# Assignment 1
# COMP543 Fall 2020 - Modern Cryptography

**Erhan Tezcan 0070881**

**30.11.2020**

## 1 Preliminaries

Note that all parts here are using the given rotor settings and a base alphabet as capital English letters.

- **Base**: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
- **Rotor 1**: `SHBMFWEIQRODTAVXCPYZUJKGNL`
- **Rotor 2**: `GYRFNUCZLQDWMKHSJOEPBVITXA`
- **Rotor 3**: `MSEWGQHDPRFNXATOIBUJLCZVYK`

Lowercase letters are also automatically uppercased. However, **do not enter any character other than the English alphabet!** It will cause an exception.

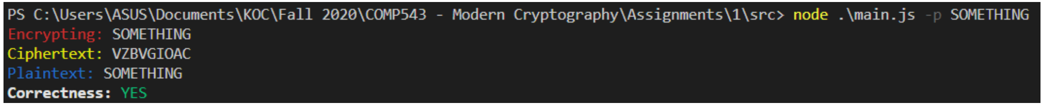
You will need to install NodeJS[1]. I am including the packages in my submission, however if things do not work please type in `npm install` at the active directory (`src`). This will install few packages that are required.

## 2 Encrypting and Decrypting

- To decrypt a message, do `node ./main.js -c <ciphertext>`
- To encrypt a message and check it's correctness, do: `node ./main.js -p <plaintext>`

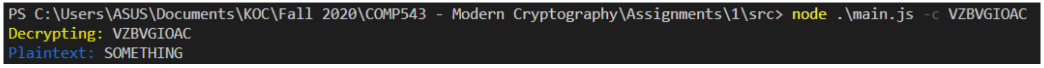
[1] https://nodejs.org/en/

Figure 1: Example encryption.



```
PS C:\Users\ASUS\Documents\KOC\Fall 2020\COMP543 - Modern Cryptography\Assignments\1\src> node .\main.js -p SOMETHING
Encrypting: SOMETHING
Ciphertext: VZBVGIOAC
Plaintext: SOMETHING
Correctness: YES
```

Figure 2: Example decryption.



```
PS C:\Users\ASUS\Documents\KOC\Fall 2020\COMP543 - Modern Cryptography\Assignments\1\src> node .\main.js -c VZBVGIOAC
Decrypting: VZBVGIOAC
Plaintext: SOMETHING
```

## 3 Bombe

- To use a bombe, do: `node ./main.js -bombe`

The program will ask you to enter a ciphertext (which you can copy paste to CLI) and then ask for an arbitrary number of keywords. Upon entering an empty keyword it will start the attack. You can see the progress of the attack as it is conducted. Note that since this progress bar is on the same line all the time, if your window is smaller than the bar it might cause a visual bug. At the end, you will see the candidate plaintexts that contain the given keywords.

The Bombe tries $26^3 \times 3!$ combinations, where the $26^3$ comes from rotation settings of 3 rotors, and 3! comes from the positions of rotors themselves.

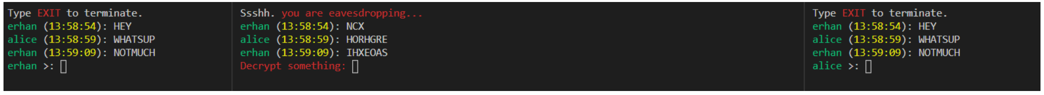
## 4 Chatting and Eavesdropping

- To enter chat room with a distinct process: `node ./main.js -chat <username>`
- To enter chat room as an eavesdropper: `node ./main.js -chat-eav`

The chatting will happen through `./chat/history.json` file. The stored file has encrypted messages. The honest users (thanks to their enigma machine) can successfully encrypt a new message there or decrypt the messages and see the content. However, an eavesdropper with just an access to the chat medium, only sees the ciphertext.

As the eavesdropper you can decrypt a message whenever you want. The logic is exactly same as described in bombe section. When the decryption is finished, the program will terminate, however you can still connect as

Figure 3: Example chat with 2 honest processes and 1 eavesdropper process.

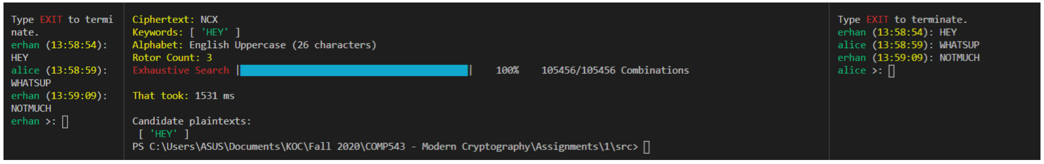


```
Type EXIT to terminate.
erhan (13:58:54): HEY
alice (13:58:59): WHATSUP
erhan (13:59:09): NOTMUCH
erhan >: ▯
```

```
Ssshh. you are eavesdropping...
erhan (13:58:54): NCX
alice (13:58:59): HORHGRE
erhan (13:59:09): IHXEOAS
Decrypt something: ▯
```

```
Type EXIT to terminate.
erhan (13:58:54): HEY
alice (13:58:59): WHATSUP
erhan (13:59:09): NOTMUCH
alice >: ▯
```

Figure 4: Example eavesdropper that uses a Bombe to decrypt a message.



```
Type EXIT to termi
nate.
erhan (13:58:54):
HEY
alice (13:58:59):
WHATSUP
erhan (13:59:09):
NOTMUCH
erhan >: ▯
```

```
Ciphertext: NCX
Keywords: [ 'HEY' ]
Alphabet: English Uppercase (26 characters)
Rotor Count: 3
Exhaustive Search |████████████████████████████|    100%    105456/105456 Combinations

That took: 1531 ms

Candidate plaintexts:
 [ 'HEY' ]
PS C:\Users\ASUS\Documents\KOC\Fall 2020\COMP543 - Modern Cryptography\Assignments\1\src> ▯
```

```
Type EXIT to terminate.
erhan (13:58:54): HEY
alice (13:58:59): WHATSUP
erhan (13:59:09): NOTMUCH
alice >: ▯
```

an eavesdropper and decrypt another message. When a new user joins (with `-chat <username>` option) the chat is reset. When an eavesdropper joins (with `-chat-eav` option) however, the chat is **not** reset.

Note that because the chat is printed to the screen with an interval where the console is cleared and chat is printed again, you might see as if your input is deleted. However, that is actually not what happens, and you can confidently type in your inputs.