# Network and Telecommunications Systems Security

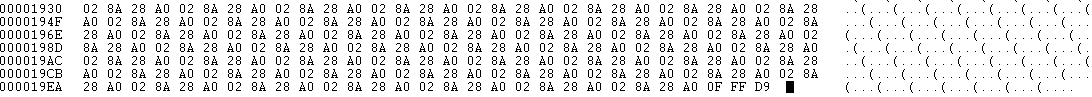
# Homework\_2

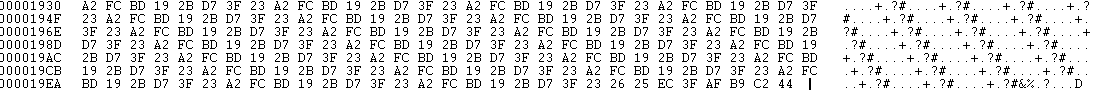
AM: EN2190001

Name :Evangelos Siatiras

# Exercise\_1\_A

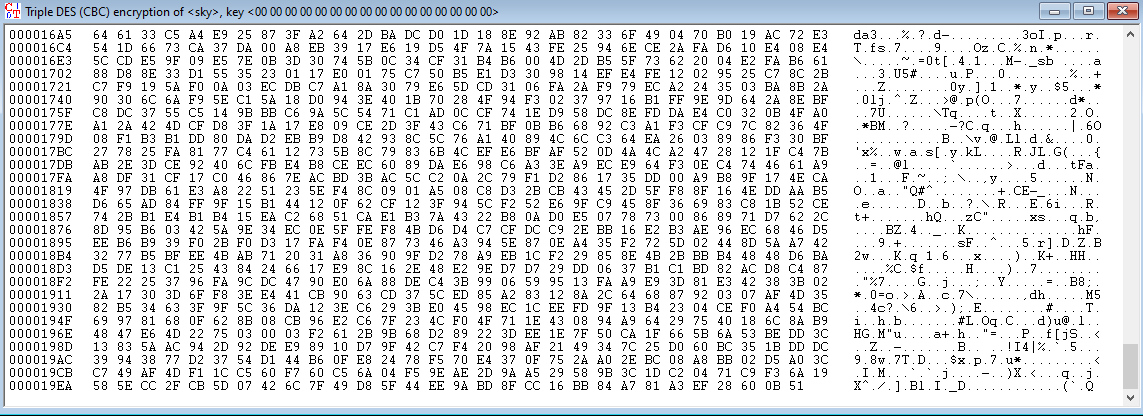
In Electronic Code Book (ECB) mode of operation does not require IV for encryption as the input plain text will be divided into blocks and each block will be encrypted with the key provided and hence identical plain text blocks are encrypted into identical cipher text blocks. Obviously, by scrolling to end in the original and encrypted image in hex we realize that the structure in the ciphertext is identical and only the ascii values have changed.

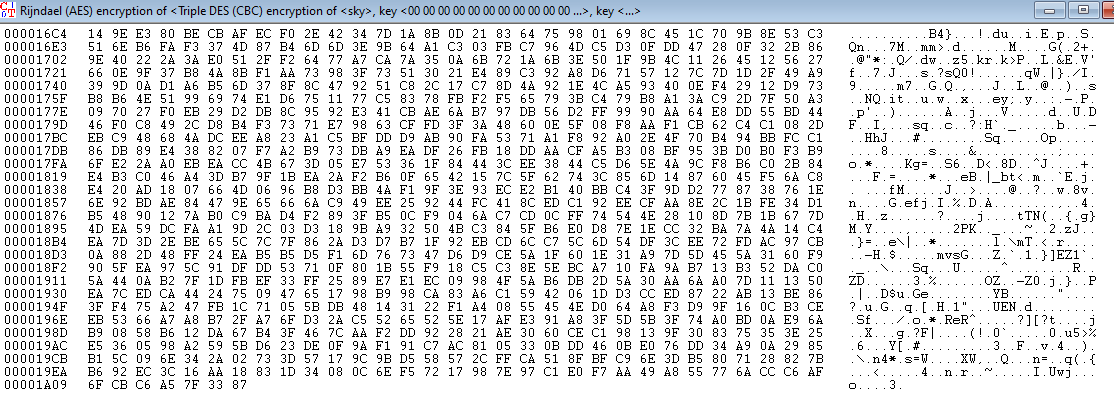




# Exercise\_1\_B

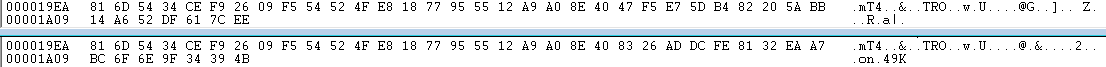
In the cipher block chaining (CBC) mode of operation, an initialization vector (IV) is exclusive-ored with the plaintext prior to encryption. For the first round of encryption, this is a random, public value. For subsequent rounds, it is the ciphertext of the previous round. This is intended to fix the issue with ECB mode where identical plaintext blocks create identical ciphertext blocks. In case that there is no IV we have always the same output for a message so an attacker can guess changed blocks. Among the differences between AES and 3DES like the encryption key length and block length by using the same mode of operation CBC in our case ensures that in every round of the encryption the ciphertext is “combined” with the ciphertext in the previous round (at first with IV) so the structure of the ciphertext in both cases will be different. Of course, there are differences in the structure of the ciphertext resulted from AES and 3DES because the encryption is totally different as mentioned above.





# Exercise\_1\_2

By changing the last digit from D9 to D8 we get the following result.



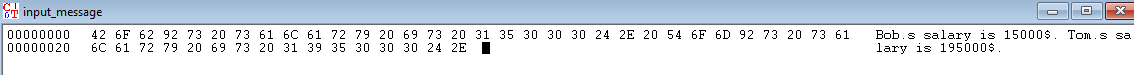
Obviously the values of the last block (last 128 bits so 16 bytes) in the above image is totally different.During the encryption process the plaintext is passed through 4 steps (SubBytes, ShiftRows, MixColumns, AddRoundKey) for many rounds.If we were able to find something in common AES would be a weak encryption algorithm based on alphabetic substitution.

# Exersice\_2\_A

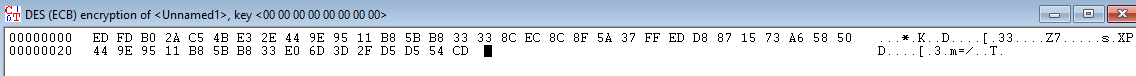
At first intuitively as the message consisting of 48 characters ( 48 bytes) and DES has message blocks of 64 bit (8 bytes) so the resulting ciphertext will consist of 6 blocks of 8 characters each.As the mode of operation is ECB means that the resulting ciphertext with the same key will always be the same. Thus we could start to guess the mapping of the plain text to cipher text was.So if we replace the blocks of 8 bytes containing the name Bob with the one containing the name Tom and vice versa we get the expected result.

Proof:

We start with the initial Message



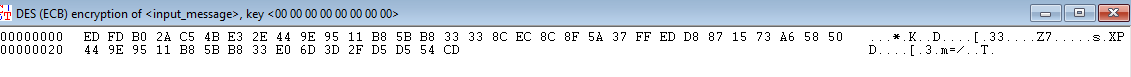
And we encrypt it using DES(ECB) with the key as per below in the image and we get :



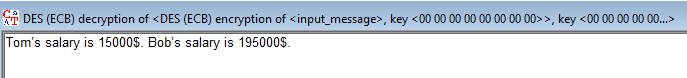
Then the two concerning blocks are the ones below:

And by doing the necessary text replacements we get the following ciphertext:



And the decrypted message is :

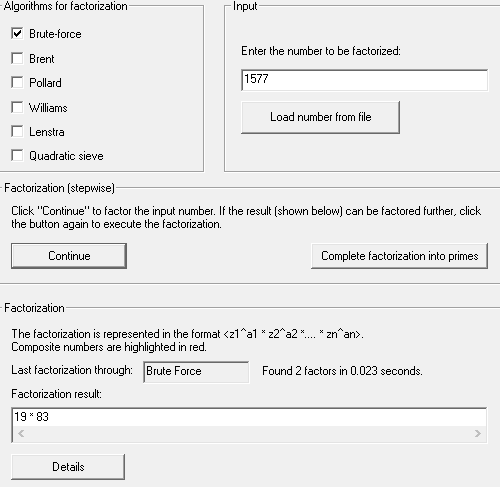


# Exersice\_2\_B

If the original plaintext is smaller than the AES block size (128 bits regardless of the key size) so it fits into a single block then editing it in a meaningful way without knowing the key is not feasible yet and if it is done the AES encryption will be broken. Once the plaintext is longer than one block, using ECB makes it trivially easy to mix and match between pieces of separately encoded messages at block boundaries like above with DES. So, we can conclude that this is a property of ECB, not of AES or DES in particular.

# Exercise\_3\_A

Let’s start with the factorization of N.Thus (module in RSA method) N is small the factorization is easy and by using just brute force algorithm we end up with 1st prime p=19 and second prime q=83.

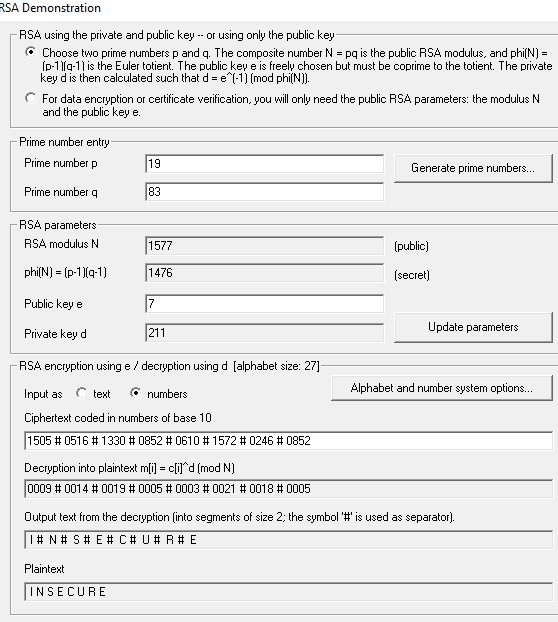


We see that the two primes are different so we can continue(if the primes are equal it will not work).Then having the knowledge about p and q and because they are different we are able to calculate φ(n) = (p-1) x (q-1) known as Euler φ function.

φ(*n*) = (*p* − 1) × (*q* − 1) 🡪1476 and by applying the above to the following formula e × d = 1 (mod φ(*n*)) and Since by definition e and ϕ(N) are coprime then with extended euclidean algorithm you can find such d: ed+kϕ(N)=1. For the chosen values of p, q, and e, we get d as:

d = 211

By having the private key d and by specifying the alphabet in the cryptool we are able to decrypt the message as per the image below.



# Exercise\_3\_B

At first we have N = 1577 and e = 7 and the maximum number in the ciphertext representing the plaintext is between 0 and N-1 we can use Plain RSA and we can get the plaintext M from the ciphertext C through the following formula : C = M^e mod N and by solving with respect to M we get the plaintext for every encrypted number in the ciphertext as per the below links :

[M[0]](https://www.wolframalpha.com/input/?i=1505%3Dm%5E7+mod+1577) , [M[1]](https://www.wolframalpha.com/input/?i=516%3Dm%5E7+mod+1577), [M[2]](https://www.wolframalpha.com/input/?i=1330%3Dm%5E7+mod+1577), [M[3]](https://www.wolframalpha.com/input/?i=0852%3Dm%5E7+mod+1577), [M[4]](https://www.wolframalpha.com/input/?i=0610%3Dm%5E7+mod+1577), [M[5]](https://www.wolframalpha.com/input/?i=1572%3Dm%5E7+mod+1577), [M[6]](https://www.wolframalpha.com/input/?i=0246%3Dm%5E7+mod+1577), [M[7]](https://www.wolframalpha.com/input/?i=0852%3Dm%5E7+mod+1577)

# Exercise\_4

Message Authentication Code (MAC) is cryptographic code, calculated by given key and given message:

auth\_code = MAC(key, msg)

Typically, it behaves like a hash function: a minor change in the message or in the key results to totally different MAC value. It should be practically infeasible to change the key or the message and get the same MAC value. MAC codes, like hashes, are irreversible: it is impossible to recover the original message or the key from the MAC code. MAC algorithms are also known as "keyed hash functions", because they behave like a hash function with a key.

With the specific MAC algorithm if we were able to create a message with the same H(M) (To achieve sha-1 signatures to collide) it would be possible to extract the aes key and then via decryption, the mac key.In such scenario an attacker will have all the information in order to send a message that will be accepted in the destination.

However, the only concrete SHA1 collision to date was Google's.If cryptoanalysis advances, an attacker might be able to create inputs that deliberately collide. Currently, however, there is no known way to do this efficiently. For example Google needed 110 GPUs running for one year with an algorithm they implemented called “SHA-1 Shattered” (with brute force in order to achieve it in one year would need 12.000.000 CPUs).So in a feasible amount of time and expenses it is not possible neither to find the secret key nor to compute another valid pair.