Modul Ekripsi-Dekripsi, Server, dan Penjadwalan Armada pada FMCS

Shah Dehan Lazuardi[1], Arief Syaichu R.[2], Arif Sasongko[3], Agung Darmawan[4]

Program Studi Teknik Elektro-STEI

Institut Teknologi Bandung

Ganeca 10, Bandung, 40135, Jawa Barat, Indonesia

[1]dehanlazuardi@students.itb.ac.id, [2]arief@stei.itb.ac.id, [3]asasongko@gmail.com, [4]agungdar68@yahoo.com

***Abstrak*— Belum adanya jadwal yang tetap masih menjadi masalah utama pada angkutan umum seperti Angkutan kota dan bis. Angkutan kota dan bis beroperasi pada jadwal yang tidak tetap bergantung pada kondisi lalu lintas saat itu. Masalah tersebut dapat diselesaikan menggunakan sistem penjadwalan pada Fleet Monitoring & Controling System untuk Guided Bus. Sistem penjadwalan armada akan menjaga tiap armada agar sesuai jadwalanya masing masing. Sistem penjadwalan ini membutuhkan input berupa posisi tiap armada dan output berupa kecepatan tiap fleet agar sesuai dengan jadwalnya. Data yang dikirimkan dari tiap armada akan di enkripsi terlebih dahulu lalu dikirimkan ke server menggunakan protokol MQTT setelah itu data tersebut didekripsi pada GUI dan di parsing sehingga data posisi dapat diolah pada algrotima penjadwalan. Pada hasil pengujian sistem ini data berhasil dienkripsi lalu dikirmkan pada server setelah itu di dekripsi pada GUI dan algortima penjadwalan dapat mejaga fleet sesuai dengan jadwalanya.**

***Kata kunci* - dekripsi, enkripsi, FMCS, MQTT, penjadwalan.**

I. PENDAHULUAN

Indonesia adalah salah satu negara dengan jumlah penduduk terpadat keempat di dunia, setelah Cina, India, dan Amerika Serikat. Jumlah penduduk Indonesia, 257 juta orang[1], berbanding lurus dengan penggunaan kendaraan pribadi maupun umum. Penggunaan kendaraan pribadi mengalami peningkatan tiap tahunnya. Dari data yang dirilis Badan Pusat Statistik (BPS) Provinsi DKI Jakarta Tahun 2015 terkait dengan statistik transportasi, jumlah kendaraan yang melintas di Provinsi DKI Jakarta terus mengalami peningkatan pada periode 2010-2014, dengan rata-rata peningkatan sebesar 9,93% per tahun. Penyumbang tertinggi peningkatan persentase tersebut adalah kendaraan sepeda motor dengan rata-rata peningkatan per tahun sebesar 10,54%, diikuti dengan peningkatan persentase mobil penumpang yaitu 8,75%. Berbeda jauh dengan pertumbuhan jumlah angkutan umum yang pada tahun 2013-2014 hanya mengalami peningkatan sebesar 1,74%.

Peningkatan jumlah kendaraan tidak di imbangin dengan pelayanan yang diberikan. Pelayanan yang diberikan oleh transportasi umum kurang memuaskan, membuat pengguna transportasi umum beralih pada transportasi pribadi. Angkutan kota, bus dan bajaj memiliki daya tarik yang lemah untuk menggerakan masyarakat agar berpindah moda transportasi. Khususnya bus dan angkutan kota yang menunggu penumpang penuh terkadang memakan waktu yang lama. Tidak jarang bus dan angkutan kota melanggar rambu-rambu lalu lintas seperti berhenti di tempat yang dilarang berhenti. Hal ini diperparah dengan belum baiknya penataan kota dan sistem lalu lintas membuat transportasi umum tidak nyaman untuk digunakan karena tidak memiliki jadwal yang tetap.

Dari masalah masalah tersebut masyarakat dibutuhkan suatu sistem transportasi yang memiliki penjadwalan tetap, sistem tersebut dapat memantau dan memberikan perintah pada armada agar tepat pada jadwalnya sehingga masyarakat dapat menggunakan transportasi umum dengan nyaman tanpa waktunya terbuang sia sia.

II. TEORI PENDUKUNG

Berikut beberapa teori pendukung dalam perancangan dan implementasi subsistem konektor dari sistem FMCS.

A. Message Queuing Telemetry Transport (MQTT)

Protokol MQTT merupakan protokol yang khusus di rancang untuk komunikasi machine to machine atau sederhananya untuk komunikasi dengan device atau mesin yang tidak memiliki alamat khusus, didesain untuk daerah/lokasi dengan resource jaringan yang terbatas.

Protokol ini memiliki kemampuan publish dan subscribe sehingga dapat digunakan untuk komunikasi 2 arah baik antara server ataupun dengan device yang lain. Terdapat 3 jenis QoS Level dalam MQTT.

- QoS level 0
  Pesan akan terkirim tanpa adanya acknowledge atau biasa disebut fire and forget, tidak ada jaminan pesan akan diterima oleh client dan pengirim juga tidak tahu apakah pesan itu diterima atau tidak.
- QoS level 1
  Pesan akan dijamin sampai ke client / penerima paling tidak sekali jika menggunakan QoS level 1, namun jika ada kendala dalam jaringan & timeout, pesan bisa saja terkirim lebih dari satu kali. Pengirim akan tetap menyimpan pesan hingga mendapat respon PUBACK dari penerima, jika pengirim tidak menerima PUBACK dalam waktu tertentu, pesan akan dikirim lagi.
- QoS level 2

QoS tertinggi dalam MQTT adalah level 2, dimana pesan akan dijamin sampai dari pengirim ke penerima hanya sekali. QoS level ini adalah metode teraman dalam mengirimkan pesan namun juga yang paling lambat karena dilakukan pengecekan di pengirim maupun di penerima. Pengirim juga dapat mengetahui apakah pesan telah terkirim atau tidak.

B. Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) merupakan algoritma cryptographic yang dapat digunkan untuk mengamakan data. Algoritma AES adalah blok chipertext simetrik yang dapat mengenkripsi (encipher) dan dekripsi (decipher) informasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut ciphertext; sebaliknya dekripsi adalah merubah ciphertext data menjadi bentuk semula yang kita kenal sebagai plaintext. Algoritma AES is mengunkan kunci kriptografi 128, 192, dan 256 bits untuk mengenkrip dan dekrip data pada blok 128 bits.

III. PERANCANGAN DAN IMPLEMENTASI

Dalam FMCS penulis mendesain dan implementasikan tiga bagian yaitu keamanan informasi, implementasi server, dan algortima penjadwalan. Adapun spesifikasi dari ketiga bagian tersebut adalah

- Spesifikasi server
  a. dapat meneruskan pesan yang diterima.
  b. Latency server kurang dari 4 ms.
- Spesifikasi algoritma penjadwalan
  a. Algortima penjadwalan dapat menjaga armada sesuai pada jadwalanya
- Spesifikasi modul enkripsi-dekripsi
  a. Data pesan antara armada dan *control station* hanya dapat dibaca oleh armada dan control station saja.

A. Server

Server terletak pada *Control Station*. GUI akan mengakses server menggunakan jaringan lokal sedangkan *fleet hardware* akan mengakses server melalui jaringan internet menggunakan suatu protocol tertentu.

1. Perancangan Komputer Server

Asumsi yang dipakai dalam perancangan server ini adalah terdapat 40 armada yang mengakses server tiap 0.7 detik sekali. Menurut hasil *benchmark* Scalagent[1] menggunakan komputer dengan spesifikasi:

- Intel Core 2 Duo CPU E8400 3.00GHz
- 4 GB RAM
- Gigabit switch

CPU usage saat menerima 2000 *message*/*second* adalah 4% dengan latency 4 ms. Maka spesifikasi komputer

- Intel i5-4300M 2.5 Ghz
- 8 GB RAM
- FS1104 fiberhome router

Dapat menanggung beban server sesuai asumsi diatas.

2. Perancangan Protokol Komunikasi Server

Dalam perancangan protokol komunikasi server digunakan asumsi:

- jaringan yang digunakan armada tidak stabil
- kemampuan komputasi *fleet hardware* terbatas

Jenis	**MQTT**	**HTTP**
Transport	TCP	TCP
Messaging	Publish/Subscribe	Request/Response
Protocol	28 bytes	491 bytes
QoS	Ada QoS	Tidak ada Qos
tipe	Asychronous	Synchronous
Resources Utilization	Rendah	Tinggi
Latency	Millisecond [1]	second[4]

tabel 3.1 table perbandingan protocol MQTT dan HTTP

Sehingga dipilih protokol MQTT sebagai protokol komunikasi dengan server.

3. Implementasi Server

Terdapat beberapa langkah dalam meng-*install* server MQTT yaitu:

- unduh *installer* Mosquitto dan *install* pada komputer server.
- Setelah itu *install* OpenSSL dan salin libeay32.dll dan ssleay32.dll dari direktori openSSL ke direktori mosquitto.
- Salin pthreadVC2.dll ke directory mosquito, pada tahap ini server telah selesai terpasang.

Untuk menjalankan server klik icon mosquito pada direktori tempat server di *install.*

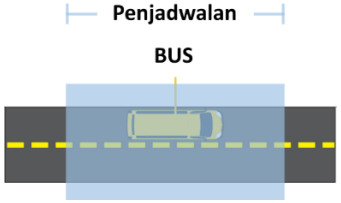
Setelah server terpasang lalukan *port forwarding* pada router sesuai dengan port yang digunakan untuk server MQTT sehingga server dapat diakses melalui jaringan internet.

B. Penjadwalan

Algoritma penjadwalan berfungsi untuk mengatur penjadwalan armada. Algoritma ini bertujuan agar persebaran armada merata pada jalur trayeknya sehingga penumpang tidak perlu menunggu armada terlalu lama.

1. Perancangan Penjadwalan

Algoritma penjadwalan akan menetukan posisi pejadwalan armada tiap waktu. Penjadwalan berupa rentang posisi apabila armada terletak pada rentang tersebut maka armada telah sesuai dengan jadwalnya.



Gambar 3.1 penjadwalan pada armada

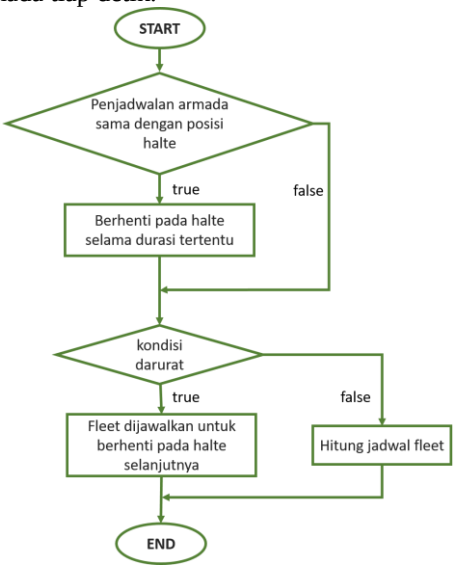
Apabila terdapat armada yang tidak sesuai dengan posisi yang ditentukan maka algoritma ini akan menghitung perbedaan armada tersebut dan mengirimkan perintah berupa kecepatan yang harus diikuti oleh pengemudi armada untuk menyesuaikan dengan posisi pejadwalan.

Apabila terdapat kondisi darurat pada salah satu armada, algoritma ini akan memberhentikan penjadwalan armada pada halte terdekat didepan armada. Ketika keadaan darurat tersebut sudah diatasai maka penjadwalan akan menghitung ulang jarak penjadwalan tiap armada agar armada tersebar merata kembali.

Adapun rancangan algortima penjadwalan adalah sebagai berikut.

- Prosedur penjadwalan

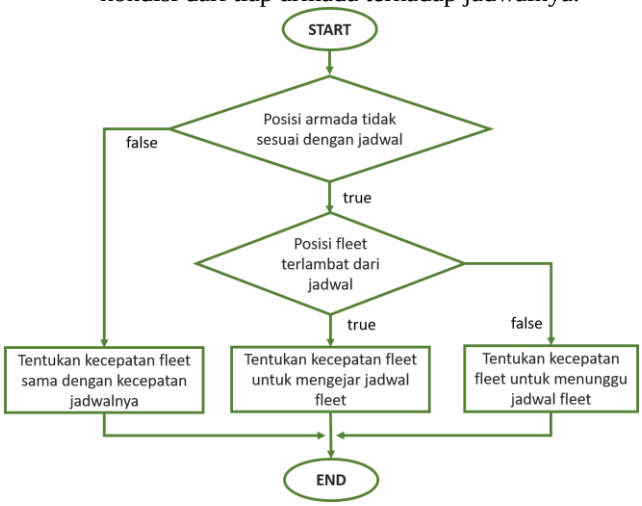
  Prosedur ini akan mengatur jadwal dari armada tiap detik.



Gambar3.2 flowchart prosedur penjadwalan

- Prosedur cek penjadwalan

  Prosedur ini berfungsi untuk memeriksa kondisi dari tiap armada terhadap jadwalnya.



Gambar 3.3 flowchart prosedur cek jadwal

- Prosedur kondisi darurat

  Prosedur ini dijalankan jika armada mengalami kondisi darurat dan pengemudi armada menekan tombol darurat pada *fleet hardware*. Prosedur ini berfungsi untuk menentukan halte terdekat sehingga dalam keadaan darurat armada berhenti pada halte tersebut

- Prosedur hitung penjadwalan

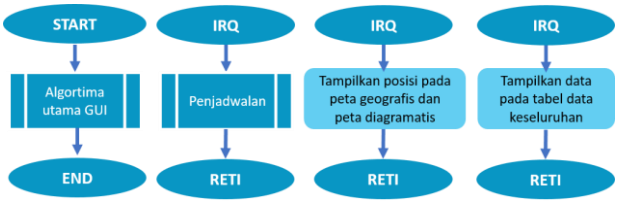
  Prosedur ini dipanggil setelah situasi darurat dapat diatasi. Prosedur ini menentukan posisi array penjadwalan yang baru sesuai dengan jumlah armada yang masih dapat beroperasi.

- Prosedur cek sch2

  Sch2 berfungsi ketika kondisi darurat terjadi. Variable sch2 berisi posisi countersch ideal. Prosedur ini akan membuat counter sch agar sama dengan sch2 sehingga penjadwalan kembali normal setelah terjadi kondisi darurat.

2. Implementasi Penjadwalan

Algoritma utama GUI memiliki susunan seperti flowchart di bawah ini.



Gambar 3.4 algoritam utama GUI

Adapun rancangan algortima penjadwalan adalah sebagai berikut.

- Prosedur penjadwalan

```
If posisi penjadwalan = posisi halte
        counter durasi halte++
        If counter durasi halte > durasi berhenti
                posisi penjadwalan ++
Else
        If kondisi darurat = true
                posisi penjadwalan ++ sampai pada halte terdekat di depan armada
        else
                posisi penjadwalan ++
```

Prosedur ini dipanggil melalui *interrupt* tiap 200ms sehingga posisi penjadwalan bergerak dengan kecepatan 54km/s

- Prosedur cek jadwal

```
If posisi armada != posisi penjadwalan
        If posisi penjadwalan > posisi armada
                If |posisi penjadwalan - posisi armada| > 30 satuan array
                        fleet speed = 86.4 km/h
                else
                        fleet speed = 64.8 km/h
        else
```

```
                If |posisi penjadwalan - posisi armada| > 30 satuan array
                        fleet speed = 21.6 km/h
                else
                        fleet speed = 43.2 km/h
Else
        fleet speed = 54 km/h
```

Prosedur ini terdapat pada algoritma utama GUI. Prosedur ini dipanggil ketika pesan dari armada selasai di *parsing.*

- Prosedur kondisi darurat

```
If kondisi darurat = true
        for i=0 until i<jumlah halte
                if posisi armada < halte[i]
                        halte terdekat didepan armada = halte[i]
```

- Prosedur hitung penjadwalan

```
for i=1 until i<fleetsch2.length
        fleetsch2[i] += fleethsch2[i-1] + jarak
```

- Prosedur cek sch2

```
If sch2 != posisi penjadwalan
        If sch2 > posisi penjadwalan
                posisi penjadwalan++
        else
                posisi penjadwalan--
```

C. *Modul Enkripsi-Dekripsi*

Modul enkripsi dan dekripsi bertujuan untuk mengamankan data saat data di-*publish* melalui internet adapun rancangan dan implementasi modul enkripsi-dekripsi adalah sebagai berikut

1. *Perancangan Modul Enkripsi-Dekripsi*

   Terdapat tiga algortima enkripsi yang memiliki kelebihan dan kekurangan[2] seperti tabel dibawah ini.

NAMA	*Avalance effect*	*throughput*
AES	93%	950 bits/sec
DES	75%	500 bits/sec
RSA	60%	100 bits/sec

Tabe 3.2 table avalance effect dan throughput AES, DES, RSA

Avalance effect merupakan perubahan hasil enkripsi yang terjadi ketika sebagian dari pesan diubah. Semakin besar nilai ini maka semakin baik

Throughput adalah jumlah data yang diproses tiap detiknya. Semakin besar *throughput* maka semakin cepat algoritma melakukan enkripsi.

Dari tabel diatas maka enkripsi dan dekripsi yang dilakukan pada sistem FMCS ini menggunakan AES.

2. *Implementasi Modul Enkripsi-Dekripsi*

   Implementasi AES dilakukan pada *fleet hardware* dan juga GUI menggunakan AES-128 bit. Pada fleet hardware digunakan library AES.h. Enkripsi dilakukan dengan menggunakan prosedur dibawah ini.

```
do_aes_encrypt(plain, length, ciphertext, key, bits, iv);
```

Adapun penjelasan dari parameter prosedur tersebut adalah sebagai berikut

- *plaintext*, variable yang berisi pesan yang akan di enkripsi.
- *length*, merupakan panjang pesan.
- *ciphertext* adalah variable hasil enkripsi, isi dari variable ini akan dikirim ke server.
- *key* merupakan *string* tertentu digunakan sebagai "kunci" untuk melakukan enkripsi dan dekripsi dari suatu pesan.
- *bits* adalah jumlah bit dari key, pada implementasi ini digunakan 128 bit key.
- *iv* merupakan *initial vector* yang digunakan dalam proses enkripsi.

Setelah pesan dienkripsi pada fleet hardware. Pesan dikirimkan ke server lalu server meneruskan pesan tersebut pada GUI. Dekripsi pesan pada GUI menggunakan fungsi sebagai berikut.

```
DecryptStringFromBytes_Aes(ciphertext, key, iv);
```

Fungsi tersebut akan menghasilkan *string* berisi pesan yang sudah terdekripsi. Untuk mendekripsi pesan harus menggunakan variable key dan iv yang sama yang digunakan pada *fleet hardware*. Pesan tersebut akan diproses oleh GUI untuk menentukan *command* yang akan dikirim pada *fleet hardware*. Command tersebut harus melalui proses enkripsi pada GUI dengan menggunakan fungsi sebagai berikut.

```
EncryptStringToBytes_Aes(plaintext, key, iv);
```

Fungsi tersebut akan mengembalikan *variable* byte berisi pesan yang sudah terenkripsi. Pesan tersebut harus didekripsi pada fleet hardware menggunakan prosedur sebagai berikut.

```
do_aes_decrypt(ciphertext, length, message, key, bits, iv);
```

Prosedur tersebut akan menyimpan pesan hasil dekripsi pada variable message.

IV. HASIL PENGUJIAN

A. *Server*

Pengujian server dilakukan untuk mengetahui kehandalan server dalam menerima banyak data / koneksi sekaligus. Asumsi yang digunakan dalam pengujian server adalah latency yang ditimbulkan murni dari server.

1. *Prosedur Pengujian Server*

   Adapun prosedur pengujian server adalah:

   - Menjalankan server dan hubungkan server dengan jaringan lokal.
   - Menjalankan aplikasi dummy data dan hubungkan aplikasi dengan server pada jaringan

lokal. Aplikasi ini mem-publish 40 pesan tiap 0.7 detik pada server.

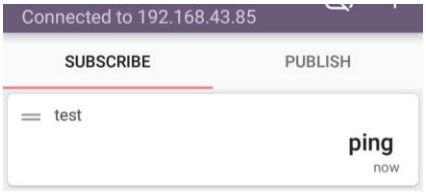
- Hitung latency server dengan cara menghitung ketelambatan pesan yang diterima.

2. *Hasil Pengujian Server*

Pengujian server untuk membuktikan bahwa server dapat meneruskan pesan yang diteruma dilakukan dengan mengirimkan suati pesan pada topik tertentu seperti gambar di bawah ini

Gambar 4.1 tampilan saat *publish message*

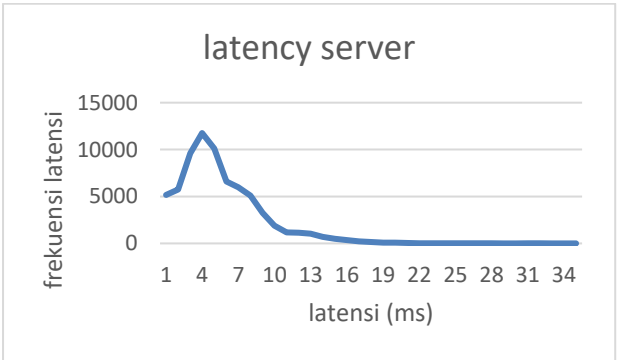
Pesan yang dikirimkan tulisan "ping" pada topik "test". Setelah itu diamati apakah server dapat meneruskan pesan pada topik tersebut dengan cara *subscribe* pada topik yang sama sebelumnya. Dapat dilihat pada gambar di bawah ini



Gambar 4.2 tampilan saat pesan diterima

Pesan "ping" dapat diterima pada topik "test". Server berhasil meneruskan pesan yang dikirim pada topiknya.

Pengujian selanjutnya dilakukan selama 20 menit untuk melihat latensi server. Terdapat 70 ribu data latensi yang didapat seperti grafik dibawah ini



Grafik 4.1 grafik *latency* server

Didapatkan rata rata latensi 4.48 detik.

B. Penjadwalan

Pengujian ini menguji algortima yang sebelumnya telah dirancang dengan kasus kasus tertentu. Setelah itu diamati respon dari algoritma tersebut apakah algoritma dapat memperbaiki posisi armada sehingga sesuai dengan jadwalnnya. Kasus pertama adalah ketika armada terlambat dari penjadwalan, kasus kedua adalah ketika armada mendhului penjadwalan, dan kasus terakhir adalah ketika armada sesuai dengan penjadwalan.

1. *Prosedur Pengujian Penjadwalan*

Pengujian ini dilakukan sebanyak 3 kali sesuai dengan kasus yang di uji. Adapun prosedur pengujian ini adalah:

- Menjalankan server dan hubungkan pada jaringan internet.
- Menjalankan GUI dan hubungkan pada server melalui jaringan lokal.
- Atur posisi fleet agar sesuai dengan kasus yang akan di uji pada program dummy data.
- Menjalankan aplikasi Dummy Data dan hubungkan pada server melalui internet.
- Merekam hasil screenshot dan perbedaan waktu yang dibutuhkan fleet untuk sesuai dengan penjadwalan.

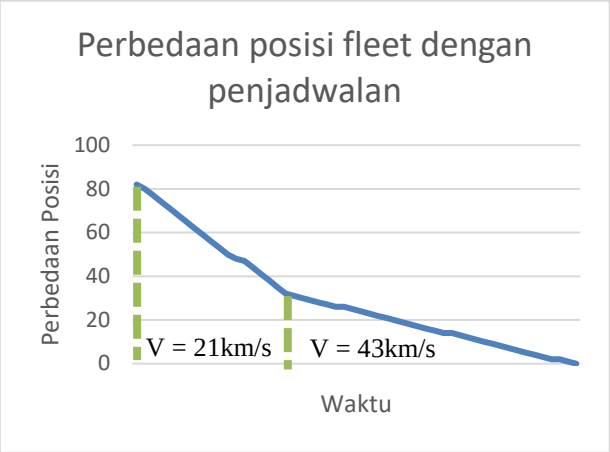
2. *Hasil Pengujian Penjadwalan*

Adapun hasil pengujian dari kasus pertama yaitu armada terlambat dari jadwalnya adalah sebagai berikut

Grafik 4.2 grafik perbedaan posisi fleet tertingga dari penjadwalan

dapat diamati dari grafik di atas ketika armada terlambat dari jadwalnya algrotima penjadwalan dapat memperbaiki posisi armada agar sesuai dengan jadwalnya.

Pada pengujian kasus kedua yaitu armada mendahului jadwalanya didapat hasil seperti grafik di bawah ini



Grafik 4.3 grafik perbedaan posisi fleet kasus mendahului penjadwalan

dapat diamati dari grafik di atas ketika armada mendahuli jadwalnya algrotima penjadwalan dapat memperbaiki posisi armada agar sesuai dengan jadwalnya.

Pada pengujian kasus ketiga yaitu armada sesuai dengan jadwalanya didapat hasil seperti gambar di bawah.

Grafik 4.4 grafik perbedaan posisi fleet sama dengan penjadwalan

dapat diamati dari grafik di atas ketika armada sesuai dengan jadwalnya algrotima penjadwalan mempertahankan kecepatan armada sesuai dengan kecepatan penjadwalan.

C. *Modul Eknripsi-Dekripsi*

Pengujian modul enkripsi dan dekripsi dilakukan untuk mengetahui apakah fleet hardware dan control station (GUI) saja yang dapat membaca terenkripsi pesan tersebut.

3. Prosedur Pengujian Modul Enkripsi-Dekripsi

Pengujian ini dilakukan dengan langkah langkah sebagai berikut:

- Menjalankan server dan hubungkan pada jaringan internet.
- Menjalankan GUI dan hubungkan pada server melalui jaringan lokal.
- Menyalakan fleet hardware.
- Mengamati hasil data yang diterima oleh GUI dan client.

Dalam pengujian ini fleet hardware melukan enkripsi dan GUI melakukan dekripsi.

4. Hasil Pengujian Modul Enkripsi-Dekripsi

Hasil pengujian menunjukkan pesan yang didekripsi GUI sama dengan pesan sebelum dienkripsi oleh fleet hardware jika menggunakan key dan iv yang sama. Dari hasil pengujian juga menunjukan ketika menggunakan iv atau key yang berbeda maka pesan hasil dekripsi tidak sesuai dengan pesan sebelum dienkripsi.

V. KESIMPULAN

Server MQTT telah berhasil diimplementasikan dengan baik. Server mampu meneruskan pesan dengan latensi 4.81 *milisecond.* Server MQTT ini dapat diakses melalui internet.

Algoritma penjadwalan dapat menjaga armada pada jadwalnya. Apabila armada tidak sesuai dengan penjadwalan maka algoritma akan memberikan perintah berupa kecepatan yang harus diikuti armada agar armada sesuai dengan jadwalanya.

Modul enkripsi dan dekripsi telah di implemetasikan pada *fleet hardware* dan juga GUI menggunakan algortima AES. Fleet hardware serta GUI dapat membaca pesan terenkripsi yang dikirimkan satu sama lain menggunakan "key" tertentu yang sama.

REFERENSI

[1] Razali Ritonga, 2014, Kebutuhan Data Ketenagakerjaan Untuk Pembangunan Berkelanjutan, BPJS.

[2] scalagent, Benchmark of MQTT servers, January 2015

[3] S.N.Ghosh. (2015, Sep.). Performance Analysis of AES, DES, RSA And AES-DES-RSA Hybrid Algorithm for Data Security. *Volume 2, Issue 5*. [Online]. 85, Available: https://www.academia.edu/15749187/Performance_Analysis_of_AES_DES_RSA_And_AES-DES-RSA_Hybrid_Algorithm_for_Data_Security

[4] Isode, M-Link & XMPP Performance Measurements over Satcom and Constrained IP Networks. Available: http://www.isode.com/whitepapers/xmpp-performance-constrained.html.