COMP2K

Codebreaking

Laboratory

Instructions

Codebreaking

Recommended you complete this part by the end of Week 3. You should demo this lab in your Week 4 practical session.

Cyphers and Codebreaking

[This is the lab sheet for the Codebreaking Demonstration. You must demonstrate it to the instructor in one of your practical sessions BEFORE the due date in order to be awarded marks. Please check the ECP for the correct due date. Note that sections are 'complete' and marks are awarded by attempting each task AND correctly answering related questions to the satisfaction of the instructor.]

The deciphering of secret messages in World War II was one of the main drivers of the computational age we currently live in. Cipher machines developed around this time were sophisticated enough that breaking the coding system by hand was no longer feasible.

This lab was designed to allow you to follow in Alan Turing's footsteps as he tackled such problems, helping building one of the world's first computational machines, the Bombe machine used to crack the code generated by the German Enigma Machine and later the more general-purpose electronic computer called the Colossus.

In the first part of the lab, we study the much simpler Caesar cipher used during Roman times that can be broken by hand, though we shall use computers instead. In the second part, we will use a pre-built simulator of an Enigma code machine in Python to crack Enigma coded messages just as the Allies would have done in World War II. Lastly, we will cover general code breaking tasks for those looking for a challenge.

Use of Generative AI

This task has been designed to be challenging, authentic and complex. Whilst students may use AI technologies, successful completion of assessment in this course will require students to demonstrate and engage in specific contexts and questions during their demonstration where the use of generative Artificial Intelligence (AI) tools will not be permitted during the demonstration. **A failure to reference generative AI along with their code submission use may constitute student misconduct under the Student Code of Conduct. Any attempted use of Generative AI during the in-person demonstration may constitute student misconduct under the Student Code of Conduct**. **Submission of code and references must be prior to your demonstration via the submission link on Blackboard.** To pass this assessment, students will be required to demonstrate their comprehension and coded solutions independent of AI tools.

Section I – Caesar Cipher (3 Marks)

The Caesar cipher is a simple plain text substitution cipher, where you replace your message alphabets to the same alphabet but shifted by a constant offset.

[See the starting code provided in `test_caesar.py` and `test_caesar_break.py`]

a) Use Python dictionaries to create a shifted directory of letters that will allow you to map your letters to shifted ones, so that you can use the encrypt and decrypt code provided in

`test_caesar.py` to encode and decode Caesar cipher messages. Demonstrate that your code works for your own custom message by running the encrypt and decrypt methods within the script provided.

b) This type of cipher is easily broken by observing that some letters in the English language occur more often than others, such as the letter 'e'. Use the initial code provided in `test_caesar_break.py` to write an algorithm to break the code and decrypt the message provided.

c) Extend your script to break any Caesar cipher message of an arbitrary shift if you haven't already done so. Test out your code with your neighbour or mate by exchanging ciphered messages of unknown shifts between yourselves.

Section II – Enigma Cipher (5 Marks)

The Enigma machine was a commercial electro-mechanical cipher device introduced to the public before World War II for encrypting communication. It was adopted by the German military with various enhancements to make it more secure before the war broke out. It can be seen as the ultimate plain text cipher system that not only incorporates shifts but also permutations that change for every letter encrypted by using a combination of rotors. You can find a detailed description of the machine and its internal workings in (Copeland, 2004) [Enigma, Section 2]. We may also find the Enigma videos in the Computation YouTube Playlist helpful.

For our purposes however, all we need to know is that the machine maps a letter to another letter in our alphabet using two main mechanisms. Firstly, there are three rotors that can be chosen from a set of 5 or 7 distinct rotors. Each rotor maps an input letter to another output letter with a shift determined by its rotation rate. All available rotors have different rotation rates and any combination of three can be chosen to be used within the machine for ciphering. The result is that the input letter passes through all three rotors to thoroughly jumble up the shifts to produce the output letter. The starting positions of the three rotors dictates the rotational offsets of the cipher and is called the window positions or key. The window positions change as the cipher is used, so that the initial window positions are required to decrypt messages. Lastly, there is an override mapping mechanism especially outfitted for the Germany military called the plugboard. The board allows one to add additional mappings of individual letters to other letters even before the rotors are applied.

[See the starting code provided in `test_enigma_simple.py` and `test_enigma_break.py`]

Shakes the Horrible has decided to purchase a set of Enigma machines to ensure communication with his armed forces (consisting mainly of Spider Monkeys) is secure as possible.

a) Demonstrate a working Enigma machine with the given simple example script.

In his arrogance however, he has decided to force his military to *always* use the window positions SSC as those are the initials of his name. He is also a cheapskate! He hasn't even bothered to purchase the plug board or any additional rotors, meaning he only has access to rotors I, II and III.

b) Decipher Shakes' message to his military given in the example script using the Enigma machine simulator.

After realising that his military forces are taking heavy losses, he decides to take the advice of his generals and allows a different fixed, but hidden window positions for all further communications. He

however still refuses to buy the plugboard. He does decree though that all messages are to end with the phrase "Hail Shakes!".

c) Using the known end phrase, phrases like which are called cribs, write an algorithm to break Shakes' new code into the provided script and decrypt the message provided.
d) Add a counter to your script to keep track of the number of tries. How many attempts does it take to crack the code? How long did it take on your computer? How long do you think it would've taken for a computer in the 1940s?!
e) If Shakes the Horrible wasn't so ignorant and worried about money, he would have purchased both the extra 2 rotors and the plugboard. How much longer would have the cracking his code taken on your computer? An estimate as a number of tries or minutes/hours is sufficient.

Section III – Code Breaking (2 Marks)

A number of sophisticated encryption schemes can be broken with some simple prior knowledge of the context in which the message was composed, but without knowing anything about the encryption scheme or device.

Crack the following message intercepted by the United States navy that was addressed to a Japanese naval officer in 1941 to reveal its contents. It is actually doable by hand, but you may use Python as well.

---

19 17 17 19 14 20 23 18 19 8 12 16 19 8 3 21 8 25 18 14 18 6 3 18 8 15 18 22 18 11

References

Copeland, B.J., 2004. Essential Turing: Classic Writings on Minds and Computers. Oxford University Press, Oxford, UK.

End of Demo