5 Testing

5.1 Unit Testing

The units being tested in our design are the NaCl encryption protocol and the CRC authentication mechanism.

To test the NaCl encryption protocol, we will assume that the authentication protocol has been passed by each ECU by populating each with the correct public key and a valid private key. We will then utilize our simulated CAN bus to ensure that the ECUs are able to communicate with one another effectively by monitoring input and output using the cansniffer function on our Linux VM.

To test the CRC authentication mechanism, we will begin our simulation on our simulated CAN bus in a state where three ECUs are connected to the system. One ECU will have a valid private key and the correct public key. The other ECU will have an invalid private key and the correct public key. The third ECU will also have a valid private key and the correct public key, and it will be used to show the output on a simulated vehicle from the information being relayed by the bus. The ECU with the valid private key will ideally be able to control the virtual vehicle's output. The ECU with the invalid private key, on the other hand, will ideally not be able to control the virtual vehicle's output.

5.2 Interface Testing

The project is based off a key exchange method development which makes use of algorithms and scripts. The interface that we would ideally deal with would be Simulink, CAN Sniffer and multiple ECUs that has dedicated computer boards running to send and receive packets through the CAN network. We might deal with a physical CAN network (hardware) as we progress into next semester where we get to test the algorithms and design that we came up with. The interface that we have been using for current testing purposes is a Linux VM that simulates the CAN network, a sniffer tool and data from multiple ECUs as they are being controlled with sample data. The interfaces involved help the testing of the design more than putting them to test. We assume there no vigorous testing needs to be done on the interface itself rather than the script and the design we have for CAN frames.

5.3 Integration Testing

We have decided to use the Big Bang Integration testing approach. This approach requires us to finish developing all components and modules in order to start testing the system. It is convenient for our project since our implementation will be done in a small CAN Bus network system, which requires each node in the CAN network to have a good connection. It also covered all modules that we developed to ensure that they integrate well with the original CAN Bus network.

For testing, we will first use the virtual environment that was given by the client to test the functionalities of our implementation to make sure the result is what we expected. After integrating our implementation with the original CAN Bus network, we will then use Simulink to test the connectivity between each node in the CAN network with some test cases and analyze the response or behavior of the ECU after receiving different kinds of packets. If everything goes as planned, the final step will be to test our design implementation on the physical hardware model.

5.4 System Testing

The system level testing requires end-to-end testing of all units, interfaces and design that we have developed so far in par with the design we have proposed. To strategic approach on testing the system would be having the physcial CAN bus system setup (hardware) with multiple ECUs and implementing our design (algortihm) for secure key exchange and message passing. This level of system test is only feasible next semester as the client would be able to provide us access to the Simulink (simulation environment) to test our design implementation and then proceed with the physical CAN network. In the testing phase, we should be able to sniff the packets and no key is revealed in the sniffer, and ECUs should be able to function according to the command sent/received between each other. This make the whole system testing (end-to-end testing) successful.

5.5 Regression Testing

Regression testing will be the primary means to ensure added functionality does not break previous software. These assessments will iterate through dependencies and other conditional statements with respect to existing code. Running regression tests at the end of each software development sprint will guarantee all behaviors are captured. It is worth noting, at a minimum, regression tests will be run after critical features have been added; CAN communication, CRC exchanges between ECUs, and decrypting the messages following the NaCl protocol. GitLab will be used to assist regression tests through its version control capabilities. Other tools - such as the CAN sniffer, Simulink, etc. - will be used to verify the code's functionality. Each of these steps will facilitate our software development and will be used as needed.

5.6 Acceptance Testing

Acceptance testing will largely come into play closer to the end of the project when we are demonstrating the overall working project to our client. That being said, we want to work throughout the project duration to ensure that we are meeting the requirements put forward by the client and his desires. Our client has an example CAN bus & ECU system set up in sukup that we will use to demonstrate our protocols on hardware. When demonstrating in person, key testing results include:

- Successfully importing the algorithm into the CAN system.
- Successfully sending input (accelerate, brake, turning, etc) to the controller.
- Ensure that messages between ECU units are getting to the correct destination.
- Analysing packets to determine the encryption from NaCl

5.7 Security Testing (if applicable)

To security test our project, we will need to test both the message security and ECU authentication aspects of the system. For our security testing mechanism, we plan to connect 2 ECSs to the simulated CAN bus. One of the ECUs will have a valid private key which will allow it to be authenticated via CRC, and it will be able to read messages by decrypting the NaCl encryption protocol. The second ECU will not have a valid private key, so it will be unable to be authenticated via CRC, and it will not be able to decrypt messages due to the NaCl encryption protocol. Thus, the first ECU will be able to monitor and/or control the systems on the CAN bus, but the second ECU will not.

5.8 Results

While we have not tested a final design just yet, the way we are planning to do this is detailed above. If the algorithm is implemented correctly via the design process explained in the previous sections, then what we will be expecting is a fully encrypted stream of communication between each ECU. This entails having every input signal lead to a different ciphertext output, so that any malicious listener wouldn't be able to decipher which outputs came from which inputs.

Our testbed will look similar to what is shown below