

# HOW TO D'amore

## Ricorda:

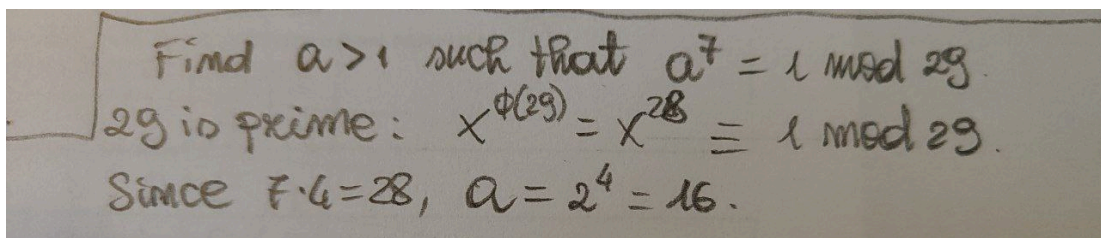
- Quanto ti chiede tipo “come faresti a fare questa cosa”, non limitarti a pensare a quello che hai studiato ma cerca anche di ragionare intuitivamente e di pensare come viene risolto quel problema nel mondo vero. Esempi:
  - Come si evita che si faccia un dictionary attack sulle password di un utente?  
Risposta possibile: metti un timeout a seguito di un numero di richieste consecutive sbagliate.
- D'amore vuole che scrivi e che scrivi tanto. Quindi parla di tutto quello che ti viene in mente.
- Ricorda: Quando nelle iptables dice “tutte gli altri pacchetti devono essere scartati” devi farlo manualmente, così  
`iptables -A INPUT -d 192.168.1.254 -s 0/0 -j DROP`  
`iptables -A OUTPUT -s 192.168.1.254 -d 0/0 -j DROP`

## Domande possibili d'amore all'esame

- Come generalizzare diffie hellman con più partecipanti (per esempio  $n=3$ )

## Tricks

- Euler's theorem practical use



Find  $a > 1$  such that  $a^7 \equiv 1 \pmod{29}$ .  
29 is prime:  $x^{\phi(29)} = x^{28} \equiv 1 \pmod{29}$ .  
Since  $7 \cdot 4 = 28$ ,  $a = 2^4 = 16$ .

## Divisibility Bézout's Lemma

$$ax + by = \gcd(a, b),$$

for some  $x$  and  $y$

Dr. Meenakshi Rana

$$\sigma^{\tau} \Lambda^{\lambda} \varphi_{\mu}$$

- Shamir secret sharing

### 4. Shamir secret sharing [5 points]

4.1. In a Shamir scheme  $(2, 5)$  users got the points  $A=(1, 5)$ ,  $B=(2, 0)$ ,  $C=(3, 2)$ ,  $D=(4, 4)$  and  $E=(5, 6)$ . Assume to work on  $GF(7)$ . Reconstruct and return the secret starting from the points A and B. [3 points]

4.2. Reconstruct again the secret starting from D and E, and verify that the secret is the same. [2 points]

$$4.1) \quad \mathcal{L}(x) = \sum_{j=1}^K y_j \ell_j(x) \quad \ell_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^K \frac{x - x_i}{x_j - x_i} \quad K=2 \quad m=5 \\ GF(7)$$

$$\mathcal{L}(x) = 5 \left[ \frac{x-2}{1-2} \right] + 0 \cdot [ ] = -5[x-2] = -5x + 10.$$

$$S = \mathcal{L}(0) \bmod 7 = 10 \bmod 7 = 3.$$

- Inverso modulare

Inizializzi A B T1 e T2

A = 64

B = 47

Perché vuoi calcolare l'inverso di 47 mod 64

T1 = 1

T2 = 0

E poi calcoli Q, R e T (con le formule che ho scritto sotto i valori) T  
è uguale a T1-T2Q

edited 16:48

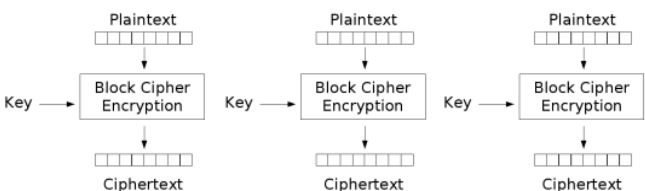
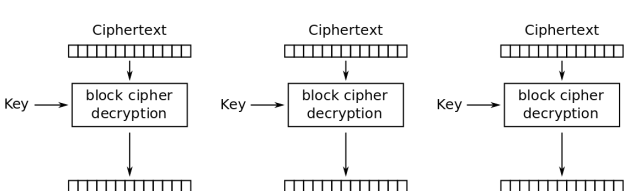
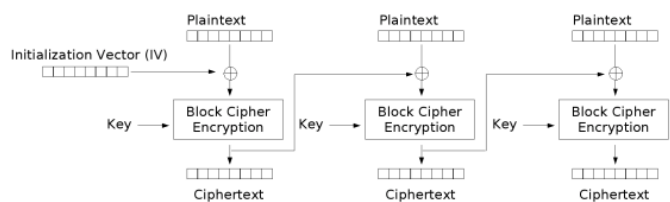
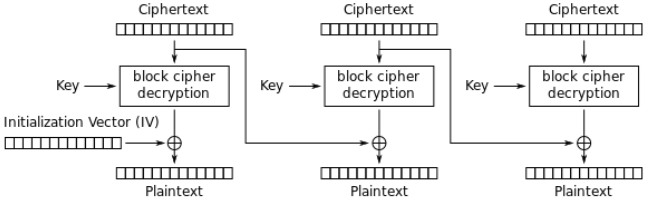
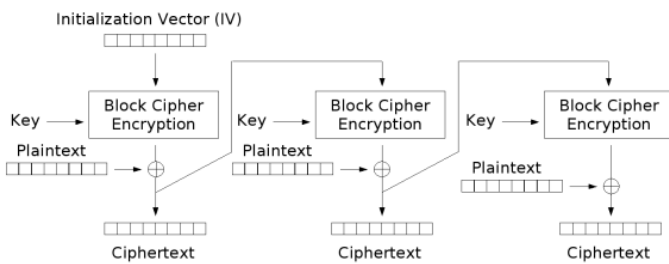
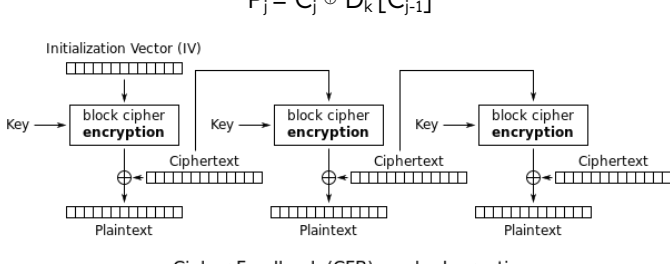
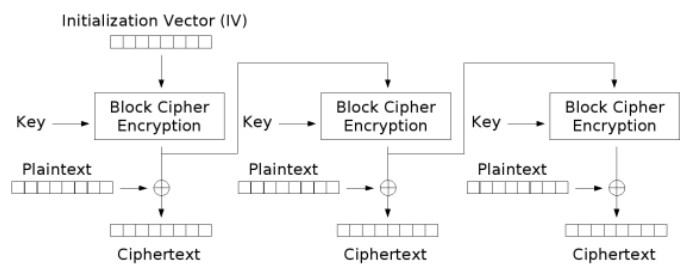
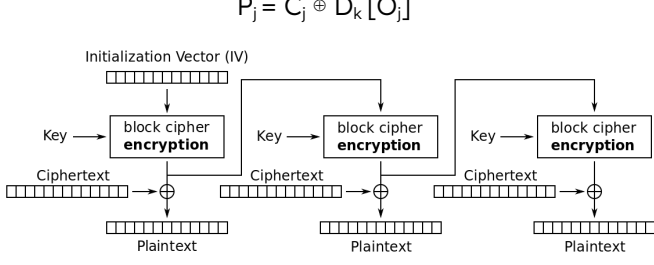
E poi niente, sposti i valori (freccette) e ricalcoli di nuovo Q,R,T  
fino a quando A=1

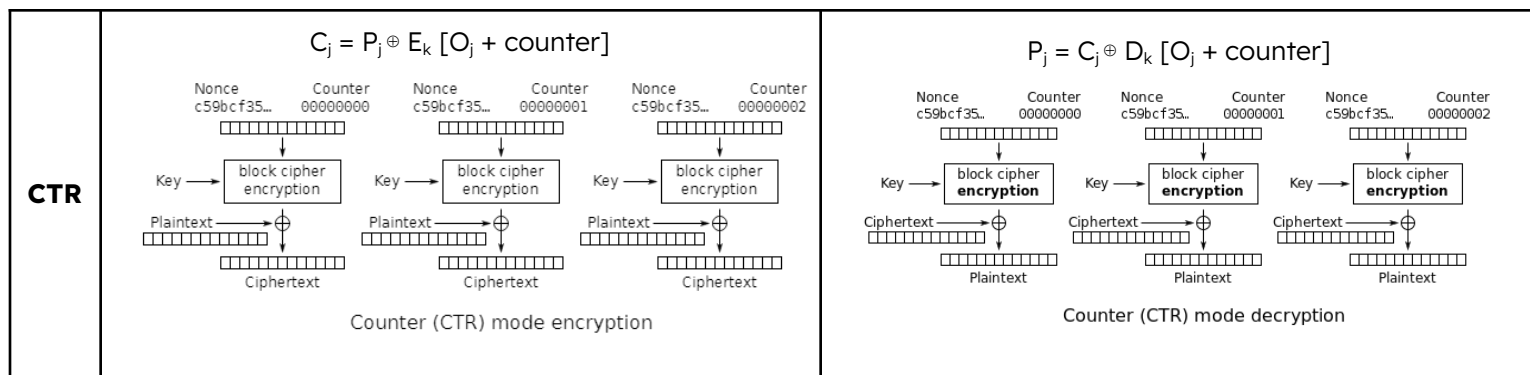
16:48

Quando (e se) A=1 allora T1 è l'inverso

16:49

Q	A	B	R	T <sub>1</sub>	T <sub>2</sub>	T
1 (A/B)	64	47	17	0	1	$0 \cdot 1 + 1 = 1$
2	47	17	13	1	-1	$1 + 1 \cdot (-1) = 0$
1	17	13	4	-1	3	$-1 + 3 \cdot (-1) = -4$
3	13	4	1	3	-4	$3 + 4 \cdot (-4) = -13$
4	4	1	0	-4	15	$-4 - 15 \cdot (-4) = 56$
	1	0		15		

	Encryption	Decryption
<b>ECB</b>	$C_i = E_k(P_i)$  <p>Electronic Codebook (ECB) mode encryption</p>	$P_i = D_k(C_i)$  <p>Electronic Codebook (ECB) mode decryption</p>
<b>CBC</b>	$C_j = E_k[C_{j-1} \oplus P_j]$  <p>Cipher Block Chaining (CBC) mode encryption</p>	$P_j = C_{j-1} \oplus D_k[C_j]$  <p>Cipher Block Chaining (CBC) mode decryption</p>
<b>CFB</b>	$C_j = P_j \oplus E_k[C_{j-1}]$  <p>Cipher Feedback (CFB) mode encryption</p>	$P_j = C_j \oplus D_k[C_{j-1}]$  <p>Cipher Feedback (CFB) mode decryption</p>
<b>OFB</b>	$C_j = P_j \oplus E_k[O_j]$  <p>Output Feedback (OFB) mode encryption</p>	$P_j = C_j \oplus D_k[O_j]$  <p>Output Feedback (OFB) mode decryption</p>



	ECB	CBC	CFB	OFB	CTR
Description	The simplest	Most used	Asynch stream cipher	Synch stream cipher	Counter
Pros	Ok for short message exchanges, like passwords	It works well, no major flaws	//	Preprocessing, hippie!	Super cool
Cons	Patterns are easily detectable	See below	//	//	No error propagation, and if counter is lost can be a problem
Pattern Hiding	NO	YES	YES	YES	YES
Parallelizability	Encryption: YES Decryption: YES	Encryption: NO Decryption: YES	Encryption: NO Decryption: YES	Encryption: NO Decryption: NO	Encryption: YES Decryption: YES
Preprocessing	NO	NO	NO	YES	YES
Errors propagation	NO	YES	YES	NO	NO

## Security:

- Confidentiality
  - Symmetric cryptography
    - Perfect cipher
      - OTP - One Time Pad
      - Shannon's theorem with proof
    - Attack models
      - Known plaintext
      - Chosen plaintext
      - Adaptive chosen plaintext
      - Chosen ciphertext
      - Lunchtime attack - chosen ciphertext for limited time
      - Adaptive chosen ciphertext
    - Stream ciphers
      - Use a seed (it's what is shared between the parties) to generate a pseudorandom keystream to xor with the plaintext
        - The keystream has a period, but should be the longest possible.
        - Previous or next values shouldn't be predictable based on current ones
        - Good randomization (distribution of 1s and 0s)
      - RC4 - shuffles an array of 256bits according to the seed, then use it to generate the keystream
      - Can be
        - synchronous - keystream depends only on seed
        - asynchronous - keystream depends also on the ciphertext
    - Block ciphers
      - DES
        - 56 bits key, broken
      - Double-DES
        - MITM attack
          - requires plain-cipher couple
          - Leverages a lookup table to compare decrypted ciphertext and encrypted plaintext
          - cost reduced to  $n * 2^n$  instead of the desired  $2^{2n}$
      - Triple-DES - can be used with just 2 keys!
        - Cost for an attack:  $n * 2^{2n}$  (basically double the security of DES = 112 bits)
      - AES
        - Performs multiplicative inverse over  $GF(2^8)$
        - Division in 4x4 grid of a block
        - KeyExpansion
        - sub bytes
        - Mix columns
        - Shift rows
        - Add round key
        - Repeat 10-12-14 times for 128-192-256 bit key
        - Last repetition without mix columns

- Modes of operation
  - Characteristics
    - Pattern hiding
    - Parallelizability
    - Preprocessing
    - Error propagation
  - Modes
    - ECB
    - CBC
    - CFB
    - OFB
    - CTR
- Ciphertext stealing
- Asymmetric cryptography
  - Why? Because it's too heavy to remember a key for every session
  - DH key exchange
    - Based on the hardness of factorization of a product of two large primes (discrete logarithms)
    - Public parameters
      - $p, g, p^a \bmod(g), p^b \bmod(g)$
    - private parameters
      - $a, b$
    - Perfect forward secrecy -  $a$  and  $b$  (private parameters) are discarded (not reused) after the key exchange, thus if an attacker manages to get the key no information about future or previous exchanges is also leaked
    - Attacks:
      - MITM - Trudy is able to start two communications with Alice and Bob and forwards (reading them) their messages
        - FIX: authentication of the parties
      - Logjam attack - Downgrade of the protocol to a version with little  $p$  and  $g$  values, easier to invert
  - RSA
    - Based on great value exponentiation invertibility hardness in modular arithmetic
    - How does it work:
      - Euler's theorem -  $a^{\phi(n)} = 1 \bmod(n)$ 
        - Euler totient function - the number of integers that are coprime with  $a$   $n$  chosen among the ones in  $[0, n-1]$
      - Two large primes are chosen:  $p$  and  $q$
      - $N = p \cdot q \Rightarrow$  being coprimes:  $\phi(N) = (p-1)(q-1)$
      - Choose  $e < \phi(N)$  such that is coprime with  $\phi(N)$
      - $(N, e)$  becomes the public key
      - Find  $d$  (private key) such that  $d \cdot e = 1 \bmod \phi(N)$  - multiplicative inverse
        - Bezout's identity can be used
      - To encrypt a message  $m$  we've got to:

- encode  $m$  into a manageable number
    - Perform  $c$  (ciphertext) =  $m^e \bmod (N)$
  - To decrypt a message  $m$  we've got to:
    - Perform  $m = c^d \bmod n$
    - Trick: to have fast encryption choose a small  $e$  (3,  $2^{16}+1$ )
- One way trapdoor function: hard to invert, becomes easy having some extra piece of information
- Attacks:
  - Factor  $N$  into  $p$  and  $q$  (badly chosen parameters)
  - Simple messages (they encrypt in a predictable way)
    - Fix: Salt
  - Chinese remainder - exploits same message being sent to multiple destinations
    - Fix: salt
  - Dictionary attack using public key - bruteforce by trying to encrypt many messages using public key
- Improved implementations:
  - PKCS (Public Key Cryptography Standard)
    - Applies some preprocessing to the string
    - It's obsolete
  - OAEP (Optimal Asymmetric Encryption Padding)
    - More complex preprocessing and encoding before encrypting
    - Salting and Padding to prevent simple messages attack
    - Concept of All or Nothing security - due to the preprocessing, taking partial knowledge of what's being sent isn't possible.
- ElGamal Encryption
  - Basically a rework of Diffie Hellman that prevents MITM
  - It works with public/private keys, used to possibly generate a session key
- Integrity - Unkeyed hashing functions
  - MAC (Message authentication codes) - similar to cryptographic checksum
    - It is appended to the message to prove it's integrity (also authenticity if it is used together with a key)
    - Can be performed with
      - CBC-MAC (see authenticity)
      - Hashing functions
    - Types of forgery
      - Existential
      - Selective
      - Universal
  - Hashing functions:
    - Collision Resistance
      - Weak
      - Strong



- Proof that strong  $\Rightarrow$  weak
  - SHA-1 (160 bits output)
    - Works with 512 bit blocks
    - Works with five starting standard values  $A \rightarrow E$  that get modified at each round and mixed with the padded message
  - Cryptographic hashing functions requirements
    - Preimage resistance - non invertibility
    - Second preimage resistance - strong collision resistance
    - Easiness of computing hash
  - Birthday paradox (at least 160 bits in output,  $2^{80}$  is infeasible)
  - Merkle Damgard Construction for block hashing
    - If the hashing function used is strong collision resistant, then also the one built with MD is strong collision resistant
    - If the hashing function works with 512 bit blocks and maps them into 160 bits, we split the message into 512-160 bits blocks and concatenate them
- Authenticity (of a message)
  - MAC
    - CBC-MAC
      - Safe only with messages with fixed length: variable length messages suffer from Length extension attack!
    - Keyed hashing functions
      - Schemes security:
        - $H(k||m)$  is vulnerable to Length extension attacks
        - $H(m||k)$  is vulnerable to Birthday paradox in case the key is exactly one block long
        - $H(k||m||k)$  is safe. Birthday paradox could theoretically still be used but it is not possible to actually check if I get a collision (because I don't have the key!)
    - HMAC
      - Construction that uses a key and a given hashing function to safely convert it into a keyed hashing function
      - $HMAC\_k(m,h) = h(k \oplus opad || h(k \oplus ipad || m))$
      - HMAC can be forged if and only if the underlying hash function is broken (collisions found)
    - Authenticated encryption
      - Provide authenticity and confidentiality
      - Using the same key shouldn't be a problem
      - Approaches
        - EtM (Encrypt then Mac) The only one proved secure
        - E&M (Encrypt and Mac)
        - MtE (Mac then Encrypt)
- Authenticity and non repudiation - Digital signatures
  - It's basically the public/private key equivalent to MAC
  - Follows always these steps:
    - Generation - of private and public key

- Signing - of the hash of the message by encrypting it with the private key (not the whole message, save space!)
    - Verification (Decrypting of the signature with the public key, compare it with the hash of the message)
  - RSA - with inverted keys usage wrt to confidentiality purposes
    - It uses SSA (Signature Scheme with Appendix), which means that the signed part is appended to the message
  - Elgamal signature scheme
    - somewhat similar to Diffie Helman logarithm concept
    - For each signature a random and temporary ephemeral private key is created (thus it is not reused for multiple signatures)
    - The ephemeral private key is used to derive an ephemeral public key, which is then used in the signature generation process. This approach provides enhanced security by preventing key reuse vulnerabilities.
  - DSS - Digital Signature Standard
    - DSA (Digital Signature Algorithm)
      - leverages:
        - ElGamal (it is a modification of it) and so discrete logarithm intractability
        - SHA
        - TimeStamping Authority (TSA): appends a timestamp to a document and digitally signs
      - Works this way:
        - p and q prime numbers such that p is multiple than q+1
        - $\alpha = 1^{(1/q)} \pmod{p}$
        - private key s chosen at random such that  $1 \leq s \leq q-1$
        - Divided in two steps:
          - Step 1: large number exponentiation
          - Step 2: SHA hashing and further exponentiation
        - Step 1 is preprocessable, step 2 is fast to compute
    - REMEMBER
      - random numbers are involved, thus the signature is not the same even if done two times on the same document.
      - Private and public keys are temporary and based on the random number k that is generated at each iteration of the process
        - This way perfect forward secrecy is achieved
        - If the adversary gets k, he can get the private key and thus break the algorithm
  - TSA (TimeStamping Authority)
    - A third party that timestamps a document and signs it, in order to add a temporal proof of the two parties' signatures.
- 
- Authentication (of an individual/machine)
    - Concept of closed world assumption
      - In absence of proof that someone is who he states to be, he just isn't
    - We need to take care of attacks:
      - MITM

- Spoofing
- With previously shared key
  - Nonce challenges exchange (basically to see if the other party has the key) with:
    - timestamps and short time interval for validity (avoid replay attacks)
  - Can be one way or two way
    - Warning: two way may suffer from reflection attack
      - Solution: different challenges for requester and receiver (avoid reflection attacks)
- Shared keys with third party (KDC - Key Distribution Center)
  - Needham Schroeder
    - It's the base on which kerberos is built
    - ticket concept implementation - something that the third party gives me but I cannot read, since it is encrypted with the recipient key. It contains:
      - my identity encrypted - to provide authentication
      - session key - accepted iff the ticket is recognized as valid from the recipient
      - timestamp, nonces and others
  - Expanded Needham Schroeder
  - Kerberos
    - Often used inside enterprises, but not only
    - It's basically an extended Needham Schroeder which implements
      - timestamps
      - lifetimes for tickets
      - AS - Authentication server: provides the TGT in exchange for the authentication of an user
      - TGT - Ticket granting ticket: it's basically a metaticket whose goal is to avoid the user inputting several times the password, it can be provided with a reusable ticket that can be presented to the TGS to get the actual ticket to send to other people.
      - TGS - Ticket granting server - takes in input a TGT and provides a ticket to talk to the actual recipient.
      - Authenticator: basically it's an encrypted couple identity-timestamp to guarantee authentication and avoid replay attacks. It is sent both to the recipient and to TGS.
    - Kerberos Realms: logical network under the same master KDC (AS/TGS)
      - v4
        - Used DES => deprecated!
        - if every authentication is done locally a request has to traverse a linear number of realms => quadratic effort!
      - v5:
        - Adds hierarchical structure of KDCs, reducing cost of traversing realms

- Delegation of rights: specific kinds of permissions are allowed (time/resource limits)
  - Renewable ticket: the same ticket can be renewed instead of recreated from scratch
- Private/public key
  - It's basically the same as the one in which the parties share a key with the trusted third party, but this time instead of encrypting tickets the server digitally signs the public keys of the parties to prove their authenticity.
  - Needham Schroeder with public keys and digital signature of the server
    - MITM from coworker Trudy (who is able to convince Alice to start a conversation with him)
- With digital certificates and Official Public key infrastructure
  - It's basically the same as before but this time the identity of a user is not asked every time to the trusted server but it's granted by a reusable digital certificate.
    - A digital certificate is just the public key of a user signed by a Certification Authority.
  - X500
    - It's basically a request for talking, and in the various steps are sent:
      - Nonces
      - timestamps
      - digital certificate
      - session key encrypted with the public key of the other party
    - One way authentication
      - Uses of nonces and timestamps to prevent replay attacks
        - A nonce is stored for its lifetime and the new nonce of every new request is compared to all the stored ones. If there's a match that's a replay attack
    - Two way authentication
      - Two different session keys to prevent reflection attacks: one per direction of messaging.
    - Three way authentication
      - Canadian attack based on replay (basically MITM)
      - Three rounds of messages to prevent replay attacks
  - (PKI) Public Key Infrastructure
    - Certification Authority
      - Can be local (valid only for private network) or official
      - Hierarchy (to allow certifications to work between networks)
      - CRL (Certificate Revocation List) - too much overhead to remember all the revoked certificates
        - A separate list for every Certification Authority
        - They are not updated every revocation, thus it's not secure
      - OCSP (Online Certificate Status Protocol) - more efficient: it's basically a tool to check if a certificate is still valid, eliminating the need to download and process large CRLs.
        - OCSP stapling: the certificate status is sent signed by the server directly during the TLS handshake to save

time and to avoid people getting information about certificates which they don't possess.

- PGP and web of trust
  - A public key of a user is seen as reliable if several other users have signed it.
- Password Based
  - KDF: Key Derivation Function
    - Algorithm that derives a key based on a password, often through salting and hashing
    - Used to stretch keys into longer keys or to obtain keys of a required format
  - The problems:
    - Dictionary attacks
    - Replay attacks
  - Solutions
    - Password Salting - every password is concatenated with the salt, so that a dictionary attack is drastically slowed down, and Rainbow Tables are prevented
    - Lamport's Hash
      - Instead of storing the password, the server stores the username, a number  $n$ , and the password hashed  $n$  times. When Alice wants to log in she types the password, her machine hashes it  $n-1$  times and the server hashes it one last time.
      - At each successful authentication  $n$  is decreased by one.
      - Small  $n$  attack: it's basically a man in the middle in which the attacker gets possession of Alice identity for some rounds, by forwarding to Alice a request with a smaller  $n$ .
        - FIX: Both parties remember  $n$  value so that it is never sent in clear. Very heavy
    - EKE (Encrypted Key Exchange)
      - like DH but with weak shared secret (basically not the password but a function  $W(pwd)$  of it)
      - It's resistant to MITM
      - Variations
        - SPEKE (Simple password "KE")
        - AEKE (Augmented Encrypted Key Exchange) - to solve the possibility where the password is stolen the parties don't remember directly the function but something derived from it
        - PDM (Password derived moduli)
- RNG
  - PRNG
    - Entropy
    - Same probability of ones and zeroes
    - No way to derive future and past values if current state gets leaked
  - CSPRNG

- No alg can guess a bit with precision better than 50%
- IPSEC (Network layer security)
  - Provides:
    - Authentication
    - Confidentiality
    - Key management (IKE - Internet Key Exchange - Symmetric key exchange based on variation of Diffie Hellman)
  - Cons: much overhead, difficult documentation
  - Use Cases:
    - VPN between networks (use in tunnel mode + ESP)
    - Host to Host private communication
  - SA (Security association) - unilateral agreement on these parameters
    - SPI (Security Parameter Index) - technical info on datagrams
    - IP address of the other party
    - SPI (Security Protocol Identifier) - decides on protocols:
      - AH (Authenticated header) - extended header to provide authentication
      - ESP (Encapsulating Security Payload) - extended header to provide encryption
    - All SAs are stored in SADB (Security Association Database)
  - Modes
    - Transport mode - host to host
      - The original header isn't touched, it's just added a new layer for IPSEC
    - Tunnel mode - VPN, network to network
      - The whole packet is encapsulated in a new one. Heavier but with extra security
- TLS (Transport Layer Security)
  - End to End encryption, unlike IPsec this times applications need to be reworked in order to work with TLS
  - Used in HTTPS and secure mail exchanging
  - Provides protection against:
    - Eavesdropping (confidentiality)
    - Tampering (integrity)
    - Message Forgery (authenticity)
  - TLS/SSL session is the result of an handshake that can comprehend several TLS/SSL connections
  - TLS handshake
    - List of crypto algorithms preferences
    - Mutual Sending of authentication (often X509 certificate)
      - Fixed - uses CA to sign DH parameters
      - Ephemeral - privately signed DH parameters
      - Anonymous (old one - vulnerable to MITM)
    - Change cipher protocol
  - Record protocol: responsible for the encapsulation, fragmentation, and encryption of data

- Attacks
  - Heart bleeding - It's basically a buffer overflow: we trick an host into sending us much more content than what he intended to
  - POODLE Attack - Downgrade to an old SSL version, confidentiality is broken
    - It's solved by forbidding the downgrading of the protocol
- SSH (Secure SHell) - based on TCP connections (port 22)
  - Can be used as:
    - SSHFS (remote storage sharing)
    - File storage
    - Remote Shell
    - Port forwarding (maybe my machine doesn't have an internet connection, thus it can ssh with a server that works as a proxy)
  - Provides
    - One way or two ways authentication
    - Data confidentiality (encryption)
    - Data integrity
  - Steps of connection
    - SSH Transport Layer Protocol (SSH-TRANS) - establishes initial connection
    - SSH Authentication Protocol (SSH-AUTH) - authentication
    - SSH Connection Protocol (SSH-CONN) - connection finalization and remote execution
    - SSH File Transfer Protocol (SSH-SFTP) - OPTIONAL to share files
- Firewalls - Datagrams analyzers
  - Can use two approaches
    - Blacklisting - default action: ACCEPT
    - Whitelisting - default action: DROP
  - DMZ (Demilitarized Zone)
    - Has the goal to expose a subset of the services of a private network to the external network. It's basically a place in which the rules of the firewall are much looser, because it's not a problem if its security gets compromised.
  - Can be defending:
    - An host
    - A network
  - Can be:
    - Session based - stateful, takes decisions on packet based also on what's previously tried to access
      - Useful to prevent:
        - syn flooding (basically a DOS, opening loads of TCP connections)
    - Packet based - stateless, just evaluates every packet on its own.
    - Application level (work directly with applications)
      - Bastion Host: often used in combination with a packet based Firewall. It's a hardened machine posed behind the packet firewall that controls all the flowing of the packets.
- IPtables
  - Implementation of a firewall

- Chains: lists of rules which can match a set of packets
  - INPUT
  - FORWARD
  - OUTPUT
- Targets: what to do with a packet that matches a rule
  - ACCEPT
  - DROP
  - RETURN
- Remember
  - -m multiport: multiple ports separated by commas
  - -m limit -limit 5/s: limita il numero di pacchetti a 5 al secondo
  - 0/0: all ip addresses are ok
  - -m state -state NEW/RELATED/ESTABLISHED
- Shamir Secret Sharing
  - Concept of Information-theoretically secure