I created a program that models a Turing Machine, also referred to as a tm. The basic function of the program is to read a message. Once the message is read, the code will step through the message and the machine through multiple steps for encryption or decryption, whichever option is selected by the user. The steps used include reading the tape or message, encrypting each character, and stopping once the message has gone through all the steps and is either fully encrypted or decrypted. The program attempts to include all the basic parts of a Turing Machine and then display it in a visual way that helps the user understand the encryption process and what is actually happening to achieve the final product of the Turing machine. Each step is straightforward. The program looks at what it sees, follows a rule, does something, and moves to another state. If it doesn't see a rule, it stops and rejects it. This program has an element that allows it to be easily understood. So, in a short summary, you see a part of the message inputted for encryption and then an area to show what was done to the letter to encrypt it. There is also a switchboard and 2 keyboards to show the letter and the letter replacing it for the encryption. The program will run until either it runs into an issue or the message is fully encrypted. The user can also speed up the process to either see the encryption process slowly or to just speed up the process to get the full new encrypted message faster.

For the encryption, I have used a system that is similar to the Enigma substitution technique. The TM begins from the first character on the message and will continue to go from that character left until it reaches the end. The TM will then replace the character with another character based on the settings, such as the ring setting or rotor order. The new character that gets produced due to the encryption is then put back as the new character on the tape. Finally, the TM moves one step to the right. The TM halts and accepts if it has scanned the entire string, ending in a blank

character. The ciphertext is now on the tape. The encryption scheme uses three rotors and a reflector. In some cases, the plugboard and the ring can also be specified. The basic idea behind the encryption scheme is that the TM is not using a fixed mapping. The rotors move as the TM scans the string. Hence, the mapping changes each time we encrypt a character. This is the reason the character changes to different letters based on the position. In order to keep this within the TM model and avoid the use of hidden variables, we have now incorporated the positions of the rotors as part of the TM's state. The state of the TM is now not only 'scan' but 'scan and the positions of the rotors' window.'

The reason for this is that the Enigma cipher of this type is symmetric. If the same rotor wiring is used, the same rotor order, the same plugboard connections, and the same starting positions for the rotors, then the machine will run the ciphertext through and produce the original plaintext. The interface provides for the full loop. It will run on the plaintext, which then produces ciphertext, and run the ciphertext again with the same settings to produce the original plaintext. The trace really helps a lot. It provides the TM view and the cipher view. The view of the ciphertext being produced makes the output of the TM not look like random output.

One of the global issues that is associated with the use of encryption is wartime intelligence sharing. In the World War II era, different nations relied on sharing secret communications to coordinate different military forces, ships, and supplies. In addition, the United Kingdom established significant efforts in codebreaking and signals intelligence, and sharing intelligence with the United States was an important factor in Allied victory. This is an example of interdependence because a country's security could depend on other nations' people and machines. For example, even if a country could intercept enemy communications, it would not

be able to decrypt them in time without sharing intelligence. In addition, sharing intelligence would be important because otherwise, the shared information would not be useful if it were not shared in time.

At the same time, it also creates inequality. The reason for this is that strong encryption, as well as codebreaking, requires certain resources. These include experts, special equipment, and large budgets. Only powerful countries can afford such equipment, while smaller ones are unable to. The same goes for people. Even within alliances, intelligence sharing is not equal. It can be withheld or delayed for political reasons. This means that some people benefit more from such information than others. Encryption, therefore, establishes two layers at once. It brings countries closer to each other due to their common need for security, but it also makes them more unequal regarding their access to secure systems.

My positionality plays a role in how I see this. As a computer science student, I think about encryption as a way to protect privacy and make communication more secure. I also have a biased view as I love cybersecurity, so these encryptions interest me more as a CS student, but also give me a little more enthusiasm for the project. At the same time, this project has made me think about how encryption can be related to power. In history, it has been a way to protect people, but it has also been a way to maintain state secrets and state surveillance. The creation of a simulator makes it seem logical, but it is not always that way. So, I see it as a protective tool and a political tool, depending on who has it and who it is against.