Ort Braude College

Cryptology 61767

Final Project 2020-2021

We're group number 3.

Our crew members are:

Elias Renawi

Gazal Khoury

Mahmoud Ataria

Razy Halaby

Project main subject:

Secure email exchange

Project sub-subjects:

Encryption and decryption using:

1) Serpent

2) OFB

Secure channel for key transmission using Deffi-Hellman algorithm of key exchange

And finally, El Gammal signature

We began our work with extended research regarding all the project subjects and their relation of each other, we tried to enter to the utmost depth of each one (regarding the time restriction), and to do all the masterpieces we studied justice.

After finishing the studies, we combined the work on code and presentation, as the presentation making gave us a better understanding and the code made us aware of the little details.

The work flow:

We wrote the code of our project using python, the coding part was like a chain, we started encrypting with Serpent and decrypted with OFB using Deffi-Hellman algorithm of key exchange in block ciphering and supported by El-Gammal Signiture to guarantee the prevention of 3rd party interventions that Deffi-Hellman doesn’t support, then validated using El-Gammal Verification algorithm. In the first part:

we used the serpent algorithm with 256-bit key which we generated using “cryptodome” python library.

The keys= ElGamal.generate(256, Random.new().read)

1. Function generate Modulus (*p*).
2. Generator (*g*).
3. Public key (*y*).
4. Private key (*x*). Optional

which we use to calculate the common key to between the two parties

The common key is the one that used for the encryption of the message in the serpent.

We used the “OFB” to encrypt the plain text. in the OFB we generated an IV using a random string of 16-character (128 bit) and save it so it will be used for the encryption and the decryption, we fixed the length of the plaintext if needed so it can be divided to an equally 128 bit each slice, we did this by adding one (“1”) and as many zeros (“0”)s as it needed to be divided to 128-bit each slice.

For the decryption version we must have the Ciphertext, the key and the same IV (Initial vector) that was used for the encryption.

Then we append all results from each block of the OFB model to return it the original plaintext after we get the original plain text, we got it in a byte stream to converted it back to the original ascii string.

The serpent:

* The input and the output of the serpent are inserted in a binary mood
* The OFB computes the binary form of each message block and sends it to the Serpent
* The Serpent calculates the 32 subkey, the initial and final permutation and runs for 32 rounds then it returns the binary form of the ciphertext

“El Gamal” signature part:

* In the signature part we computed the parameters and generated the keys (based on Diffie-Hellman scheme) we followed the steps of El Gamal signing to compute the DS= (s1, s2) (the signature)
* We added other party public key (Yb) to simplify the imputation of the algorithm
* In the verification we followed the steps of El Gamal for the verification to compute the results

Obtained results and conclusion:

1. In conclusion we sent:
2. The encrypted message using the common key
3. The Initializing vector
4. The signature using a private key
5. The other party is supposed to receive:
6. The IV
7. encrypted message
8. The other party duty:
9. To decrypt the encrypted message using the common key and the iv
10. To verify the signature using the public key of the sender

Some of the many references we used:

The great lectures and explanations of our course professors

A Digital Signature Scheme using Diffie-Hellman Key Exchange, Muhammad Fareed, Uddin and Kashif Siddiqui

Differential-Linear Cryptanalysis of Serpent, Eli Biham, Orr Dunkelman, and Nathan Keller

Quality of Encryption Measurement of Bitmap Images with RC6, MRC6, and Rijndael Block Cipher Algorithms Nawal El-Fishawy and Osama M. Abu Zaid

Wikipedia for all of the subjects

https://olvid.io/faq/secure-channel-creation-in-cryptography/