Let me first delve into the preliminary phase, which consisted of research and discussions pertaining to which topic our team should choose. The first thing each one of our group members did was make a speed run through the topics listed by the course, from which we had to choose, and come up with a shortlist. After doing this, we had a joint meeting where we discussed the pros and cons of all the shortlisted topics. After weighing them as such, we kept 2 major options, which we then took to the meeting we had with professor Roberto Guanciale and clearly highlighted to him what we found on either topic, namely The Onion Router (TOR) and code obfuscation. After the said meeting, we carried out another team meeting among the 4 of us, and voted on one topic which we would eventually implement. Our subject of choice ended up being the router.

Once we got the election of the topic out of the way, all 4 of us started researching on how to implement TOR. After doing this, we split ourselves into teams of 2, and each team eventually split its corresponding tasks among themselves. Following this division, my primary responsibility was related to the cryptography side of things. Hence, I wrote several functions in the onionRouter.py file of the feat/onion-router branch. Firstly, I wrote the Diffie-Hellman key exchange protocol, implemented as taught in the lectures. Then, I performed extensive research to find what the most suitable encryption scheme for our packets, and, upon discussing with my peers as well, ended up going for Fernet. Consequently, I implemented the Fernet encryption and decryption mechanisms for our packets. During the encryption/decryption phase, we write the padded key into a file, called pass.key. Hence, I also wrote a call_key() method that allows us to retrieve the key from the said file, and encrypt/decrypt it using a newly generated Fernet object. For the decryption phase, an adjustment had to be made concerning the mathematical procedure, because the received packet also contains the circuit ID, an identifier that all TOR packets have. Hence, we first separate the circuit ID from the rest of the packet, then decode the packet using Fernet, and obtain the whole package in its entirety again by concatenating the encoded circuit ID, the packet, and the corresponding padding after each other.

Furthermore, I implemented the destroyControllCell() and createdControllCell() methods, which are part of the steps that a TOR network goes through during its configuration. The destroyControllCell() functionality is normally called after successfully sending data through a circuit, in order to close the circuit, and also helps for failure management. The createdControllCell() is called after a circuit has been established, to confirm its successful establishment.

It is important to also mention that our team has held meetings multiple times a week to test the latest modifications in the code and check for any necessary tweaks. I have never missed a meeting of the sort, and neither have my colleagues.