**Q1**

.1: T

.2: F

.3: T

.4: F

.5: F

.6: T

.7: F

.8: F

.9: T

.10: F

**Q2**

**.1:**

AES (Advanced Encryption System) is a symmetric cipher based on 3 steps: Confusion (Substitution), Diffusion (Shift + Mix), Key whitening (Key addition layer).

It takes in input 128 bit blocks. If document is greater, use operation modes.

State is a 4x4 matrix, each cell – 1 byte.

Confusion – Substitution:

In this layer there is the only non-linear operation, we consider every byte as a 2^8 polynomial, take the inverse in GF(2^8) and then multiply it with an S-Box, a random matrix.

Diffusion – Shit Rows:

We take the state, transpose it and do this right shift: 1st row – no shift, 2nd row – 1 position shift, 3rd row – 2 position shift, 4th row – 3 position shift.

Diffusion – Mix Column:

Now consider every column as an array and multiply each for an irreducible GF(2^8) polynomial.

Key Whitening – Key addition Layer:

A the beginning we choose for Key length: 128 bits (10 rounds) – 192 bits (11 rounds) – 256 (12 rounds).

At the end of each round we xor the entire block with the key – output.

At each round the key change (key schedule), it is divided in 4 words of 32 bits. The first word is processed with a g function varying with round counters, and then that word is xored with the others.

**.2:**

RSA is so slow due to the huge numbers you have to choose. In order to make RSA secure you should use 1024 bits numbers, and exponentiation with this huge numbers is so slower than xoring or multiplying 128 bits as in AES. [..]

In fact, Usually we use hybrid schemes, with RSA as Key Exchange Protocol to exchange AES-keys.

**.3:**

1. With the IV we ensure the univocity of the result.
2. CBC has a particular behaviour regarding propagation of error. If you manipulate the IV the first block will change, but the other blocks won’t have changes since in the decryption every block yi, after the cipher is xored with yi-1, the first, instead, after the cipher is xored with the IV.

**Q3:**

**.1:**

There are three types of RNG:

1. TRNG: Based on physical events (e.g. coin flips) expensive, not so practical applicable.
2. PRNG: Are RNG cheaper but more secure than TRNG, anyway they starts from specific seeds and then use some math. Function to compute future random sequence.
3. CSPRNG: PRNG more “random”, they try to use unpredictable seeds and ciphers. Unix has Kernel Entropy Pool, a pool of random numbers got from physical events, such as thermal noise. Then seeds are compute using the same mechanism of ciphers, such as: RSA,DES and soon.

Two examples are NETSCAPE (obsolete since uses MD5 and pid) or MERSENNE TWISTER.

A good RNG must have some properties: Sequence numbers have to be unpredictable, starting from a number of a sequence it must be impossible to compute next numbers, and starting from a number it has to be impossible to understand how the sequence has been built.

**.2:**

1. 1) Keeping fixed X, there is the same problem of OTP, If we XOR two results we’ll get X.

2) Keeping both fixed

1. .

**Q4:**

**.1:**

X – message

S - signature

First of all I have to verify that the authentication tag (x,s) is valid. Assuming the correctness of the pair, I can send him a Nonce N and my Identity (e.g. “A”) encrypted with his public key, if and only if he is Bob will decrypt it. Assuming he is doing a MITM attack, he has already started an authentication with Bob with his identity (e.g. “T”) and now he is sending me a fake identity B, but when he will send my message to Bob in order to decrypt it and catch info Bob get my identity (A) and will get suspicious.

**.2:**

A ticket is a message composed by some parameters for the receiver in order to be secure of the communication.

Kerberos is the current standard in DS for authentication with Servers. It provides one-way authentication. When a user A wants to use a service provided by a server B, he will contact C (trusted party that has a key for every user) sending A,B,N,t (timestamp), so C will send to A: Ticket for B, Kac(Kab, N, L, B) where Ticket of B = {Kbc(Kab, L, ta, A)} where Kbc is the key used between B and C and Kab is the session key between A and B.Now A will send to B his ticket and Kab(A,ta). B will decrypt both and check validity of identity.

**.3:**

Unless we can construct public-key mechanisms or have a certificated authority, We have to use simple challenge-response mechanisms.

Since we have a secret shared key and can use a cryptographical hash function (assume SHA-3), and assuming both parties can apply AES, we already have two strong functions.

In order to authenticate herself, Alice can ask for Bob to authenticate, so Bob will send her AESkab{N,B}. Alice knows the key so will decrypt it and sends back to B AESkab{SHA-3(N),A}.

Bob can decrypt it and since he knows N can hash it and check if it is equal to the one sent by A.

The nonce ensures to avoid replay attack. The key with AES the confidentiality since AES is not broke. In a man in the middle attack, the attacker should be able to decrypt the message in order to understand something. Impossible.

Or, since the clocks are synchronized, Alice can send a message in which she put her identity and SHA-3(Kab{Timestamp}). Bob will do the same and can understand that it is Alice.

**Q5:**

**.1:**

Each table has particular tasks and contains several chains. Tables are: Filter, Nat, Mangle, Raw. Filter table has to filter packets, Nat table has to translate addresses, Mangle table manipulates tcp headers and Raw table is for general purpose. Chains represent the “type” of that packet: INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING, and what to do with each of this.

**.2:**

The two rules differ only for the Chain, both have as output interface the eth2, as destination port the 443, use protocol TCP, and accept the packet. The main difference is that in the first rule we are talking about how behave with packet coming from the outside and not for us. The second is referred to packets going outside the network.

**.3:**

This rule ACCEPT INPUT packets (so packets from the outside) of connections already ESTABLISHED, using tcp protocol, sent by interface eth1, with whatever port as destination, and 22 as input port.