The project study process:

Asaf Ben-Shabat

Or Eliyahu

Sami Odeh

Tal Vakrat

The requirements for our project are:

* Implementation of the Enigma protocol as a part of the network activity.
* Key generation and key distribution.
* Digital signature using El-Gamal.
* Improve the Enigma.

Study process:

Every 2 days we met on Zoom meetings defining work distribution and sharing conclusions and knowledge about the project.

The initial study of the project was alone for each group member.

After few days after we received the subject we studied the subject for 3 days, later on we met on Zoom and shared conclusions and explained to each other the points we did not understand alone (for example, Asaf did not understand the different notch point for each rotor).

After we understood the idea of enigma encryption we had to decide the programming language for coding the project, the general design and the way to present it.

We decided of python since it’s easy to handle, studying and maintain internet connection between 2 Pc’s.

Information sources:

* Lectures and presentations.
* Mathematical and Computer modelling online book, <https://www.sciencedirect.com/science/article/pii/S0895717711001129>.
* Cryptography, theory and practice online book, <http://index-of.es/Cryptography/Cryptography%20Theory%20And%20Practice%20-%20Douglas%20Stinson.pdf>.
* General other resources.

Obstacles we met upon code-implementation:

* CBC mode of operation:

CBC usually receives the bits for the XOR operation **before** encryption.

But on enigma we could not make it, because enigma receives only A-Z characters and by the premier XOR operation we could get 8-bit set with no ASCII value representing an English character.

Solution: we did the XOR operation on the encrypted data bits and not the plain text.

* Padding for CBC:

CBC uses a fixed size block of data, in out specific implementation we used 3 characters block size. Hence every block of data where we added padding of “F” in the end of the message.

* Modulation of physical-electrical machine to a virtual machine (implementation details).

Implementation details:

Enigma machine had few version, Our project implementing the WW2 military version:

5 rotors to choose 3 of them for actual usage, 3 reflectors to choose 1 of them for actual usage and Plug-Board.

Enigma key distribution is one-type-pads type.

The “pad” in our project is a configuration file of .JSON type.

The Json defines:

* Key exchange (Diffie-Hellman)
* The encryption algorithm (enigma).
* The Mode of operation (CBC).
* Key for enigma (initial position for the rotors Derived from the key).
* Plug board pairs.
* Reflector (which one of the 3 implemented reflectors will be used).
* Digital signature algorithm (El-Gamal).
* Hash function for the Digital signature (sha1).
* Key for Mode of operation (IV).

The project consists many interfaces in order to enable us program parallel and in order to make the project more general and flexible in the meaning of changing the algorithms\keys etc’.

Results:

Python project as described in the Class diagram.

Given IP and Port address, the script creates a secured socket between 2 computers.

Our program defines a chat between the users, but the socket generally could be used to secure any platform of message transformation.

Initial key Generation(server side user), as in real enigma, is Defined by a json file (equal to the configuration paper).

The first key exchange is made by Diffie-Hellman.

Upon connection between the users, each one of them sends the other one his public keys.

As for the enigma machine, each user initializes two virtual enigma machines, one to encrypt and send messages, and the other one is to receive messages and decrypt them.

Conclusions:

Enigma is a symmetric secret-key algorithm.

There are few version to it but usually it has possible keys.

One might think that algorithm with no unhidden messages is good, but the fact that there is no letter encrypted to itself was actually a weakness and a lead to hack the Enigma in WW2.

The bombe machine used to crack the enigma, without the fact that there are no hidden messages it would be impossible (brute force with 1 million check per second would take 5 years to crack, and with the knowledge that there are no hidden messages it would take 1 hour).

The biggest advantage of our project is the fact that we managed to overcome the flaw of enigma (no hidden messages) with the CBC mode of operation, now every letter may be encrypted to any value from 0x00 to 0xFF including itself.

Also, the way we implemented the cryptosystem is a hybrid cryptosystem (key exchange by asymmetric algorithm, and data encryption is done by symmetric algorithm).

Together with the digital signature, our cryptosystem supplies a secured and fast communication channel.

How to run the Project:

Navigate to the source folder then open the Cmd or the Powershell twice

On the first cmd:

py App.py -server -ip <IP\_address> -port <port>

on the second cmd:

py App.py -client -ip <IP\_address> -port <port>

Start sending messages to each side.