Configuration Manual

MSc Research Project
MSc in Cybersecurity

Sudhanshu Mane
Student ID: 22114599

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sudhanshu Mane		
Student ID:	22114599		
Programme:	MSc in Cybersecurity	**Year:**	2023-24
Module:	Research in Computing		
Lecturer:	Khadija Hafeez		
Submission Due Date:	14/12/2023		
Project Title:	Simulation of False Data Injection Attack (FDIA) & Threshold-Based Detection Strategy against (FDIA)		
Word Count:	1005 words **Page Count:** 8 Pages		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Sudhanshu Sunil Mane

Date: 14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sudhanshu Mane
Student ID: 22114599

1 System requirements for MATLAB

Operating System [1]:

I. Windows 11
II. Windows 10
III. Windows Server 2019
IV. Windows Server 2022

Processor & RAM [1]:
Minimum any intel processors or AMD x86-64 processor with two or more core. Recommended any intel or AMD x86-64 processor with four or more cores and AVX2 instruction set support. Minimum 8GB and recommended 16 GB RAM required.

Internet Browsers:

I. Google Chrome
II. Mozilla Firefox
III. Brave Browser
IV. Microsoft Bing

2 Starting with MATLAB

1] Go to link: [2] and click on ‘Get MATLAB’.

2] Create an account or login if you already have an account using your university email or work email and then verify the email to get started.

3] After logging in Click on “Open MATLAB online (Basic)”. This takes you to your work page.

4] After getting to the work page press “CTRL + N” or go to ‘New’ and then ‘New Script’.

3 Running the False Data Injection Attack (FDIA) simulation

After opening the new script page use the code to run the False Data Injection Attack Simulation. To run the simulation, we need to save the file by pressing ‘CTRL + S’ to save the file. You need to save the file with ‘.m’ extension. The Simulation of False data injection attack code file is uploaded with the name of ‘testupd2.m’.

```
vehicle1.speed = 60;
receivingvehicle2.speed = 0;

% Time INTErval Settings
timeinterval = 1;
total_time = 120;
time_vector = 0:timeinterval:total_time;

% for data storing
vehicle1_speed_data = zeros(1, length(time_vector));
receivingvehicle2_speed_data = zeros(1, length(time_vector));
attack_vector = zeros(1, length(time_vector));

% storage for after and before attack
speed_before_attack = 0;
speed_after_attack = 0;
attack_start_time = 60;
attack_duration = 30;
attack_end_time = attack_start_time + attack_duration;


for t = 1:length(time_vector)

    vehicle1_speed_data(t) = vehicle1.speed;
    receivingvehicle2.speed = vehicle1_speed_data(t);
    receivingvehicle2_speed_data(t) = receivingvehicle2.speed;

    % FDIA payload
    if time_vector(t) >= attack_start_time && time_vector(t) <= attack_end_time
        vehicle1_speed_data(t) = vehicle1.speed * 1.5;
        attack_vector(t) = 1;
    end
```

Figure[1] FDIA SIMULATION

In figure 1, the parameter ‘vehicle1.speed’ is defined for attacker vehicle who is transmitting data to ‘receivingvehicle2.speed’ who is victim vehicle. ‘timeinterval’ is used to display speed of attacker vehicle at each 1 second.

```
% Printing output
fprintf('\nSpeed before attack: %.2f km/h\n', speed_before_attack);
fprintf('Speed after attack starts: %.2f km/h\n', speed_after_attack);

% Visualization
figure;
plot(time_vector, vehicle1_speed_data, 'b', 'DisplayName', 'Transmitted Speed');
hold on;
plot(time_vector, receivingvehicle2_speed_data, 'r', 'DisplayName', 'Received Speed');
plot(time_vector, attack_vector * max(vehicle1_speed_data), 'k--', 'DisplayName', 'FDIA Impact');


plot(attack_start_time, speed_before_attack, 'go', 'MarkerFaceColor', 'g', 'DisplayName', 'Speed Before Attack');
plot(attack_end_time, speed_after_attack, 'mo', 'MarkerFaceColor', 'm', 'DisplayName', 'Speed After Attack');

title('Simulation of FDIA in V2V Communication');
xlabel('Time (seconds)');
ylabel('Speed (km/h)');
legend show;
```

Figure [2] FDIA SIMULATION

In figure 2, the visualization parameters are set.

4 Existing Dataset

The dataset was taken from [3] which provides electric vehicles information. The dependent information was taken from the dataset. The two test cases were taken from the dataset named as ‘ElectricCarData_Clean.csv’.

	A	B	C
1	Brand	Model	AccelSec
2	Tesla	Model 3 Long Range Dual Motor	4.6

Figure 3: Dataset values taken

5	BMW	iX3	6.8

Figure 4: Dataset values taken

The Figure 3 and Figure 4, shows the values taken from dataset to create two test cases.

4.1 Steps for downloading Dataset

1] Go to Kaggle’s official website: [3] and click on download.
2] Two datasets will displayed in zip file.
3] Extract the file named ‘ElectricCarData_Clean.csv’.

5 Threshold-Based Detection Simulation

1] Click on ‘New’ and then select ‘Script’. After opening the new script page use the code to run the Threshold Based Detection for FDIA. To run the simulation, we need to save the file by pressing ‘CTRL + S’ to save the file. You need to save the file with ‘.m’ extension.

2] There are two test cases for detection. 1st test case is Tesla Model3 Long Range Dual Motor and 2nd test case is BMW iX3.

3] The simulation of ‘Threshold Based Detection for FDIA’ code file is uploaded with the name of ‘anomalydet2.m’. for 1st test case.

4] The simulation of The simulation of ‘Threshold Based Detection for FDIA’ code file is uploaded with the name of ‘detectiontest2.m’. for 2st test case.

Brand	**Model**	**Time taken (Final Velocity)**	**Acceleration per Second**
Tesla	Model 3 Long Range Dual Motor	4.6 seconds	21.74 km/h
BMW	iX3	6.8 seconds	14.71 km/h

5.1 For 1st test case:

```
attackingtransmitter.speed = 60;
victimreceiver.speed = 0;

% Time INTErval Settings
timestep = 1;
total_time = 120;
time_vector = 0:timestep:total_time;

% for data storing
vehicle1_speed_data = zeros(1, length(time_vector));
receivingvehicle2_speed_data = zeros(1, length(time_vector));
attack_vector = zeros(1, length(time_vector));
anomaly_detected_vector = zeros(1, length(time_vector));

% Threshold
max_acceleration = 21.74;

% storage for after and before attack
speed_before_attack = 0;
speed_after_attack = 0;
attack_start_time = 60;
attack_duration = 30;
attack_end_time = attack_start_time + attack_duration;


for t = 1:length(time_vector)

    vehicle1_speed_data(t) = attackingtransmitter.speed;
    victimreceiver.speed = vehicle1_speed_data(t);
    receivingvehicle2_speed_data(t) = victimreceiver.speed;

```

Figure 3 : Threshold-Based Detection for False Data Injection Attack

The 1st test case shows attack has occurred as per the threshold values set by parameter 'max_acceleration = 21.74'.

```
    % FDIA payload
    if time_vector(t) >= attack_start_time && time_vector(t) <= attack_end_time
        vehicle1_speed_data(t) = attackingtransmitter.speed * 1.5;
        attack_vector(t) = 1;
    end

    % Printing output
    fprintf('Time %d sec - Speed: %.2f km/h\n', time_vector(t), vehicle1_speed_data(t));

    % Printing output2
    if time_vector(t) == attack_start_time
        speed_before_attack = victimreceiver.speed;
    elseif time_vector(t) == attack_end_time && speed_after_attack == 0
        speed_after_attack = vehicle1_speed_data(t);
    end

% Threshold-Based Detection
if t > 1 && abs(vehicle1_speed_data(t) - vehicle1_speed_data(t-1)) > max_acceleration
    fprintf('False Data Injection Attack Detected at time %d sec\n', time_vector(t));
    anomaly_detected_vector(t) = 1;
end

end

% printing outputs 3
fprintf('\nSpeed before attack: %.2f km/h\n', speed_before_attack);
fprintf('Speed after attack starts: %.2f km/h\n', speed_after_attack);

```

Figure 4: Threshold-Based Detection for False Data Injection Attack

In figure 4, ‘% FDIA payload’ the ‘attackingtransmitter.speed’ parameter is multiplied by 1.5 to invoke ‘% Threshold-Based Detection’.

```
% Visualization
figure;
plot(time_vector, vehicle1_speed_data, 'b', 'DisplayName', 'Transmitted Speed');
hold on;
plot(time_vector, receivingvehicle2_speed_data, 'r', 'DisplayName', 'Received Speed');
plot(time_vector, attack_vector * max(vehicle1_speed_data), 'k--', 'DisplayName', 'FDIA Impact');
plot(time_vector, anomaly_detected_vector * max(vehicle1_speed_data), 'g*', 'DisplayName', 'Anomaly Detected');
plot(attack_start_time, speed_before_attack, 'go', 'MarkerFaceColor', 'g', 'DisplayName', 'Speed Before Attack');
plot(attack_end_time, speed_after_attack, 'mo', 'MarkerFaceColor', 'm', 'DisplayName', 'Speed After Attack');

title('Simulation of FDIA in V2V Communication with Anomaly Detection');
xlabel('Time (seconds)');
ylabel('Speed (km/h)');
legend show;
```

Figure 5: Threshold-Based Detection for False Data Injection Attack

In conclusion 1st test case is simulated to show attack and detection scenario.

5.2 For 2nd Test Case

```
attackingtransmitter.speed = 60;
victimreceiver.speed = 0;

% Time INTErval Settings
timestep = 1;
total_time = 120;
time_vector = 0:timestep:total_time;

% for data storing
vehicle1_speed_data = zeros(1, length(time_vector));
receivingvehicle2_speed_data = zeros(1, length(time_vector));
attack_vector = zeros(1, length(time_vector));
anomaly_detected_vector = zeros(1, length(time_vector));

% Threshold
max_acceleration = 14.71;

% storage for after and before attack
speed_before_attack = 0;
speed_after_attack = 0;
attack_start_time = 60;
attack_duration = 30;
attack_end_time = attack_start_time + attack_duration;


for t = 1:length(time_vector)

    vehicle1_speed_data(t) = attackingtransmitter.speed;
    victimreceiver.speed = vehicle1_speed_data(t);
    receivingvehicle2_speed_data(t) = victimreceiver.speed;

```

Figure 6 Threshold-Based Detection for FDIA 2nd Test Case.

The 1st test case shows attack has occurred as per the threshold values set by parameter ‘max_acceleration = 14.71’.

```
        % FDIA payload
        if time_vector(t) >= attack_start_time && time_vector(t) <= attack_end_time
            vehicle1_speed_data(t) = attackingtransmitter.speed * 1.2;
            attack_vector(t) = 1;
        end

        % Printing output
        fprintf('Time %d sec - Speed: %.2f km/h\n', time_vector(t), vehicle1_speed_data(t));

        % Printing output2
        if time_vector(t) == attack_start_time
            speed_before_attack = victimreceiver.speed;
        elseif time_vector(t) == attack_end_time && speed_after_attack == 0
            speed_after_attack = vehicle1_speed_data(t);
        end

    % Threshold-Based Detection
    if t > 1 && abs(vehicle1_speed_data(t) - vehicle1_speed_data(t-1)) > max_acceleration
        fprintf('False Data Injection Attack Detected at time %d sec\n', time_vector(t));
        anomaly_detected_vector(t) = 1;
    end

    end
```

Figure 7: Threshold-Based Detection for FDIA 2nd Test Case.

In figure 4, '% FDIA payload' the 'attackingtransmitter.speed' parameter is multiplied by 1.2 to check '% Threshold-Based Detection' algorithm.

```
% printing outputs 3
fprintf('\nSpeed before attack: %.2f km/h\n', speed_before_attack);
fprintf('Speed after attack starts: %.2f km/h\n', speed_after_attack);

% Visualization
figure;
plot(time_vector, vehicle1_speed_data, 'b', 'DisplayName', 'Transmitted Speed');
hold on;
plot(time_vector, receivingvehicle2_speed_data, 'r', 'DisplayName', 'Received Speed');
plot(time_vector, attack_vector * max(vehicle1_speed_data), 'k--', 'DisplayName', 'FDIA Impact');
plot(time_vector, anomaly_detected_vector * max(vehicle1_speed_data), 'g*', 'DisplayName', 'Anomaly Detected');
plot(attack_start_time, speed_before_attack, 'go', 'MarkerFaceColor', 'g', 'DisplayName', 'Speed Before Attack');
plot(attack_end_time, speed_after_attack, 'mo', 'MarkerFaceColor', 'm', 'DisplayName', 'Speed After Attack');

title('Simulation of FDIA in V2V Communication with Anomaly Detection');
xlabel('Time (seconds)');
ylabel('Speed (km/h)');
legend show;
```

Figure 8: Threshold-Based Detection for FDIA 2nd Test Case.

In conclusion, 2nd Test Case is to show normal transmission within the threshold scenario.

References

[1] "System Requirements for MATLAB." Accessed: Dec. 14, 2023. [Online]. Available: https://uk.mathworks.com/support/requirements/matlab-system-requirements.html

[2] "MATLAB." Accessed: Dec. 14, 2023. [Online]. Available: https://uk.mathworks.com/products/matlab.html

[3] "EVs - One Electric Vehicle Dataset - Smaller." Accessed: Dec. 14, 2023. [Online]. Available: https://www.kaggle.com/datasets/geoffnel/evs-one-electric-vehicle-dataset