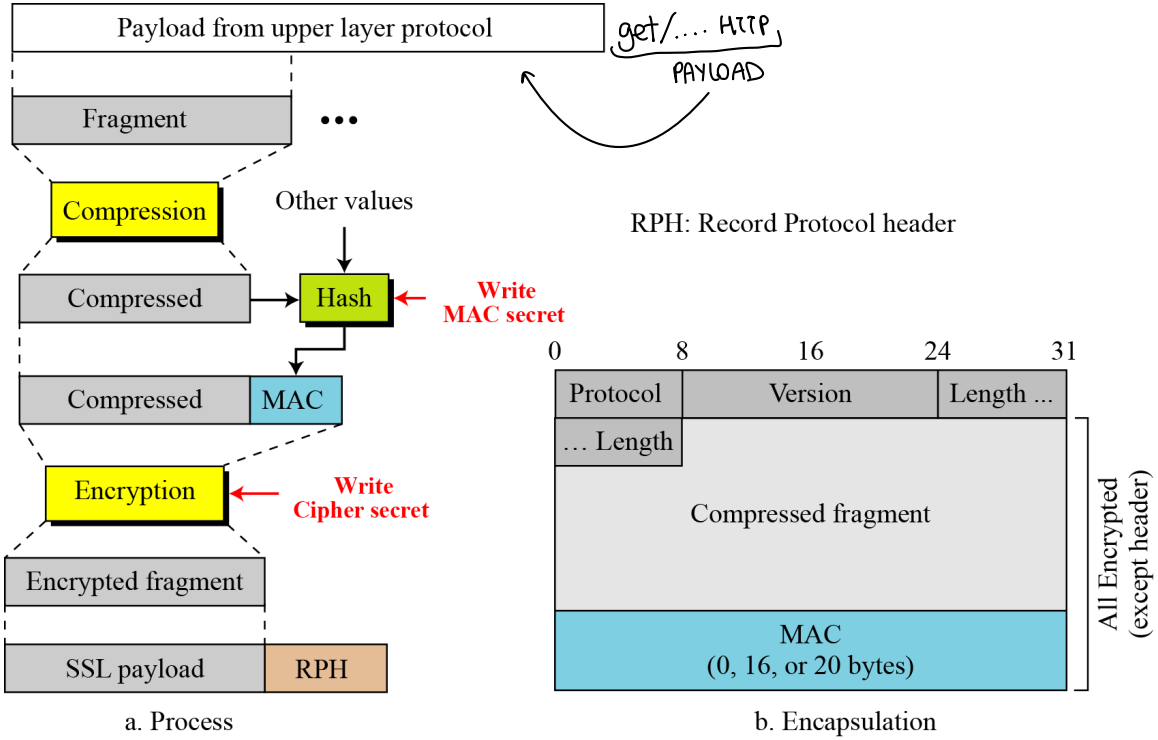
- TLS Record implements the real security transformations, using connection-specific information set up from session keys
- **message integrity**
  - using a MAC with shared secret key
  - similar to HMAC but with different padding
  - with anti-replay mechanisms
- **confidentiality**
  - using symmetric encryption with a shared secret key defined by Handshake Protocol
  - E.g.: NULL, AES, RC4, DES, 3DES…
  - message is fragmented (fragments up to 2^{14} = 16384 byte) and compressed before encryption

TLS Record Protocol

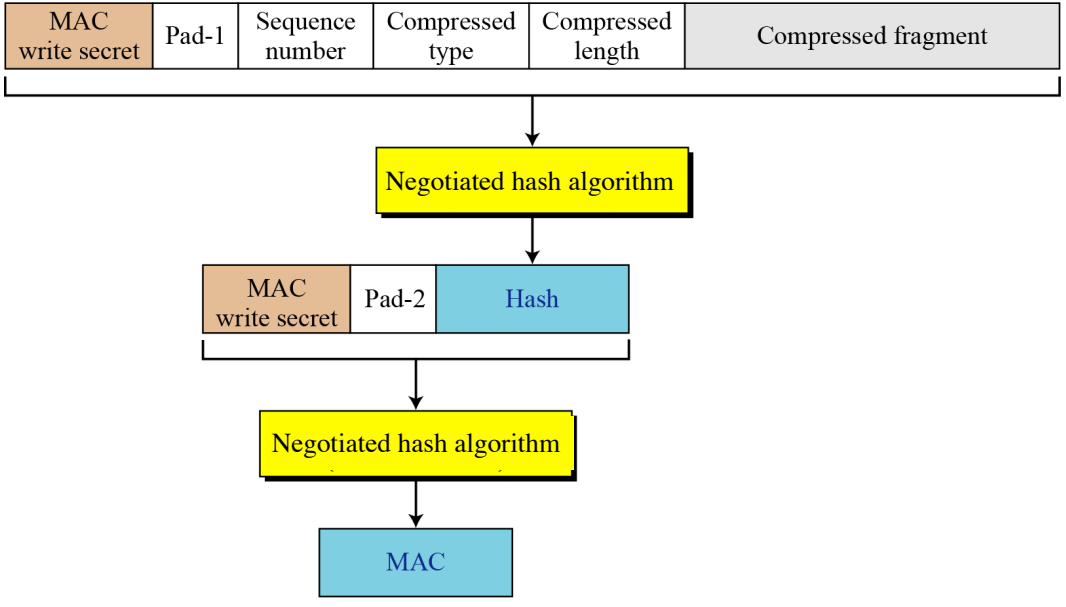
- TLS Record implements the real security transformations, using connection-specific information set up from session keys



SSL/TLS Record Protocol



Record protocol: calculation of MAC

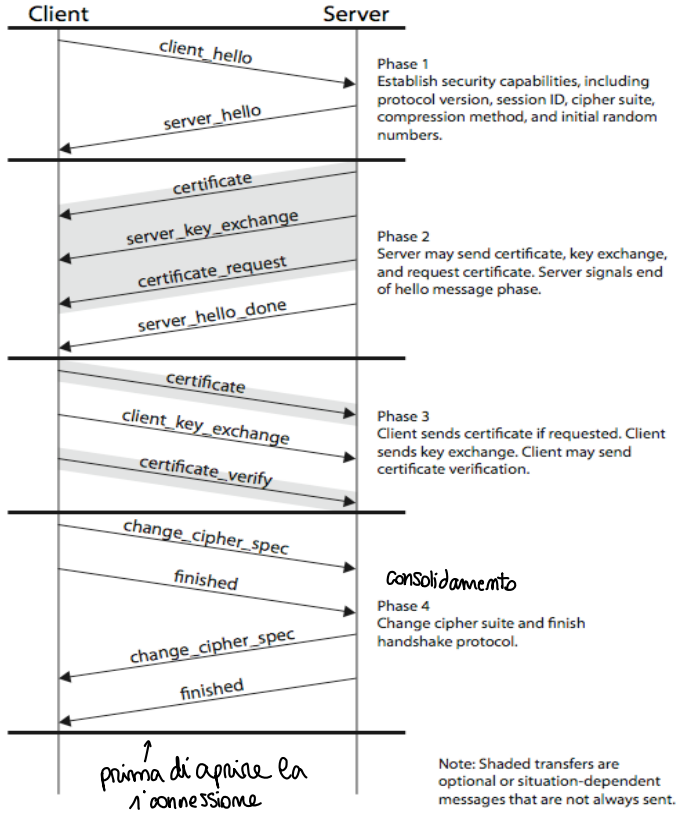


- Notice that the sequence number is used in the calculation of the MAC, in order to avoid replay attacks

Handshake Protocol

↳ determina il MASTER SECRET

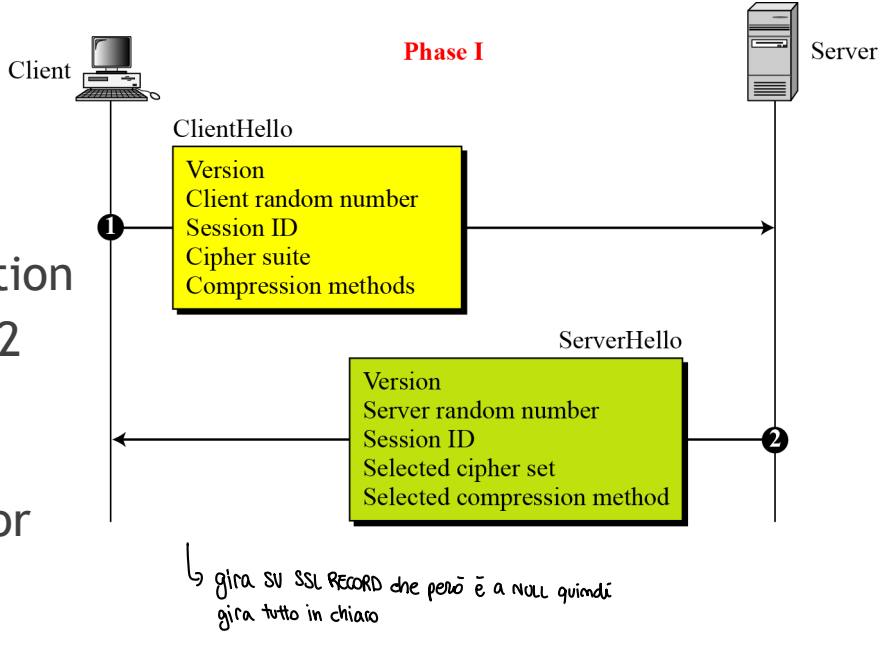
- establishes session data
- allows server & client to:
  - authenticate each other
  - to negotiate encryption & MAC algorithms
  - to negotiate cryptographic keys to be used
- comprises a series of messages in phases
  1. Establish Security Capabilities
  2. Server Authentication and Key Exchange (e.g. for Diffie-Hellman)
  3. Client Authentication and Key Exchange
  4. Finish



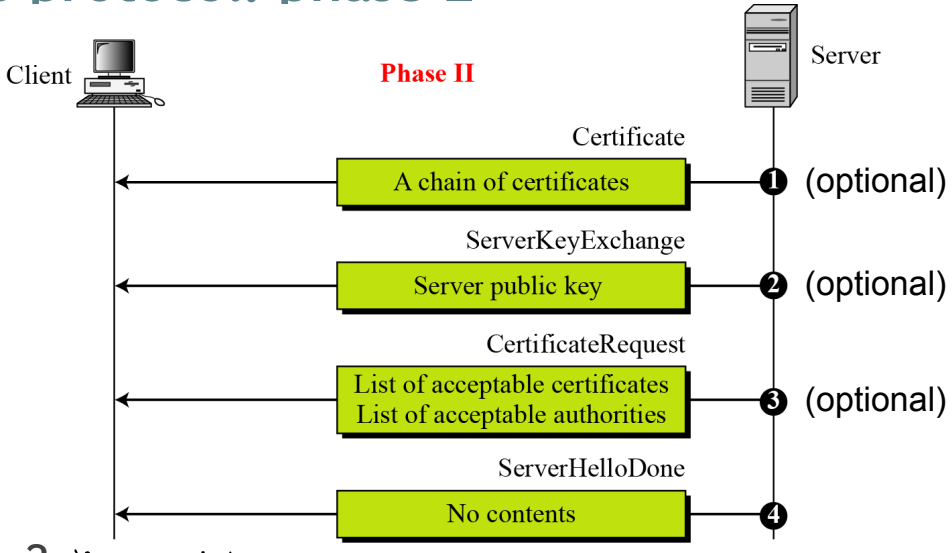
Handshake protocol: phase 1

- After Phase 1, the client and server know and have decided
  - The version of SSL/TLS
  - The algorithms for key exchange, message authentication, and encryption
    - This decides how phases 2 and 3 are done
  - The compression method
  - The two random numbers for key generation

"decidiamo le regole del gioco"

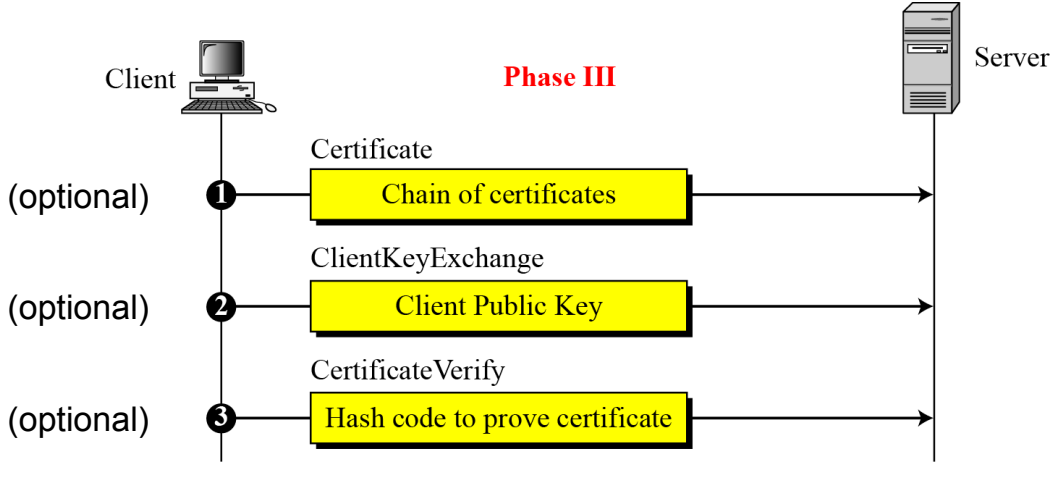


Handshake protocol: phase 2



- After Phase 2 il server si è AUTENTICATO al client
  - The server is authenticated to the client.
  - The client knows the public key of the server if required.

Handshake protocol: phase 3



- After Phase 3
  - The client is authenticated for the server.
  - Both the client and the server know the pre-master secret.

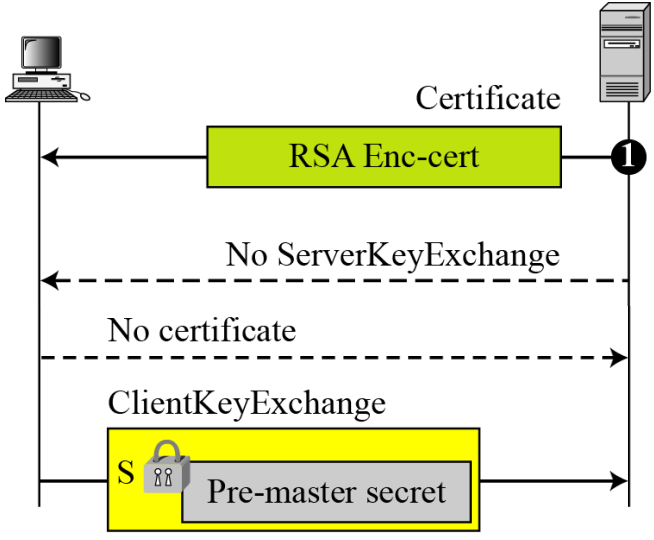
ora possono calcolare il master secret

Handshake protocol: phases 2-3, case RSA

- **RSA**: key is generated by client and sent to server encrypted with its public RSA key.
- Server need to have a certificate, but client does not
- Most common situation

IX.509

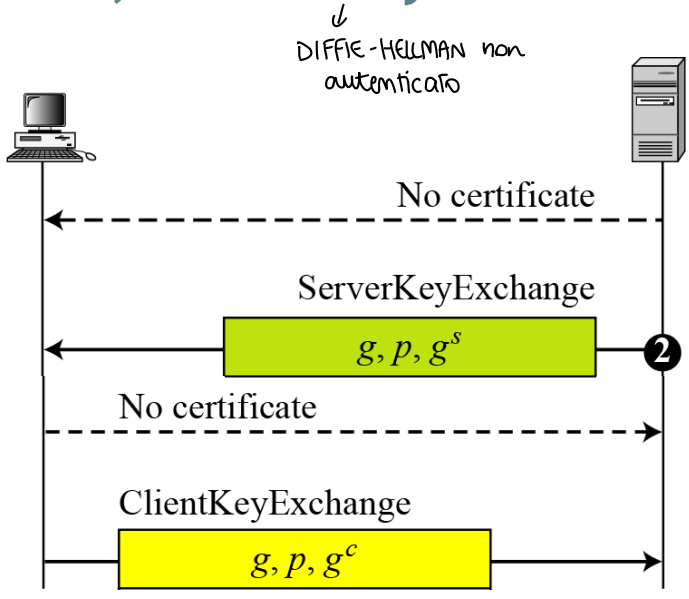
ora il server è autenticato per il client
il client non è autenticato



a. RSA

Handshake protocol: phases 2-3, case Anonymous D-H

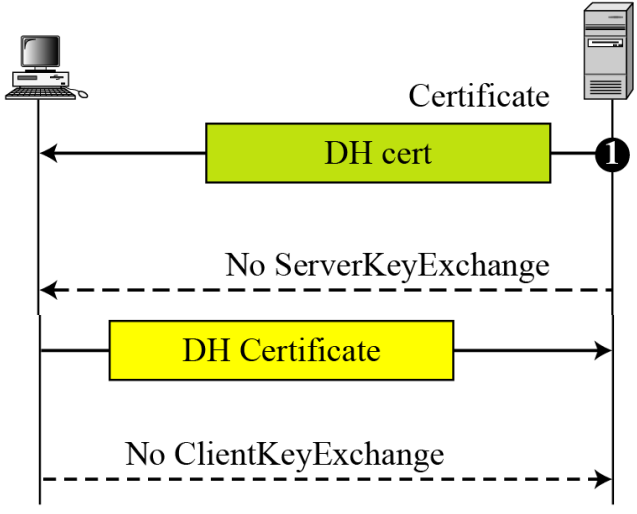
- Anonymous D-H: D-H without authentication.
- Vulnerable to MITM attacks
- No certifications are required
- Admitted until TLS 1.2
- Not available in TLS 1.3



Handshake protocol: phases 2-3, case Fixed D-H

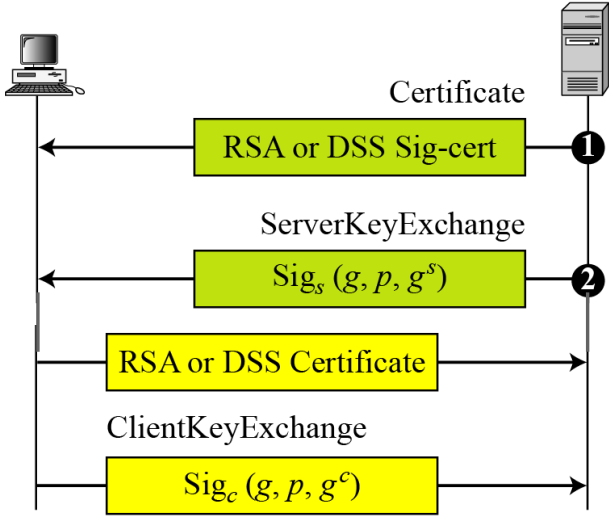
↳ DIFFIE HELLMAN FISSATO

- Fixed Diffie-Hellman: D-H where fixed public parts are contained in certificates (and signed by CA).
- Generates a fixed key for each pair of peers, even between different sessions

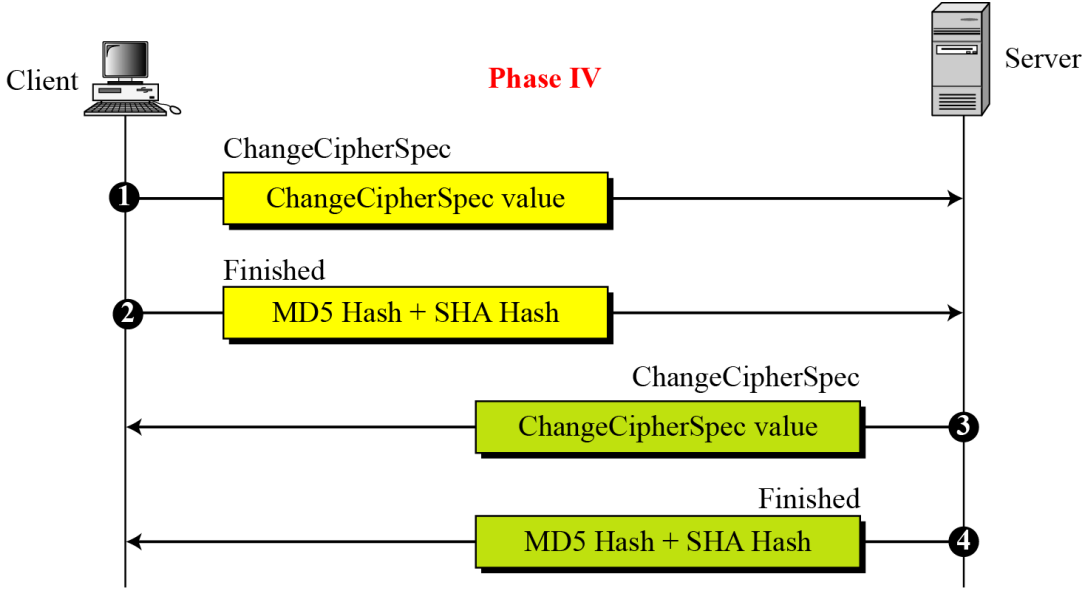


Handshake protocol: phases 2-3, case Ephemeral D-H

- Ephemeral D-H: D-H using random numbers, sent signed with RSA or DSS
- Most robust, best choice (when possible)
- Requires both the client and the server to have a X.509 certificate



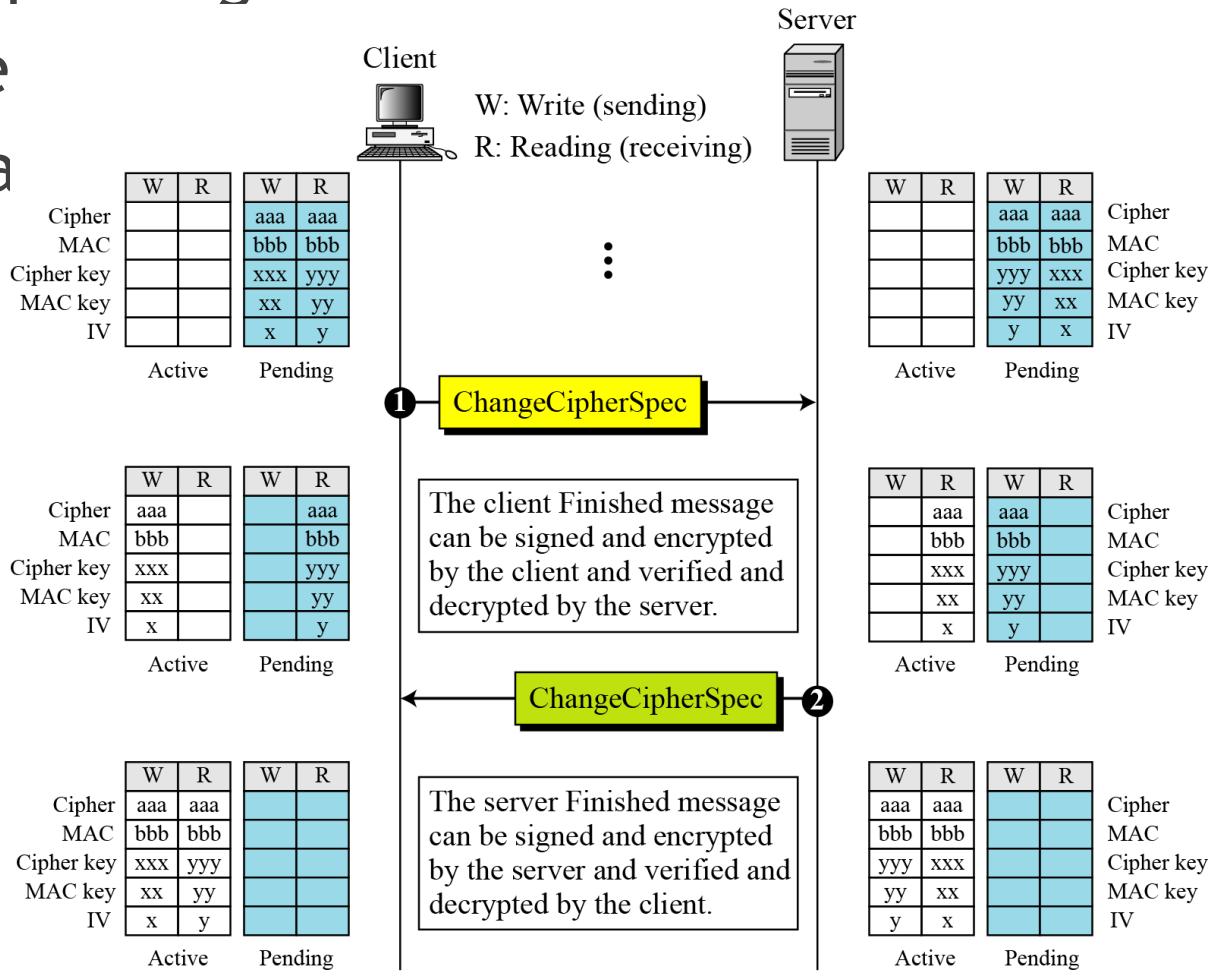
Handshake (ChangeCypherSpec) protocol: phase 4



After Phase IV, the client and server are ready to exchange data.

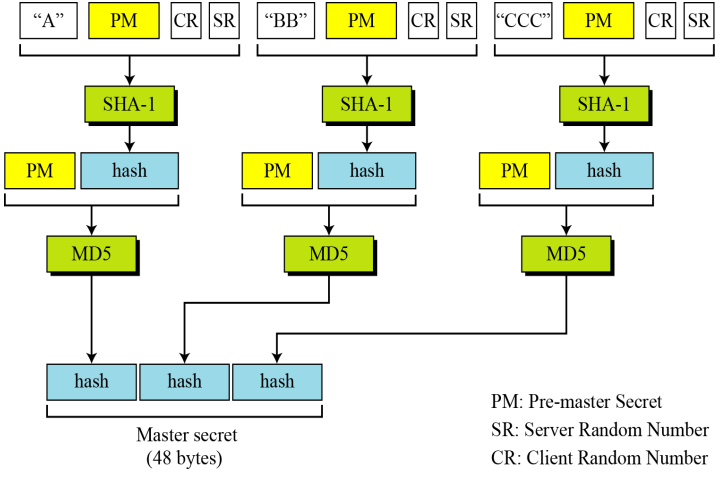
SSL Change Cipher Spec

- a single message, causes pending state to become current
- hence updating the ciphe
- Sequence number is resta

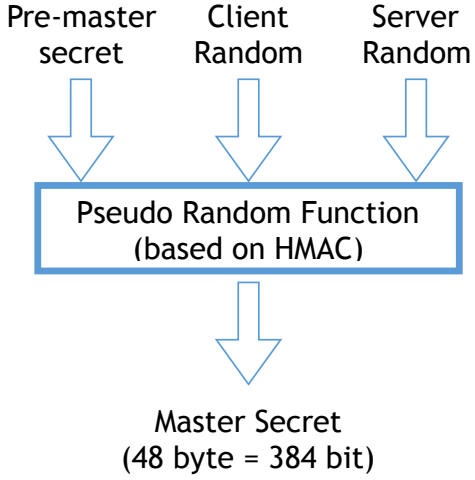


SSL/TLS master secret generation

SSL



TLS

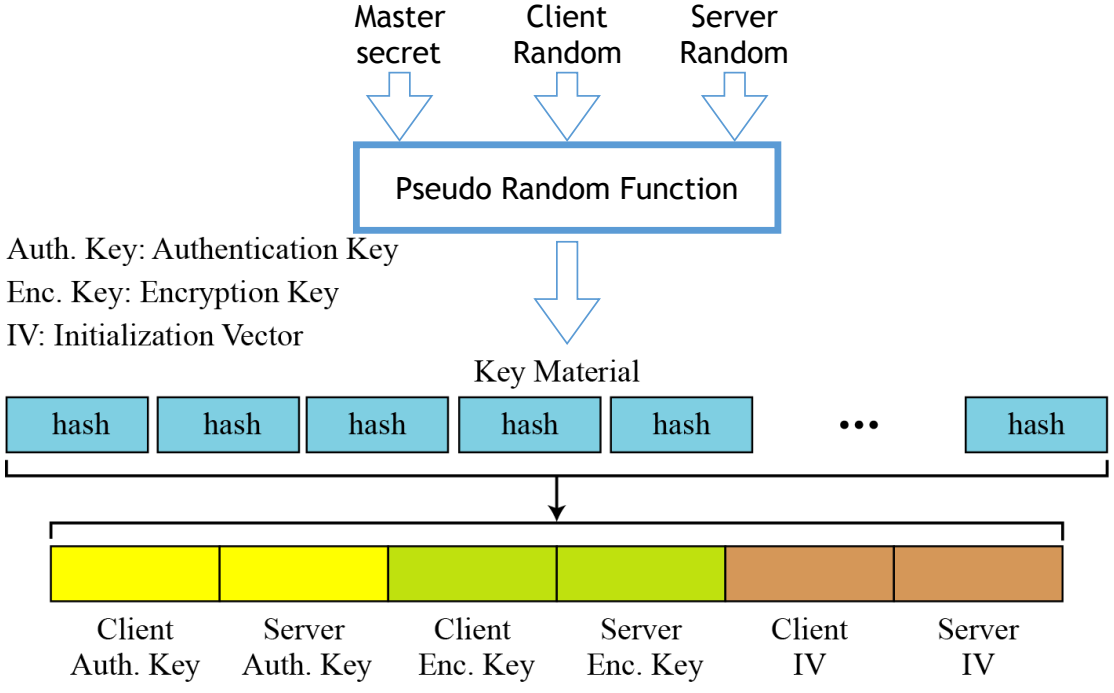


TLS Connection state

- After created a session, one or more **connections** can be established
- For each of them, secrets and IVs are derived from the Master Secret (from the session state) and new Server and Client Randoms, CR during connection handshake

Parameter	*Description*
Server and client random numbers	A sequence of bytes chosen by the server and client for each connection.
Server write MAC secret	The outbound server MAC key for message integrity. The server uses it to sign; the client uses it to verify.
Client write MAC secret	The outbound client MAC key for message integrity. The client uses it to sign; the server uses it to verify.
Server write secret	The outbound server encryption key for message integrity.
Client write secret	The outbound client encryption key for message integrity.
Initialization vectors	The block ciphers in CBC mode use initialization vectors (IVs). One initialization vector is defined for each cipher key during the negotiation, which is used for the first block exchange. The final cipher text from a block is used as the IV for the next block.
Sequence numbers	Each party has a sequence number. The sequence number starts from 0 and increments. It must not exceed $2^{64} - 1$.

TLS connection secrets generation



Porting applications to SSL/TLS

↳come lo uso sulle porte?

- Not difficult - but it requires to access and modify the source code!
- Some calls are specific, other are very similar to sockets
  - SSL_library_init(3), SSL_CTX_new(3): sets various options regarding certificates, algorithms etc.
  - SSL_new(3): creates session (SSL object)
  - SSL_set_fd(3): associate network to object
  - SSL_accept(3), SSL_connect(3): TLS/SSL handshake (from server, from client)
  - SSL_read(3), SSL_write(3): read and write data on the TLS/SSL connection.
  - SSL_shutdown(3): close the TLS/SSL connection.

To Conclude

To conclude…

- This is the end of the course, but not of the story

- In this course we have seen lots of network aspects, fundamental for your professional career
  - The ISO/OSI architecture
  - The main problems in the design, construction and management of computer networks…
  - … and how these are solved in the TCP/IP suite
  - Security issues of computer communications, and some solutions

What's next

- We haven't seen
  - Specific application level protocols and applications
    - See the various specific courses
    - "Distributed Systems" at the Master, for general design and implementations issues and solutions
  - Other security issues
    - See "Computer Network Security" at the Master
  - Quality of service, multimedia applications…
  - Solutions for specific scenarios (e.g. routing for IoT, mobile, multimedia…)
  - Formal models
    - See "Semantica e Concorrenza"

But overall

- Networks will be every day more complex
  - New problems, new application domains
  - New technologies, protocols, solutions will be added over the old ones - which will refuse to die anyway
- No course can describe all the possible evolutions
  - Most of the nowadays network were not even imagined when the basic protocols have been defined

- **The best is yet to come**. Keep learning. You can never settle.


We’ve got a city to burn

And finally...

Thanks for staying with me, until the very end.