Concordia Institute for Information System Engineering (CIISE)

INSE 6630 Recent Developments in Information Systems Security

Final Project Report

Submitted To :

Prof. Walter Lucia

Submitted By :

Name	ID #
Shreya Monpara	40105352
Jaldhi Prajapati	40107457
Hetvi Shah	40089272

Summer 2020

ACKNOWLEDGEMENT

We take this opportunity with much pleasure to thank all the people who have helped us through the course of our journey towards producing this report. We sincerely thank our guide Prof. Dr. Walter Lucia for their guidance, help and motivation. Apart from the subject of our project, we learnt a lot from him, which we are sure, will be useful to us in different stages of our life. We would like to express our gratitude to them for much help with the project design and methodology, and for review and many helpful comments. We are also thankful to him for being supportive in such pandemic situations. We also pay our respects to our teaching assistant Shima Savehshemshaki for the guidelines given to us for the various activities performed during the project. We would like to thank her for the assistance in finalizing the project problem and useful comments. We would like to thank our Concordia University for their caring and supportive attitude.

Shreya Monpara 40105352

Jaldhi Prajapati 40107457

Hetvi Shah 40089272

TABLE OF CONTENTS

TABLE OF FIGURES

Project Overview

In this project, we are going to analyze the Aircraft Systems. Different design tasks have been performed including its stimulation in Matlab, namely False Data Injection attack, Replay attack, Cover attack. The attack detection techniques were also performed such as, Chi-square detection technique ,Watermarking technique and Moving-target technique on the model.

Part 1 Implementation of False Data Injection Attack.

Implement an additive FDI attack ua(t) on the actuation channel such that the resulting corrupted command u′(t) = u(t) + u^a (t) has the following temporal behavior:

$$u^a(t) = \begin{cases} 0, & if \ \ 0 \leq t < 40 \\ [-0.18 \quad -0.18]^T, & if \ \ 40 \leq t < 50 \\ 0, & if \ \ t \geq 50 \ \ \text{(the attack is over)} \end{cases}$$

and study the effect of this attack on the plant behaviors.

#	Question	Short Answer
Part 1-a.	Does the system violate the plant constraints under attack?	Yes, the system violates the plant constraints under the attack as the system is not in the desired equilibrium state during the attack and it affects the plant.
Part 1-b.	Does the plant reach the desired equilibrium during the attack?	No, the plant does not reach the desired equilibrium during the attack.
Part 1-c.	Does the plant recover the desired equilibrium after the attack?	Yes, the plant recovers the desired equilibrium after the attack.

To support the answers, below is its simulation. Fig 1. is the simulation of the false data injection attack. When the attack is performed in the actuation channel, there is a deviation in the system.

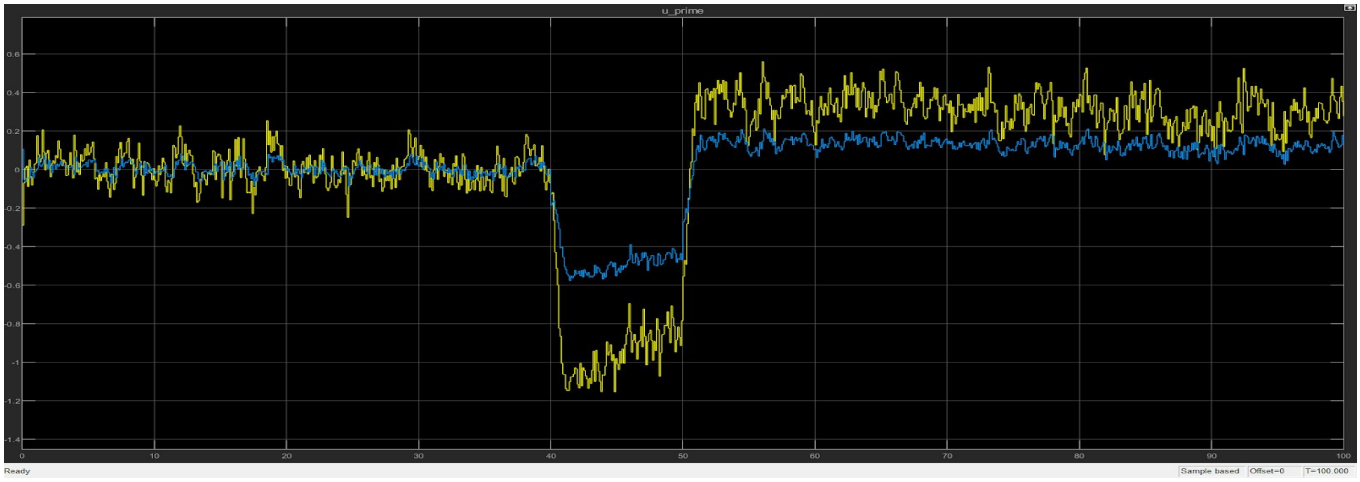

Figure 1. False Data Injection Attack.

Part 2 Chi Square Detection Technique

Design 2 different attack detection schemes capable of detecting the following FDI attack on the actuation channel. By dening u0(t) = u(t) + ua(t) the attack has the following temporal behavior

$$u^a(t) = \begin{cases} 0, & if \ 0 \leq t < 40 \\ [-0.18 \quad -0.18]^T, & if \ 40 \leq t < 50 \\ 0, & if \ 50 \leq t < 300 \\ [-0.21 \quad -0.21]^T, & if \ 300 \leq t < 310 \\ 0, & if \ t \geq 310 \end{cases}$$

Tune the detectors to maximize the performance index J_d.

#	Question	Short Answer
Part 2-a.	For the 2 detector, averaging the result over 50 trials, report $f_{alarm}(0; 350)$; $d_{rate}(0; 350)$; $J_d(0; 350)$	The final average result over 50 trials of : $f_{alarm}(0; 350)$ = 2.4112% $d_{rate}(0; 350)$ = 77.0435% $J_d(0; 350)$ = 32.3330 unit
Part 2-b.	For the second detector, averaging the result over 50 trials, report $f_{alarm}(0; 350)$; $d_{rate}(0; 350)$; $J_d(0; 350)$	**Paper title**: CUSUM and Chi-Squared Attack Detection for Compromised Sensors by Carlos Murguia and Justin Ruths The final average result over 50 trials of : $f_{alarm}(0; 350)$ = 1.7601% $d_{rate}(0; 350)$ = 30.5078% $J_d(0; 350)$ = 17.4942 unit

The following simulation is used to support the above answers. Fig 2. is the simulation of the Chi-Square algorithm. The analysis of different values of α for chi-square detector are shown below:

	$f_{alarm}(0; 350)$	$d_{rate}(0; 350)$	$J_d(0; 350)$
For α = 0.001	0.2794%	46.7305%	157.9286
For α = 0.01	2.4150%	76.9553%	32.3836
For α = 0.05	11.0229%	93.1731%	8.5103
For α = 0.09	18.6621%	96.1969%	5.1994
For α = 0.1	20.0480%	96.5930%	4.8931

Table 1. Analysis of α values.

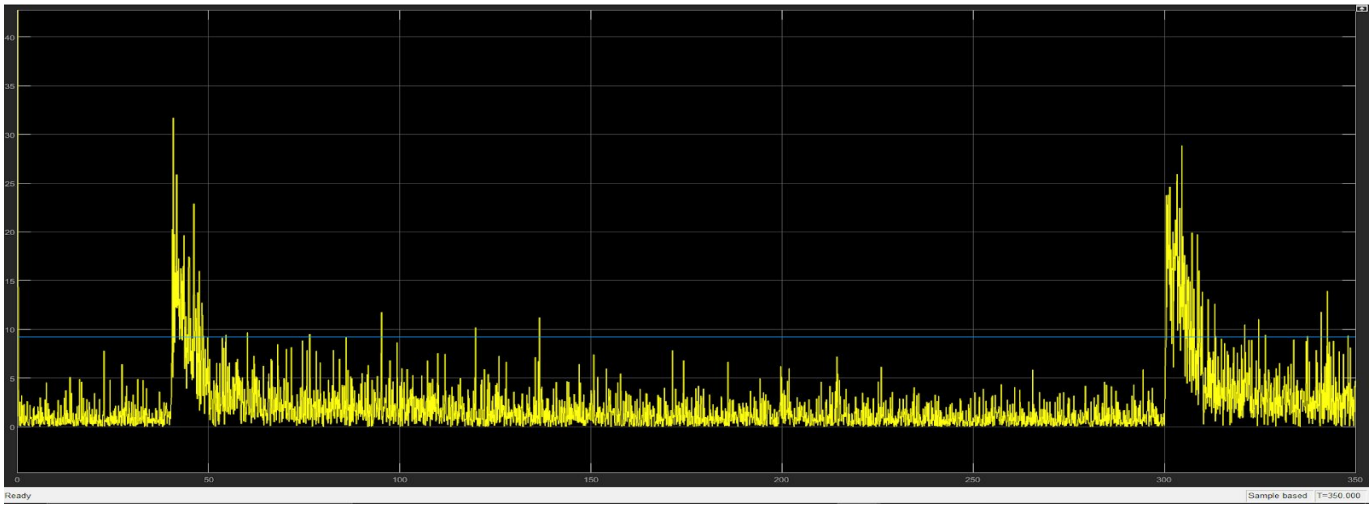

Figure 2.. FDI attack detection using Chi- Square Detection Technique

Below Fig 3. depicts the CUSUM Detection simulation.

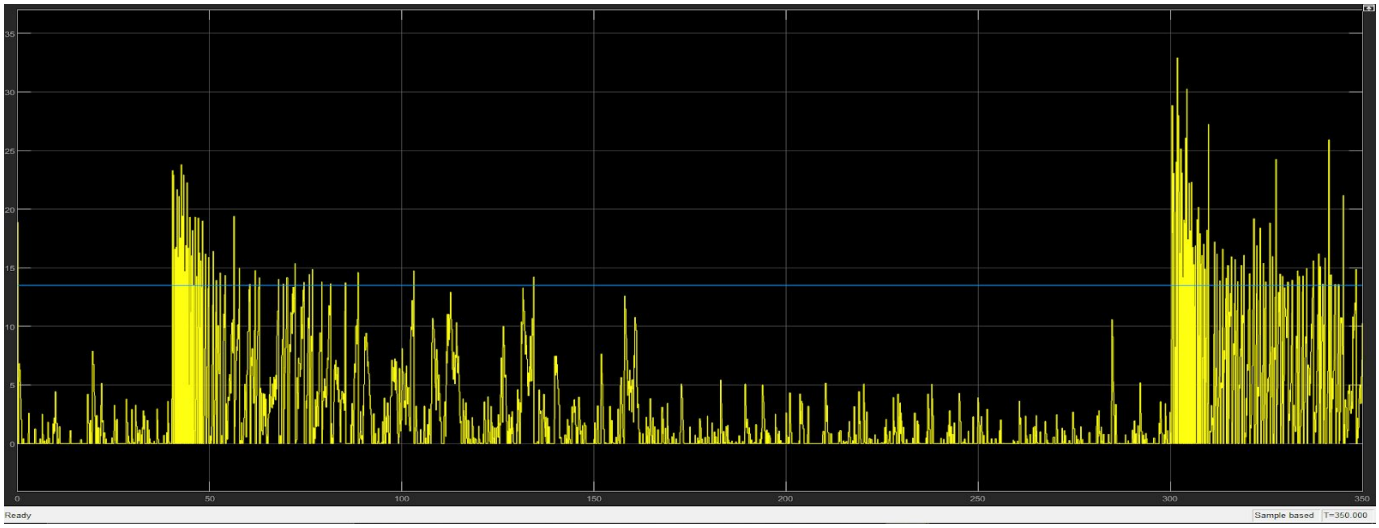

Figure 3. CUSUM Detection Technique

Part 3 Replay Attack

Design the following replay attack when the system is in steady-state condition.

$$u^a(t) = \begin{cases} 0, & if\ \ 0 \le t < 100 \\ [-0.05 \quad -0.05 \quad]^T, & if\ \ 100 \le t < 120 \\ 0, & if\ \ t \ge 120 \ \ \text{(the attack over)} \end{cases}$$

$$y'(t) = \begin{cases} y(t), \ \text{while the attacker records y(t)}, & if\ \ 70 \le t < 100 \\ y(t-20), \ \text{i.e. the attacker replay the recorded y(t)}, & if\ \ 100 \le t < 120 \\ y(t), & if\ \ t \ge 120 \ \ \text{(the attack over)} \end{cases}$$

#	Question	Short Answer
Part 3-a.	Does the system violate the plant constraints under attack?	Yes, it does violate the plant constraint. The plant is affected due to replay attack.

Part 3-b.	Can such attack be detected by the detectors designed in Part 2?	No, the detectors mentioned in Part 2 do not detect the Replay Attack, as chi-square and CUSUM detectors are passive detectors.
Part 3-c.	For the 2 detector, averaging the result over 50 trials, report $f_{alarm}(0,130)$, $d_{rate}(0,130)$, $J_d(0,130)$	The average result of 50 trials for: $f_{alarm}(0,130) = 20.4273\%$ $d_{rate}(0,130) = 13.1063\%$ $J_d(0,130) = 0.6560$ unit
Part 3-d.	For the second detector (if implemented), averaging the result over 50 trials, report $f_{alarm}(0,130)$, $d_{rate}(0,130)$, $J_d(0,130)$	The average result of 50 trials for: $f_{alarm}(0,130) = 5.1525\%$ $d_{rate}(0,130) = 1.3572\%$ $J_d(0,130) = 0.2518$ unit

To support the above answers below are the graphs depicting Replay Attack on Sensor Measurement and Replay Attack on Corrupted Sensor Measurement.

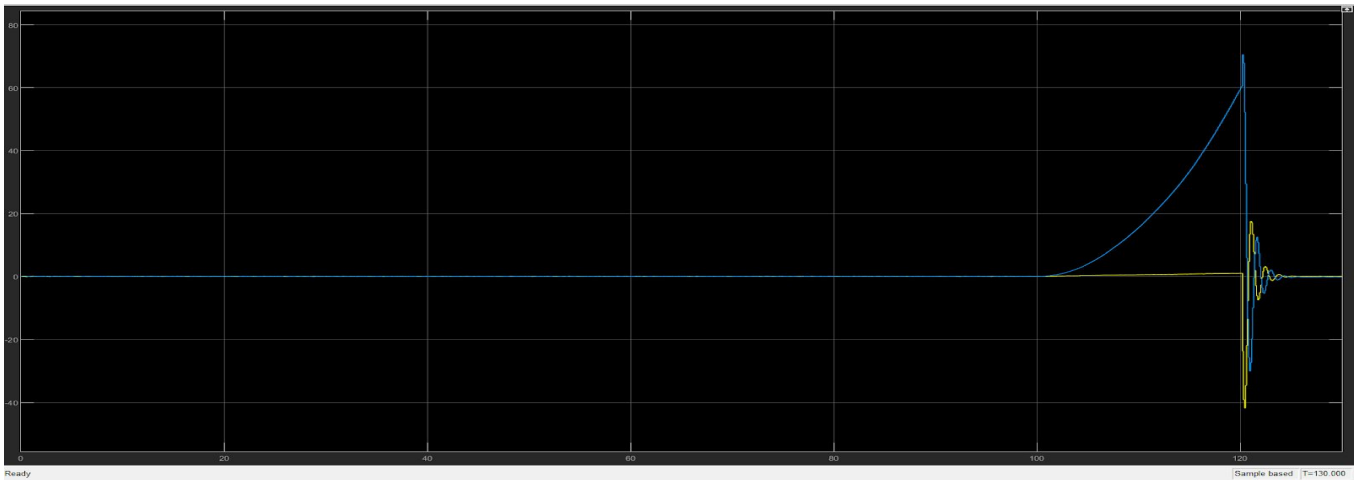

Figure 4.1. Replay Attack on Sensor Measurement

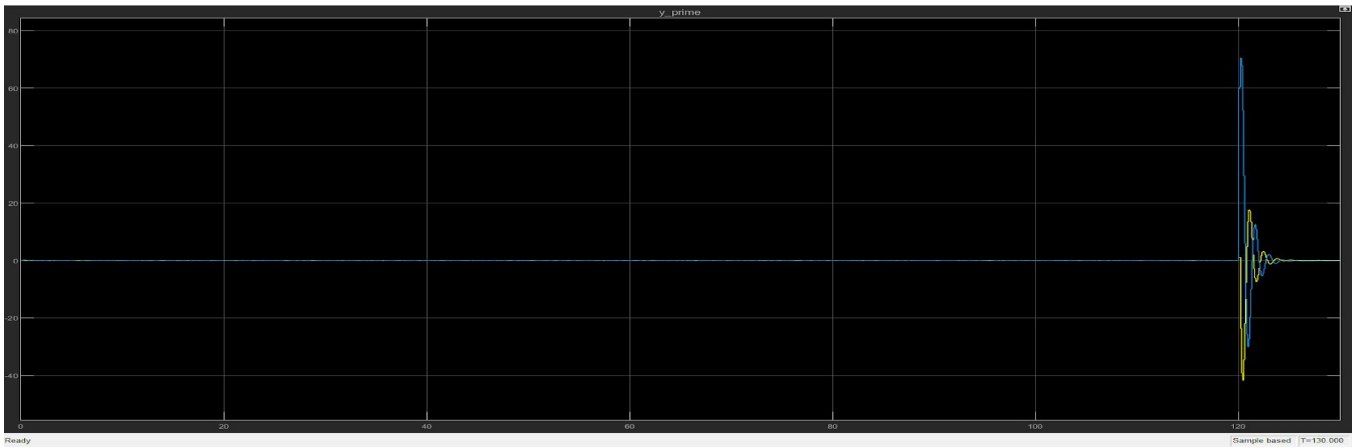

Figure 4.2. Replay Attack on Corrupted Sensor Measurement

[In Fig 4.1, we injected an attack vector between interval 100 to 120 in the actuation channel and after that we stored the data between 80 to 100 and injected between interval 100 to 120]

Part 4 Water-marking Detection Attack

Design watermarking-based detection mechanisms on top of the previously used detection scheme to detect the replay attack in Part 3. Tune the watermarking scheme to maximize the following index

$$J_w(t_1, t_2) = J_w(k_1 T_s, k_2 T_s) = \frac{d_{rate}(k_1 T_s, k_2 T_s)}{J_e(k_1 T_s, k_2 T_s)}$$

quantifying the trade-off between detection rate and control performance loss.

#	Question	Short Answer
Part 4-a.	For the 2 detector+watermarking, averaging the result over 50 trials, report d_{rate} (0; 130); J_e (0; 130) and J_w (0; 130)	The average result of 50 trials for: d_{rate} (0; 130) = 98.3981% J_e (0; 130) = 0.1595 unit J_w (0; 130) = 610.0826 unit
Part 4-b.	For the second detector (if implemented)+watermarking , averaging the result over 50 trials, report d_{rate} (0; 130); J_e (0; 130) and J_w (0; 130)	The average result of 50 trials for: d_{rate} (0; 130) = 48.8692% J_e (0; 130) = 0.0782 unit J_w (0; 130) = 628.3884 unit
Part 4-c.	Does watermarking + χ^2 affect the control performance?	Yes, it affects the performance. The increase in β increases the value of L which is the covariance of the watermarking signal. Thus it decreases the controller performance. So the design problem consists in finding a good compromise between detection rate β and loss of control performance.

In support of above answers, Fig 5.1 illustrates the watermarking + chi-square detection technique for Replay Attack. Fig 5.2 shows the detection done through watermarking + CUSUM

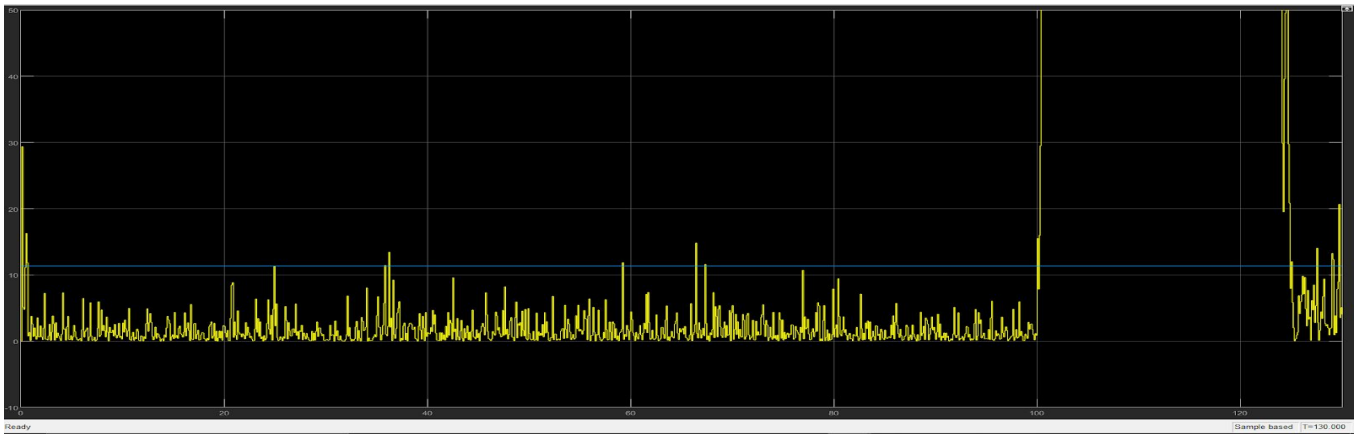

Figure 5.1.Replay Attack Detection using Chi-Square + Watermarking Technique

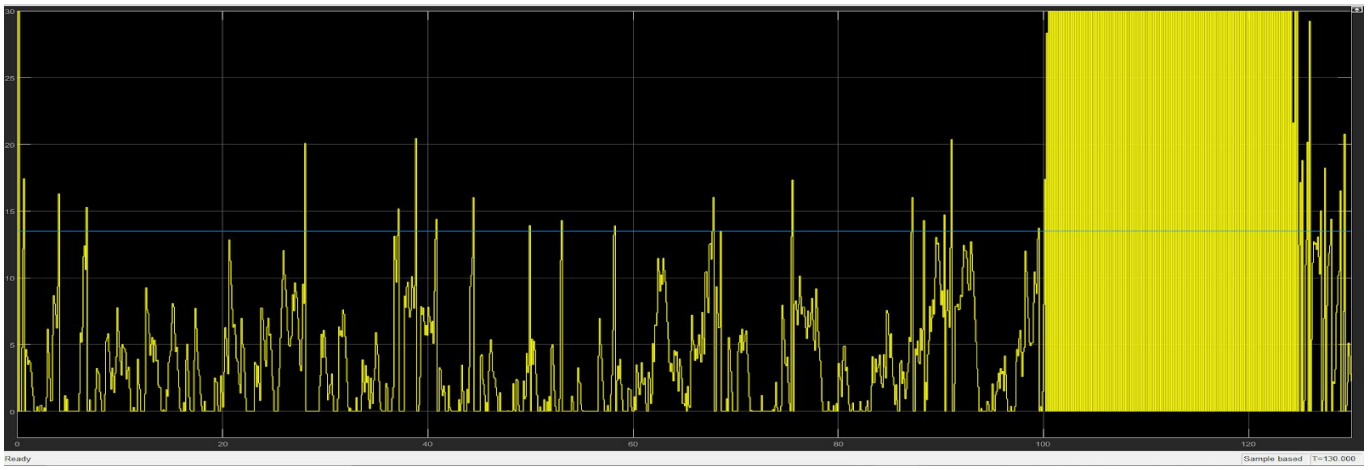

Figure 5.2. Replay Attack Detection using CUSUM + Watermarking Technique

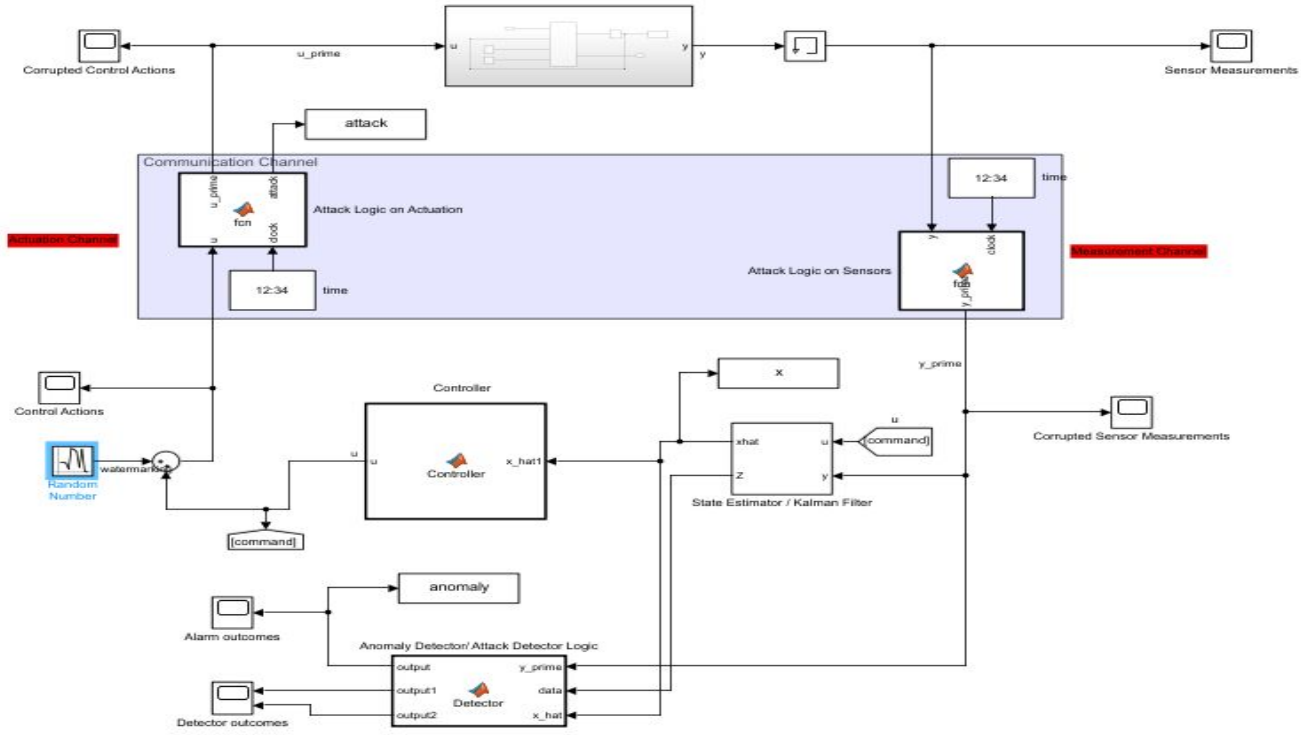

Figure 5.3. Plant Model of Watermarking

[In this watermarking detection technique, we took a Gaussian random signal with zero mean and covariance 0.0001. Then we added this watermarking signal with the output of the controller (u) which is the input signal. Here, we added a signal to live plant data for detection of the replay attack otherwise in a steady state condition replay attack is not detected by chi-square detector. In this plant model we used To Workspace tools (anomaly, attack and x) for calculation of 50 trials of the simulink. Also a memory block was added for creating unit delay in sensor measurement].

Part 5 Covert Attack

Starting from the input attack implemented in part 1, design a covert attack action on the sensor measurements for $40 \leq t \leq 50$ such that the previously implemented χ^2 detector fails to achieve detection. Then, design an ad-hoc strategy for detecting covert attacks.

#	Question	Short Answer
Part 5-a.	Can any of the detectors previously designed, detect a covert attack for $0 \leq t \leq 50$?	No, none of the designed techniques can detect the covert attack so far, as all are passive detectors.
Part 5-b.	Which detection strategy did you use to detect this attack?	Moving target detection (simplified control scheme)
Part 5-c.	For the implemented detection strategy, averaging the result over 50 trials, report $f_{alarm}(0,50)$, $d_{rate}(0,50)$, $J_d(0,50)$, $J_e(0,50)$	The average of detection strategy over 50 trials are: $f_{alarm}(0,50) = 2.2876\%$ $d_{rate}(0,50) = 94.4363\%$ $J_d(0,50) = 40.2823$ unit $J_e(0,50) = 0.0229$ unit
Part 5-d.	Does the implemented detector affect the control performance?	No, the detection mechanism does not affect the control performance, as the static function does not need to be coupled with the plant dynamics.

The detection is done by Moving Target technique. Below Fig 6.1 and Fig 6.2 shows the detection.

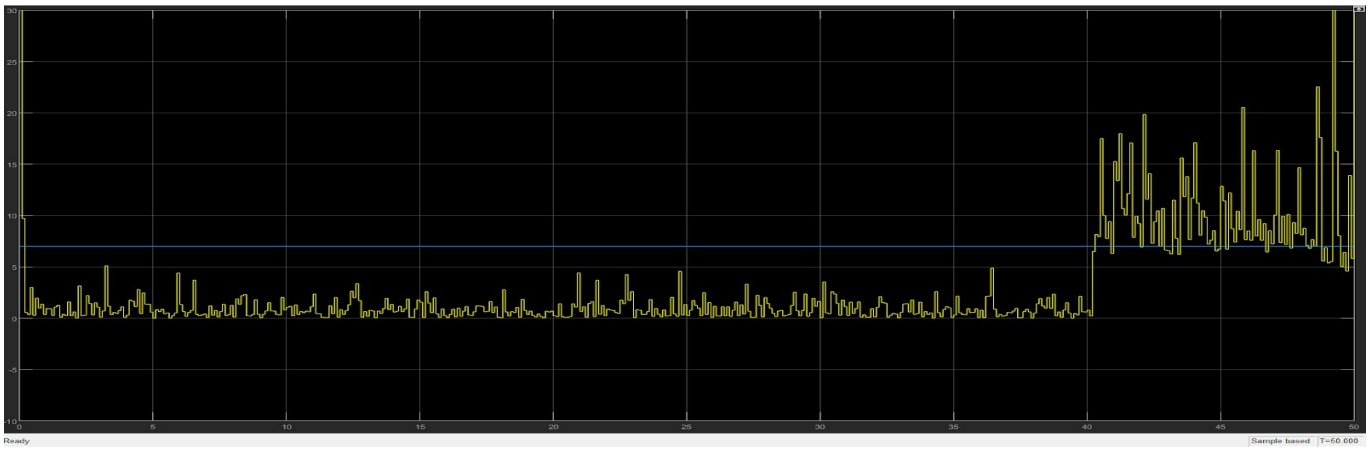

Figure 6.1 Covert Attack Detection using Moving Target

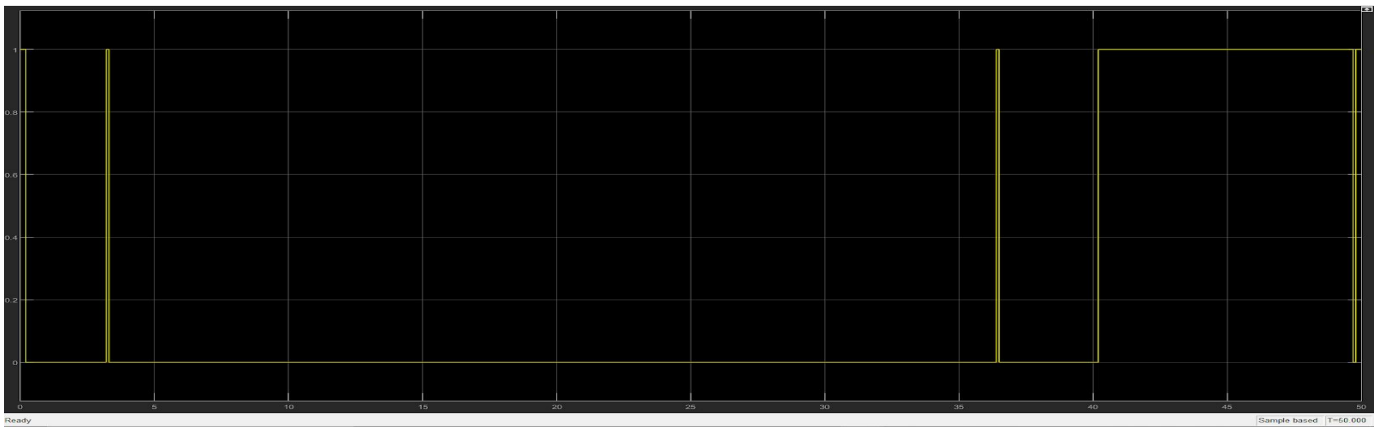

Figure 6.2 Alarm Occurrence Detection using Moving Target Detection Technique

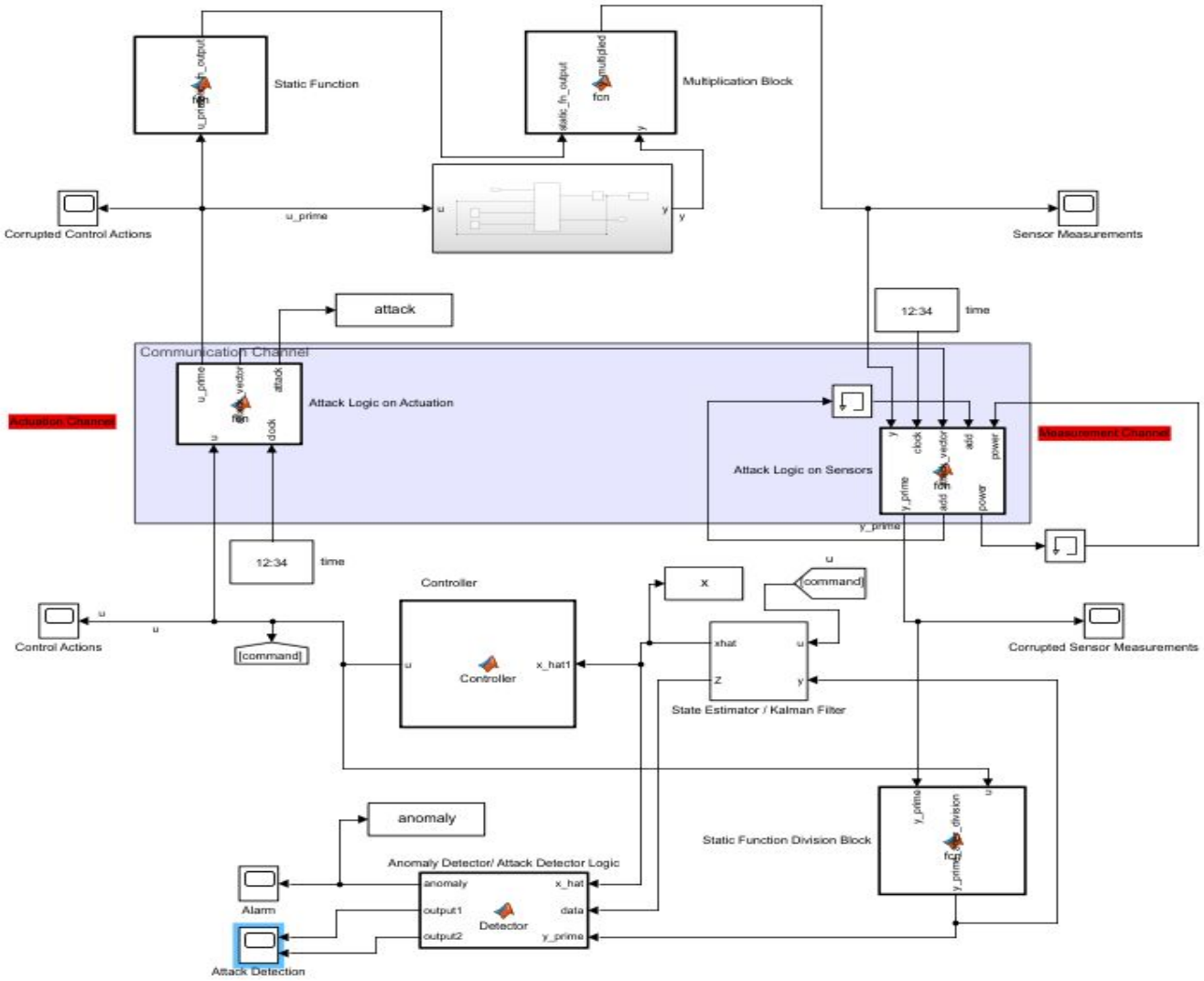

Figure 6.3. Plant model for Moving Target

[In this Moving Target detection technique, we took a nonzero, nonlinear static function x = e ^ (u (k) + 3)(in the Static Function). We multiplied (in the Multiplication Block) it with the output of the plant (y) and sent it using a sensor measurement network channel. Before we sent that corrupted sensor single to the state estimator and detector we did inverse function by dividing (in the Static Function Division Block) corrupted sensor measurement with the static function x = e ^ (u (k) + 3) and get back sensor measurement without corrupted then sent it to the state estimator and the detector. Then we detected False data injection attacks using chi-square detection technique. In this plant model we used To Workspace tools (anomaly, attack and x) for calculation of 50 trials of the simulink.]

APPENDIX

(1) Part 2 (a):Chi-Square Technique

```
C = [1 0 0; 0 1 0]; R = 0.00008*eye(2); P = data;
sigma = (C * P * (transpose(C))) + R;
r = y_prime - (C * x_hat);
z = transpose(r) * (inv(sigma) * r);
alpha = 0.01;
v = size(r);
 tau = chi2inv(1-alpha , v);
if z <= tau
        anomaly = 0;
else
        anomaly = 1;
end
% Calculation of 50 runs of simulation
n = 50;count = 1;falarm = 0; drate = 0; jd = 0;i=0;j=0;k=0;l=0;
for loop = 1:n
         result = sim('project_model_part2_detector2');
        attacks = attack.data; anomalies = anomaly.data;
 for m = 1:3501
         if (attacks(m)== 0 && anomalies(m) == 1)
                 i = i + 1;
         end
         if attacks(m) == 0
                j = j + 1;
        end
        if attacks(m) == 1 && anomalies(m) == 1
                 k = k + 1;
        end
        if attacks(m) == 1
                 l = l + 1;
 falarm = falarm + ((i / j) * 100); drate = drate + ((k / l) * 100); jd = jd + (drate / falarm);
disp(falarm / 50); disp(drate / 50); disp(jd / 50);
```

(2) Part 2 (b): CUSUM Technique

```
C = [1 0 0; 0 1 0]; R = 0.00008*eye(2); P = data;
sigma = (C * P * (transpose(C))) + R;
r = y_prime - (C * x_hat);
z = transpose(r) * (inv(sigma) * r); b = 2; beta = 13.5;
persistent s;
if isempty(s)
    s = 0;
```

```
 end
if s <= beta
   s = max([0,s + z - b]); output = 0;
else
   s = 0; output = 1;
end
```

(3) Part 3: Replay Attack

```
y_prime = y;
persistent i , j, n
if isempty(i)
   i = 1;
end
if isempty(j)
   j = 1;
end
if isempty(n)
   n = zeros(200,2);
end
if clock >= 80 && clock < 100
    n(i,:) = transpose(y);  i = i+1;
elseif clock >= 100 && clock < 120
    y_prime = transpose(n(j,:));  j = j+1;
else
    y_prime = y;
```

(4) Part (4): Watermarking Technique

```
for loop = 1:n
   result = sim('project_model_part4_watermarking');
   attacks = attack.data;  anomalies = anomaly.data; x_data = x.data;
   for m = 1:1301
        if attacks(m) == 1 && anomalies(m) == 1
                k = k + 1;
         end
      if attacks(m) == 1
                 l = l + 1;
      end
      if (attacks(m)== 0 && anomalies(m) == 1)
                normlize = norm(x_data(m) - xeq); i = i + (normlize);
end
      if attacks(m) == 0
        j = j + 1;
      end
drate = drate + ((k / l) * 100);  je = je + (i / j); jw = jw + (drate / je);
% Chi-square detection algorithm
```

```
C = [1 0 0; 0 1 0]; R = 0.00008*eye(2); P = data;
sigma = (C * P * (transpose(C))) + R;
r = y_prime - (C * x_hat);
z = transpose(r) * (inv(sigma) * r);
alpha = 0.01;
v = size(r);
tau = chi2inv(1-alpha , v);
if z <= tau
   anomaly = 0;
else
   anomaly =1;
end
```

(5) <u>Part 5:</u> Covert Attack detection(Moving target)

```
% Logic on actuation channel
attack_vector = transpose([-0.18 ,-0.18]);
if 0 <= clock && clock < 40
   u_a = transpose([0,0]); attack = 0;
elseif clock >= 40 && clock < 50
   u_a = attack_vector; attack = 1;
else  u_a = transpose([0,0]);  attack = 0;
u_prime = u + u_a
% Logic of injecting non-linear and non-zero static function
function static_fn_output = fcn(u_prime)
static_fn_output = exp(u_prime(1,1) + 3);
% multiplication of static function output and output of plant i.e. y
function y_multiplied = fcn(static_fn_output,y)
y_multiplied = static_fn_output * y;
% attack logic. Here attacker knowes plant model(i.e. matrix A,B,C)
function [y_prime,add,power] = fcn(y,clock,attack_vector,add ,power)
y_prime = y;
A = [1.0000 ,0.0001 ,-0.0010;0.5747 ,1.0000 ,-0.0003;0.0162 ,0.0000 ,1.0000];
B = [0.1 ,-0.2; -0.0536 ,0.0332;0.1955 ,-0.5294];
C = [1 ,0 ,0;0 ,1 ,0];
if (clock >= 40) && (clock < 50)
      add = add + ((A^power) * B * attack_vector);
      C_mul_add = C * add;
      y_prime = y - (C_mul_add);
      power = power + 1;
end
```

REFERENCES

RESEARCH PAPER REFERENCES :

- https://www.researchgate.net/post/How_to_run_simulink_model_from_matlab_script_for_every_time_instant_and_retain_the_updated_values_in_simulink_model_after_each_run
- https://www.mathworks.com/matlabcentral/answers/403942-the-block-computed-at-time-is-inf-or-nan
- https://www.researchgate.net/post/I_am_receiving_the_following_error_msg_in_my_simulink_model_what_should_i_do
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6512826/
- https://www.researchgate.net/publication/309242563_CUSUM_and_Chi-squared_Attack_Detection_of_Compromised_Sensors
- https://www.researchgate.net/publication/281567648_The_CuSum_algorithm_-_a_small_review

MATLAB REFERENCES :

- https://www.mathworks.com/products/statistics.html
- https://www.mathworks.com/matlabcentral/answers/104876-matrix-size-mismatch-in-embedded-function-simulink
- https://www.mathworks.com/matlabcentral/answers/95310-what-are-algebraic-loops-in-simulink-and-how-do-i-solve-them
- https://www.mathworks.com/matlabcentral/answers/54603-how-to-solve-algebraic-loop-error
- https://www.mathworks.com/matlabcentral/answers/165862-how-can-i-solve-an-algebraic-loop-error
- https://www.mathworks.com/help/simulink/ug/using-the-sim-command.html
- https://www.mathworks.com/help/simulink/slref/simulink.simulationoutput-class.html
- https://www.youtube.com/watch?v=mCJNuH5PdoU
- https://www.mathworks.com/help/matlab/data_analysis/time-series-objects.html
- https://www.mathworks.com/help/matlab/ref/persistent.html
- https://www.youtube.com/watch?v=TCRLMPIWwIc
- https://www.mathworks.com/help/simulink/slref/unitdelay.html
- https://www.mathworks.com/help/simulink/slref/simulationmanager.html
- https://blogs.mathworks.com/simulink/2017/09/27/new-in-matlab-r2017b-the-simulation-manager/
- https://www.mathworks.com/help/simulink/ug/analyse-results-with-simulation-manager.html