TUGAS AKHIR - TE 141599

PERENCANAAN SISTEM KEAMANAN PADA JARINGAN KOMUNIKASI ITS *(INTELLIGENT TRANSPORT SYSTEM)* ANTARA OBU DAN TMC SERVER

DIANA MUSABBIHAH
NRP 07111440007004

Dosen Pembimbing
Dr. Ir. Achmad Affandi, DEA.
Ir. Djoko Suprajitno Rahardjo, MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT - TE 141599

DEVELOPMENT OF SECURITY SYSTEM FOR WIRELESS COMMUNICATION OF ITS (INTELLIGENT TRANSPORT SYSTEM) BETWEEN OBU AND TMC SERVER

DIANA MUSABBIHAH
NRP 07111440007004

Lecture Advisor
Dr. Ir. Achmad Affandi, DEA.
Ir. Djoko Suprajitno Rahardjo, MT.

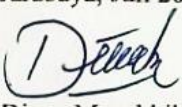
DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty Of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan tugas akhir saya dengan judul "*Perencanaan Sistem Keamanan pada Jaringan Komunikasi ITS (Intelligent Transport System) Antara OBU dan TMC Server"* adalah benar – benar hasil karya yang dikerjakan secara mandiri, diselesaikan tanpa menggunakan bahan – bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.
Semua referensi yang dikutip maupun diambil telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Juli 2018



Diana Musabbihah
07111440007004

LEMBAR PENGESAHAN

PERENCANAAN SISTEM KEAMANAN PADA JARINGAN ITS (*INTELLIGENT TRANSPORT SYSTEM*) ANTARA OBU DAN TMC SERVER

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Telekomunikasi Multimedia
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Achmad Affandi, DEA	**Ir. Djoko Suprajitno R., MT.**
NIP. 196510141990021001	**NIP. 195506221987011001**

SURABAYA
16 JULI 2018

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI ELEKTRO DEPARTEMEN TEKNIK ELEKTRO

ABSTRAK

Intelligent Transport System (ITS) merupakan sebuah sistem transportasi cerdas yang dirancang dengan harapan untuk mengoptimalkan kondisi angkutan umum yang saat ini mulai diintegrasikan melalui jaringan TIK. Salah satu bentuk optimalisasinya adalah dengan memonitoring kondisi armada yang bertugas dan segala transaksi yang berlangsung secara realtime oleh pengelola di ruang kontrol. Untuk dapat melakukan hal itu, maka OBU (On Board Unit) yang berada di dalam armada harus selalu mengirimkan data-data yang dimilikinya, seperti data lokasi dan transaksi ticketing, kepada TMC server (Traffic Management Center) setiap saat.

Namun, setiap informasi yang melewati jaringan internet sangatlah rawan terhadap kejahatan cyber yang bermacam-macam yang bisa dilakukan oleh siapapun. Misalnya saja eavesdropping, attacker yang mengetahui lokasi armada tertentu dapat menguba-ubah lokasi armada sehingga mengacaukan jadwal keberangkatan, atau dengan mengetahui nomor serial dari kartu RFID seseorang, attacker bisa saja menggunakannya untuk keperluan pribadi yang tidak semestinya. Kejahatan-kejahatan tersebut tentu sangatlah merugikan bagi pihak-pihak terkait. Oleh karena itu, dibuatlah suatu sistem keamanan yang dapat melindungi data atau informasi yang dikirimkan dari OBU menuju TMC server.

Pada pengerjaan tugas akhir ini dilakukan pengamanan dengan cara mengenkripsi payload atau data yang dikirim dengan metode AES-128-CBC serta HMAC SHA256 untuk tandanya. Proses enkripsi dilakukan dengan menggunakan modul Fernet yang berada di dalam package cryptography. Setelah program enkripsi berhasil dijalankan, dilakukan juga pengujian dengan beberapa skenario yang ditentukan, sehingga didapatkan rata-rata delay terbesar adalah 0.00858 s, throughput 19063,667 bps yang semuanya didapatkan karena proses enkripsi dan dekripsi yang terjadi. Dan untuk tingkat keamanan terbaik dimiliki oleh pengiriman dengan enkripsi dan diikuti proses dekripsi di sisi subscriber seingga informasi yang diterima aman dan dapat dibaca.

Kata kunci : *Intelligent Transport System*, Protokol MQTT, AES-128, SHA-256

Halaman ini mengandung kekosongan

ABSTRACT

Intelligent Transport System (ITS) is a system designed with the aim to optimize the condition of public transport that is now beginning to be integrated through ICT networks. One form of optimization is to monitor the condition of the fleet in charge and all transactions that take place in real time by the manager in the control room. In order to do so, the OBU (On Board Unit) within the fleet must always transmit its data, such as location data and ticketing transactions, to the TMC server (Traffic Management Center) at any time.

However, any information that passes through the Internet network is very vulnerable to the various cybercrimes that can be done by anyone. For example, eavesdropping, an attacker who knows the location of a particular fleet can change the location of the fleet to disrupt the departure schedule, or by knowing the serial number of a one's RFID card, the attacker may be using it for undue personal purposes. Those crimes are certainly very detrimental to related parties. Therefore, a security system is created that can protect the data or information that is sent from the OBU to the TMC server.

In this final project is done by encrypting security payload or data sent by method of AES-128-CBC and HMAC SHA256 for the mark. The encrypted data is data to be sent by the publisher to the subscriber. Encryption process is done by using Fernet module which is in package cryptography. After the encryption program successfully executed, also tested to see the resulting delay, so the smallest delay is 0.00565 s and the biggest delay is 0.05799 s. In accordance with the ITU-T G.114 recommendation, the delay value of less than 150 ms belongs to the acceptable category so that the chosen security method is still applicable.

Keywords : AES-128, cybercrimes, Intelligent Transport System, OBU, SHA-256, TMC server

Halaman ini mengandung kekosongan

KATA PENGANTAR

Pertama-tama, saya ingin mengucapkan puja dan puji syukur kepada Allah SWT serta Nabi Muhammad SAW yang telah memberikan saya kesempatan dan bantuan untuk mengerjakan serta merampungkan buku tugas akhir ini yang memiliki judul:

PERENCANAAN SISTEM KEAMANAN PADA JARINGAN ITS (*INTELLIGENT TRANSPORT SYSTEM*) ANTARA OBU DAN TMC SERVER

yang mana tugas akhir ini disusun sebagai salah satu syarat dalam menyelesaikan studi pada bidang studi Telekomunikasi Multimedia yang ada di jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam kesempatan ini, saya selaku penulis ingin menyampaikan rasa terima kasih yang sangat mendalam kepada semua pihak yang telah mendukung selama proses penyelesaian tugas akhir ini, khususnya kepada :

1. Dr. Ir. Achmad Affandi, DEA., dan Ir. Djoko Suprajitno R., M.T., Selaku dosen pembimbing, yang telah memberikan bimbingan dan masukan dalam proses pengerjaan tugas akhir ini.
2. Bapak Michael Ardita dan Mas Hadi Purnama yang telah banyak memberikan bantuan dalam proses penyelesaian tugas akhir ini.
3. Seluruh anggota keluarga di rumah yang senantiasa memberikan dukungan moral jarak jauh agar tetap semangat dalam mengerjakan tugas akhir ini.
4. Bapak dan Ibu dosen jurusan teknik elektro ITS, khususnya bidang studi Telekomunikasi Multimedia, atas segala ilmu yang telah diberikan selama saya kuliah di ITS.
5. Semua teman-teman E-54, khususnya bidang Telekomunikasi Multimedia, terutama lab Jaringan Telekomunikasi yang sama-sama mengerjakan proyek Tugas Akhir ini.
6. Teman-teman D14, terutama 7 Bidadari yang berjuang bersama-sama dan menjadi keluarga baru sejak masuk di kampus ITS hingga akhir ini.

Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca dan mampu memberikan kontribusi terhadap perkembangan keilmuan, khususnya dalam bidang telekomunikasi. Meskipun saya sadari bahwa dalam penyusunan buku ini masih jauh dari kata sempurna sehingga saran, kritik, dan diskusi untuk pengembangan tugas akhir ini sangat diharapkan.

Surabaya, 4 Juli 2018

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang, perumusan masalah yang dihadapi, batasan permasalahan, tujuan, metodologi penulisan, sistematika penulisan, dan relevansi.

1.1 Latar Belakang

Untuk menangani kemacetan yang terjadi di kota-kota besar, pemerintah saat ini sangat gencar dalam membangun fasilitas transportasi publik yang memadai. Pada era digital saat ini, sistem transportasi modern yang dikenal dengan sistem transportasi cerdas atau Intelligent Transportation System (ITS) sangat membutuhkan penerapan teknologi informasi dan telekomunikasi, di antaranya adalah dalam sistem komunikasi data yang digunakan pada jaringan ITS. Setiap data yang digunakan dalam transaksi pada ITS seperti data diri penumpang, lokasi armada, status armada hingga data saldo dari penumpang yang dikirim dari sisi OBU ke server TMC harus melewati jaringan internet untuk kemudian diolah.

Salah satu aspek penting yang harus diperhatikan dalam sistem komunikasi data ini adalah dari segi keamanannya. Demi efektivitas infrastruktur jaringan, pada sistem ITS digunakan jaringan internet publik untuk transportasi data yang dikirim dari OBU ke TMC server. Dengan internet publik yang bisa diakses oleh siapapun, setiap orang yang terhubung dengannya akan dapat mengakses data-data dan trafik yang melintas termasuk data dari transaksi di dalam ITS.

Oleh karena itu diperlukan suatu sistem keamanan yang dapat menjaga integritas data yang dikirimkan dari sisi OBU menuju TMC server agar tidak ada pihak yang dapat melihat ataupun menginterferensi data yang melintas dalam jaringan internet. Aspek lain dalam network security yang perlu diperhatikan pada jaringan ITS ini adalah availability, yakni ketersediaan informasi dimana data atau infomasi yang dikirim oleh OBU dapat tersampaikan ke TMC server. Maka dari itu, pada tugas akhir ini akan dilakukan perancangan protokol keamanan pada jaringan ITS yang akan

menjamin integritas data yang dikirim dari sisi client melalui internet publik menuju server dan sebaliknya, serta dapat memastikan bahwa data yang terkirim benar-benar telah sampai di TMC server sehingga data dapat dikelola dengan aman.

1.2 Rumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimanakah sistem keamanan yang tepat untuk digunakan dalam jaringan ITS yang berjalan dengan protokol MQTT.
2. Bagaimana cara menjaga agar keamanan informasi antara OBU dan TMC server tetap terjaga apabila ada pihak yang mengawasi jalur komunikasi ITS.
3. Bagaimana efek diterapkannya sistem keamanan dalam protokol MQTT yang dibangun dengan keandalan jaringan ITS sendiri.

1.3 Batasan Masalah

Batasan masalah dari Tugas Akhir ini adalah sebagai berikut.

1. Menggunakan protokol MQTT
2. Digunakan pada jaringan komunikasi antara OBU dan server TMC

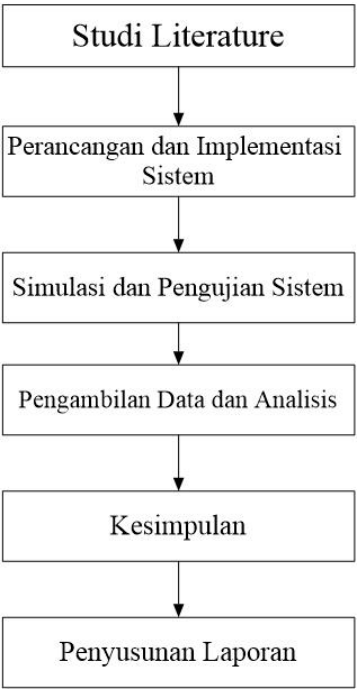
1.4 Tujuan Tugas Akhir

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Terdapat sebuah sistem keamanan jaringan yang dapat menjamin keamanan data transmisi antara sisi OBU dan TMC server
2. Menjamin kerahasiaan dan integritas data yang dikirim pada jaringan ITS ole OBU maupun TMC server.
3. Mengetahui efek dari diterapkannya sistem keamanan tersebut terhadap keandalan jaringan ITS.

1.5 Metodologi

Metodologi yang digunakan pada pengerjaan tugas akhir ini adalah sebagai berikut.



Gambar 1.1 Metodologi Penelitian

Dari gambar metodologi penelitian pada Gambar 1.1, dapat dilihat tahapan-tahapan yang akan dilalui dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi literatur

   Mengumpulkan dan memelajari buku serta referensi yang dapat membantu mewujudkan tujuan dari tugas akhir ini antara lain, *Intelligent Transportation System*, *Network Security*, Protokol Telekomunikasi, Keamanan pada sistem

RFID, kebijakan keamanan, kriptografi, Enkripsi data, serta beberapa bahasa pemrograman yang diperlukan seperti Delphi, PHP, Python, dll.

2. Perancangan dan Implementasi Sistem Keamanan Jaringan ITS

   Setelah mengetahui dan mendapatkan semua bahan-bahan yang diperlukan dalam tugas akhir ini, kemudian dibuatlah suatu rancangan sistem keamanan yang kemudian diimplementasikan pada jaringan ITS.

3. Simulasi dan Pengujian Sistem

   Apabila sistem sudah berhasil diimplementasikan, maka selanjutnya adalah melakukan simulasi dan pengujian sistem dengan melakukan penetrasi untuk melihat ketahanan dari sistem keamanan yang telah dirancang.

4. Pengambilan Data dan Analisis

   Data diperoleh dari hasil simulasi dan pengujian sistem keamanan terhadap jaringan ITS. Kemudian dilakukan analisis untuk mengetahui tingkat performansi keamanan dari jaringan. Berikut merupakan parameter yang harus

5. Kesimpulan

   Kesimpulan dibuat berdasarkan data - data yang telah diambil saat pengujian sistem dan sudah dianalisis dengan matang.

1.6 Sistematika Penulisan

Sistematika Penulisan tugas akhir ini adalah sebagai berikut :

BAB I PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang, perumusan masalah yang dihadapi dalam sistem, batasan permasalahan yang akan digunakan, tujuan dari pembuatan tugas akhir, metodologi penulisan, sistematika penulisan buku, dan relevansi dari tugas akhir ini.

BAB II DASAR TEORI

Berisi teori-teori dasar yang berhubungan dengan pembuatan dan penyusunan tugas akhir. Dasar teori ini terdiri dari beberapa bagian yakni Intelligent Transport System (ITS), dasar-dasar keamanan jaringan, keamanan pada sistem RFID sebagai metode pembayaran dan ticketing, metode dan algoritma kriptografi, protokol MQTT, keamana pada protokol MQTT, Python, dan bahasa pemrograman lainnya.

BAB III PERANCANGAN DAN IMPLEMENTASI SISTEM

Di sini dibahas mengenai desain rekayasa perangkat lunak pembuatan sistem keamanan ITS dengan metode enkripsi AES-128-CBC serta implementasinya untuk pengamanan antara OBU dan server TMC.

BAB IV PENGUJIAN DAN ANALISIS DATA

Merupakan pengujian perangkat lunak terhadap kinerja sistem keamanan yang dibangun serta pengaruh proses enkripsi dan dekripsi data terhadap pengiriman data tersebut dalam jaringan.

BAB V PENUTUP

Pada bab ini berisi kesimpulan dari keseluruhan materi dan hasil analisis data pada bab IV. Selain itu juga dibahas mengenai saran yang bisa dilakukan untuk pengembangan penelitian selanjutnya.

1.7 Relevansi

Hasil Dari tugas akhir ini diharapkan dapat diimplementasikan pada jaringan Komunikasi ITS untuk memberikan keamanan paket data yang dikirim dari sisi OBU kepada sisi TMC server sehingga transaksi dapat diolah dengan aman dan lancar.

Halaman ini mengandung kekosongan

BAB 2 DASAR TEORI

Pada bagian ini akan dijelaskan mengenai istilah-istilah serta beberapa materi dasar mengenai hal-hal yang dilakukan dan digunakan dalam pengerjaan tugas akhir ini.

2.1 Kebijakan Keamanan

Banyak sekali bidang-bidang yang dapat dibicarakan apabila membahas tentang kebijakan keamanan. Keamanan dapat digolongkan dalam beberapa bidang antara lain keamanan informasi, keamanan komputer, keamanan jaringan, keamanan multimedia, dan masih banyak lagi. Hal yang dapat diunggulkan dalam bidang keamanan ketika banyak pihak menawarkan sebuah sistem keamanan adalah dari segi kekuatan, seberapa kuat dan seberapa lama sistem keamanan tersebut dalam menahan serangan. Terdapat tiga aspek utama yang tidak bisa diabaikan dalam bidang security adalah [1] :

1. *Confidentiality*, memastikan bahwa tidak ada yang mengetahui isi pesan selain pengirim dan penerima.
2. *Integrity*, dapat membuktikan keaslian dari informasi, dimana pesan yang diterima merupakan pesan yang dikirim.
3. *Availability*, yakni kemampuan sistem dalam melakukan layanan setiap kali akan diakses oleh pengguna.

Selain tiga hal di atas, seringkali ditambahkan dua aspek lainnya yang juga dianggap penting dalam lingkup keamanan, yakni :

1. *Non-repudiation,* kondisi di mana pihak-pihak yang melakukan transaksi seperti pengirim dan penerima pesan tidak bisa mengingkari perbuatannya.
2. *Authentication,* validasi identitas dari pihak pengirim dan penerima sehingga kedua belah pihak memang merupakan pihak yang diinginkan.

Dalam layanan keamanan, masih banyak hal lagi aspek-aspek yang jarang diperhatikan namun akan menjadi penting tergantung dengan permintaan layanan sistem seperti obliviousness (keraguan), information flow, authorization, *verifiability*, *unforgetability*, *distinguishability*, dan *detectability*.

Dalam implementasinya, layanan keamanan dipilih berdasarkan kebutuhan dan layanan yang ditawarkan oleh sistem. Apakah akan menitikberatkan di segi integritas, availabilitas, privasi, atau yang lainnya.

2.2 Intelligent Transport System

Intelligent Transport System (ITS) atau dalam bahasa Indonesia disebut dengan Sistem Transportasi cerdas, merupakan suatu sistem transportasi hasil dari pemanfaatan teknologi informasi dan telekomunikasi yang dapat mengoptimalkan semua model transportasi dengan cara meningkatkan efektivitas biaya dan cara kerjanya [2].

Terdapat tiga bidang dalam sistem transportasi cerdas, yakni:

1. Intelligent Infrastructure (Infrastruktur Cerdas)
2. *Smart Vehicle* (Kendaraan Cerdas)
3. *Information Services* (Layanan Informasi)

Pada level praktikal, sistem ITS dibagi menjadi enam kategori, yakni [3] :

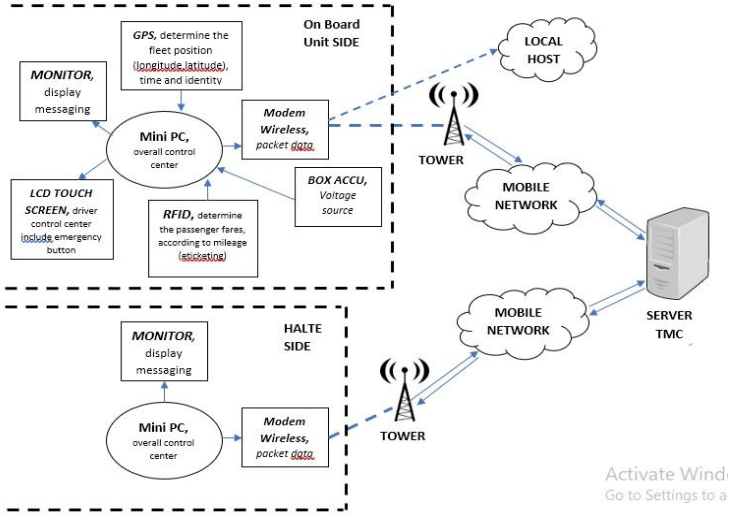
1. Advance Traffic Management Systems (AMTS)
2. Advance Vehicle Control Systems (AVCS)
3. Advance Traveler Information Systems (ATIS)
4. Advance Public Transportation Systems (APTS)
5. Commercial Vehicle Operations (CVO)
6. Advance Rural Transportation System (ARTS)

2.3 Infrastruktur Jaringan ITS

2.3.1 OBU dan VMS

Dalam sistem transportasi cerdas, terdapat dua komponen utama yang harus tersedia yakni OBU (On-board Unit) dan VMS (Variable Messaging Unit). OBU merupakan suatu komponen yang

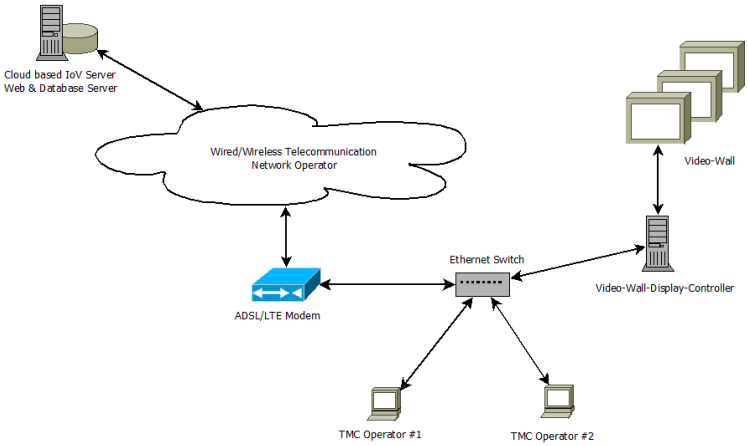
diletakkan dalam armada, berfungsi sebagai sistem pengendali masukan dan keluaran yang terkait dengan fungsional manajemen armada, pendapatan (tiket), lalu lintas, dan sistem darurat. Sedangkan VMS bertugas untuk memberikan informasi kepada pengendara dan (calon) penumpang selama perjalanan armada. Secara global, sistem OBU dan VMS akan terbagi menjadi dua macam yaitu OBU side dan Halte Side [4], [5]. Sistem OBU dan VMS dapat dilihat pada gambar 2.



Gambar 2.1 Arsitektur jaringan OBU

2.3.2 TMC

TMC (Traffic Management Center) dalam ITS merupakan pusat kendali dan pemantauan pergerakan armada transportasi publik. TMC dibangun dengan tujuan untuk mewujudkan suatu sistem manajemen armada dan manajemen revenue yang dapat berjalan dengan baik serta dapat digunakan untuk memantau kinerja secara real time. Konsep kerja dari TMC dimulai dari perangkat OBU yang mengirimkan data berupa data koordinat, data ticketing, dan data kondisi dalam bus maupun luar bus, kemudian seluruh data tersebut disimpan dalam server TMC dan ditampilkan ke wall-display [4], [6]. Arsitektur jaringan TMC dapat dilihat pada gambar 3.



Gambar 2.2 Arsitektur jaringan TMC

2.4 Sistem Keamanan RFID

Sebagaimana seperti sistem teknologi telekomunikasi dan informasi, sistem RFID juga berpotensi menghadapi ancaman bahaya seperti diawasi atau dimanipulasi. Berikut akan dijelaskan beberapa tipe serangan yang mungkin terjadi serta beberapa prosedur perlindungan dari serangan [7].

2.4.1 Penyerangan pada RFID

Terdapat beberapa jenis serangan yang dapat dilakukan oleh *attacker* tergantung pada bagian sistem RFID [7], yakni :

1. Bagian *Transpoder*, pada bagian ini bisa dilakukan beberapa aksi penyerangan antara lain:
   - *Destruction*, dengan merusak / memotong antena, menghancurkan chip secara langsung, serta meletakkan transponder ke medan yang kuat seperti *microwave oven.*
   - Shielding / stunning, melapisi transponder dengan permukaan metal seperti aluminium foil.
   - Spoofing dan cloning, dengan membuat tiruan transponder (transponder clone) dengan cara mengganti PROM dengan EPROM.

2. Bagian Reader
3. Bagian RF Interface, di bagian ini juga terdapat banyak kemungkinan penyerangan antara lain :
    - *Eavesdropping* (menguping), dimana pihak ketiga hanya ingin melihat pesan yang dikirimkan tanpa mengubah apapun.
    - *Jamming*, mengirim sinyal yang dapat mengganggu atau menginterferensi sinyal pembawa.
    - Memperluas jangkauan pembacaan, yang mana pihak penyerang dapat membaca data transponder dari jarak aman yang tidak terdeteksi dengan melakukan *inductive coupling* atau *backscatter coupling.*
    - *Relay attack*, memperluas jangkauan antara *reader* dan *transponder* menggunakan perangkat transmisi *relay*.
    - DoS (*Denial of Service*) dengan *blocker tags*, yakni penyerang memblokir seluruh daerah pencarian dari *reader* sehingga pengguna tidak dapat melakukan transaksi.

2.4.2 Perlindungan dengan Kriptografi

Pada aplikasi yang membutuhkan keamanan tinggi seperti ticketing atau pembayaran sebagaimana digunakan dalam sistem ITS ini, kelalaian dalam prosedur kriptografi bisa berakibat pada kerugian besar apabila attacker berhasil memanipulasi transponder dan menggunakannya untuk mendapatkan layanan tanpa otorisasi.

Sistem RFID dengan sistem keamanan yang tinggi harus memiliki pertahanan untuk mengatasi [7] :

- Skimming pada sinyal pembawa data yang ingin melakukan duplikat atau modifikasi pada data.
- Adanya pembawa data asing yang sengaja diletakkan oleh attacker agar memperoleh akses tanpa otoritas sehingga bisa bertransaksi tanpa bayar.
- *Eavesdropping & replaying*, reader RFID palsu mengumpulkan data penting saat terjadi komunikasi antara reader dan transponder lalu mengimitasinya untuk kemudian digunakan kembali agar mendapatkan akses ilegal.

2.4.2.1 Mutual Symmetrical Authentication

Autentikasi mutual diantara reader dan transponder didasarkan pada prinsip autentkasi yang saling terpisah sesuai yang ada dalam ISO/IEC 9798-2. Di mana setiap pihak yang berkomunikasi saling memeriksa kunci kriptografi dari pihak lainnya [7].

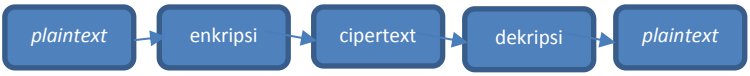
Prosedur dimulai dengan reader mengirim GET_CHALLENGE ke transponder. Bilangan acak RA dibangkitkan oleh transponder lalu dikirim kembali ke reader. Selanjutnya reader akan membangkitkan bilangan acak RB. Dengan menggunakan kunci privat K dan algoritma kunci ek, reader menghitung blok data yang terenkripsi (Token 1), yang mengandung bilangan acak dan beberapa kontrol data, dan kemudian mengirim blok data ke transponder.

2.4.2.2 Autentikasi menggunakan Derived Key

Dengan menggunakan prosedur mutual autentication, terdapat beberapa kelemahan yang salah satunya adalah setiap transponder dilindungi dengan kunci yang identik, di mana hal tersebut tidak terlalu aman untuk digunakan pada sistem ticketing atau pembayaran. Sehingga cara yang bagus untuk meningkatkan prosedur autentikasi adalah dengan memberikan kunci yang berbeda di setiap transponder.

2.4.2.3 Autentikasi menggunakan Derived Key

Apabila digeneralisasi, terdapat dua tipe penyerangan dasar yang biasa ditemui pada keamanan sistem, yakni yang bersikap pasif seperti eavesdropping dan aktif yang berusaha memanipulasi data. Dalam mengenkripsi data, pola yang dimiliki dalam metodenya hampir selalu sama seperti gambar berikut.

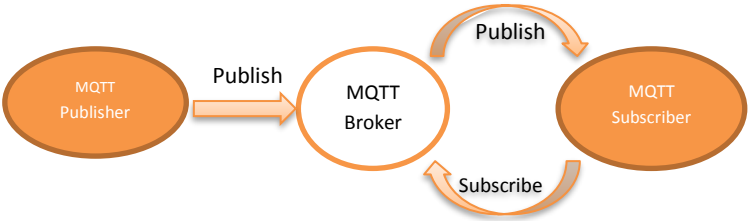


Gambar 2.3 Alur enkripsi-dekripsi

Dikatakan dalam buku RFID Handbook bahwa dalam sistem RFID hanya memiliki prosedur simetrik karena dipercaya bahwa kecepatan akan lebih diutamakan di dalam layanan penggunaannya

2.5 MQTT

MQTT (Message Queuing Telemetry Transport) merupakan protokol komunikasi antara machine to machine dan Internet of Things (IoT) yang memiliki cara kerja hampir sama dengan client-server, namun dalam MQTT lebih dikenal dengan istilah publisher dan subscriber. MQTT pertama kali dikembangkan oleh Andy Stanford-Clark dari IBM dan Arlen Nipper dari Arcom pada tahun 1999 dan distandardisasi pada tahun 2013 oleh OASIS (Organization for the Advacement of Structured Information Standards). Tujuan awal dari diciptakannya MQTT adalah untuk meminimalkan daya dari proses serta meminimalkan penggunaan bandwidth [8], [9].



Gambar 2.4 Komponen utama Protokol MQTT

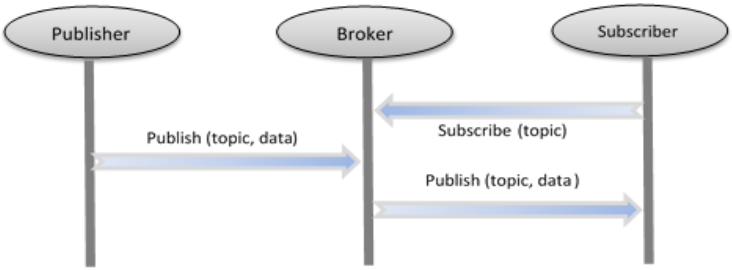
Terdapat tiga komponen utama dalam protokol MQTT yakni broker, subscriber, dan publisher. Publisher dan subscriber saling bertukar pesan melalui broker menggunakan MQTT paket kontrol. Publisher menghasilkan data lalu mengirimkan data tersebut

ke broker. Di broker, data yang diterima dari publisher diorganisir dalam bentuk topik yang di subscribe oleh subscriber untuk menerima notifikasi dari broker kapanpun data baru ditulis di dalam topik.

2.5.1 Komunikasi MQTT

Seperti yang telah disebutkan sebelumnya, terdapat tiga komponen utama dalam MQTT yakni broker, publisher, dan subscriber. Meskipun MQ adalah singkatan dari Message Queue yang berarti antrian pesan, namun saat ini model komunikasi pada MQTT tidak menggunakannya lagi melainkan menggunakan model lain, publish dan subscribe [10].

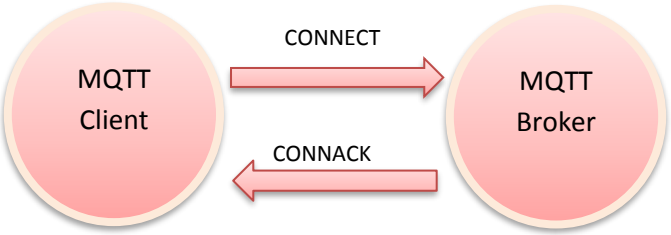
Pada model komunikasi client-server yang biasa, client akan berkomunikasi langsung dengan sebuah end-point. Namun dalam model publish-subscribe, terdapat pihak ketiga sebagai penengah yang disebut dengan broker, yang menengahi komunikasi antara pengirim atau publisher dan penerima atau subscriber.



Gambar 2.5 Alur komunikasi sederana protokol MQTT

Publisher dan subscriber saling bertukar informasi melalui broker menggunakan MQTT paket kontrol. Beberapa tipe paket kontrol yang sering ditemui adalah CONNECT, CONNACK, PUBLISH, PUBACK, SUBSCRIBE, SUBACK, dan DISCONNECT. Sisi client memberitahukan keadaannya ke broker dengan saling bertukar paket kontrol PINGREQ (request) dan PINGRESP (respon) [11].

Koneksi diawali dari sisi client yang mengirimkan pesan CONNECT ke broker. Kemudian broker akan membalas dengan mengirimkan CONNACK dan kode status. Sekali terhubung, broker akan terus membuka koneksi selama tidak ada perintah diskoneksi. Terdapat dua entri data dalam pesan CONNACK, flag sesi saat ini dan return code koneksi. Berikut adalah gambaran sederhana koneksi antara client dan broker.



Gambar 2.6 koneksi sederhana antara Client dan Broker

2.5.2 Mengamankan Data MQTT

Untuk memberikan layanan keamanan pada protokol MQTT, secara garis besar dibagi menjadi dua cara yakni dengan memasang TLS/SSL atau dengan mengenkripsi payload dengan kriptografi [12], [13].

TLS/SSL merupakan teknologi yang digunakan pada web dan merupakan bagian dari protokol TCP/IP, bukan MQTT. Dengan TLS, akan menghasilkan sebuah saluran yang telah terenkripsi yang mana data-data MQTT akan dilewatkan, sehingga akan mengamankan seluruh bagian dari MQTT, tidak hanya payload dari data. Jika memasang TLS, maka MQTT akan bekerja pada port 8883 yang merupakan port aman (secure) dari MQTT dan tidak lagi berada pada port 1883 yang merupakan port default.

Enkripsi payload merupakan enkripsi pada data-data spesifik, biasanya dalam payload paket PUBLISH atau CONNECT LWT, yang berada pada level aplikasi. Dengan ini bisa digunakan skenario pendekatan end-to-end encryption meskipun melewati atau berada pada lingkunngan tidak aman [14]. Selain itu juga

kerahasiaan data akan terjamin karena pihak broker tidak akan bisa melihat isi dari data yang lewat.

2.6 Hash Function

Fungsi hash berfungsi untuk meringkas sebarang pesan dengan panjang yang bervariasi. Fungsi hash yang baik memiliki properti yang akan menghasilkan output yang terdistribusi rata dan random ketika diaplikasikan pada set input yang besar. Nilai hash function dinotasikan sebagai h, dengan formula h= H(M). Dimana H adalah fungsi hash dan M adalah blok-blok data dengan panjang bervariasi yang berperan sebagai input [1].

Di dalam bidang keamanan, fungsi hash yang dibutuhkan biasa disebut dengan cryptographic hash function. Yang mana cryptographic hash function merupakan algoritma yang tidak mudah dipecahkan dari segi komputasi, sebagaimana seperti sifat dari hash function sendiri yang merupakan fungsi satu arah (one way function) serta bebas dari data yang memiliki nilai hash sama (collision-free). Dengan begitu, fungsi hash dapat digunakan untuk menentukan adanya perubahan dalam data [1].

2.6.1 Penggunaan Hash Function

a. Autentikasi Pesan (Message Authentication)

Adalah sebuah mekanisme untuk membuktikan integritas dari suatu pesan. Hal ini akan menyatakan bahwa data yang diterima sesuai dengan data yang dikirimkan tanpa adanya perubahan apapun. Umumnya, autentikasi pesan didapatkan dengan menggunakan Message Authentication Code (MAC). Biasanya MAC digunakan oleh dua pihak yang saling berbagi kunci private untuk mengautentikasi pertukaran pesan. Apabila ingin memeriksa integritas dari pesan, maka fungsi MAC dapat diaplikasikan di pesan dan asilnya akan dibandingkan dengan nilai MAC yang berhubungan. Apabila ada attacker yang ingin mengubah pesan, maka Ia tidak akan bisa mengganti nilai MAC tanpa mengetahui kunci private.

b. Digital Signature

Cara kerja dari digital signature hampir sama dengan MAC. Bedanya adalah nilai hash dari pesan dienkripsi dengan kunci private pengirim. Sehingga jika ingin memeriksa integritas dari pesan tersebut, penerima harus memiliki kunci publik pengirim untuk mengetahui tanda tangan digital tersebut.

c. Aplikasi Lainnya

Fungsi hash juga sering digunakan untuk membuat file one-way password, yang mana hash dari password disimpan oleh sebuah sistem operasi daripada password itu sendiri sehingga password asli tidak bisa didapatkan oleh hacker yang menginginkan hak akses. Selain itu, fungsi hash juga dapat digunakan sebagai deteksi intrusi dan deteksi virus. H(F) dari setiap file disimpan di dalam sistem serta nilai hash juga harus diamankan. Untuk mengetahui adanya perubahan harus dihitung H(F), sehinga apabila ingin mengubah pesan tanpa diketahui harus mengubah F tanpa mengubah H(F).

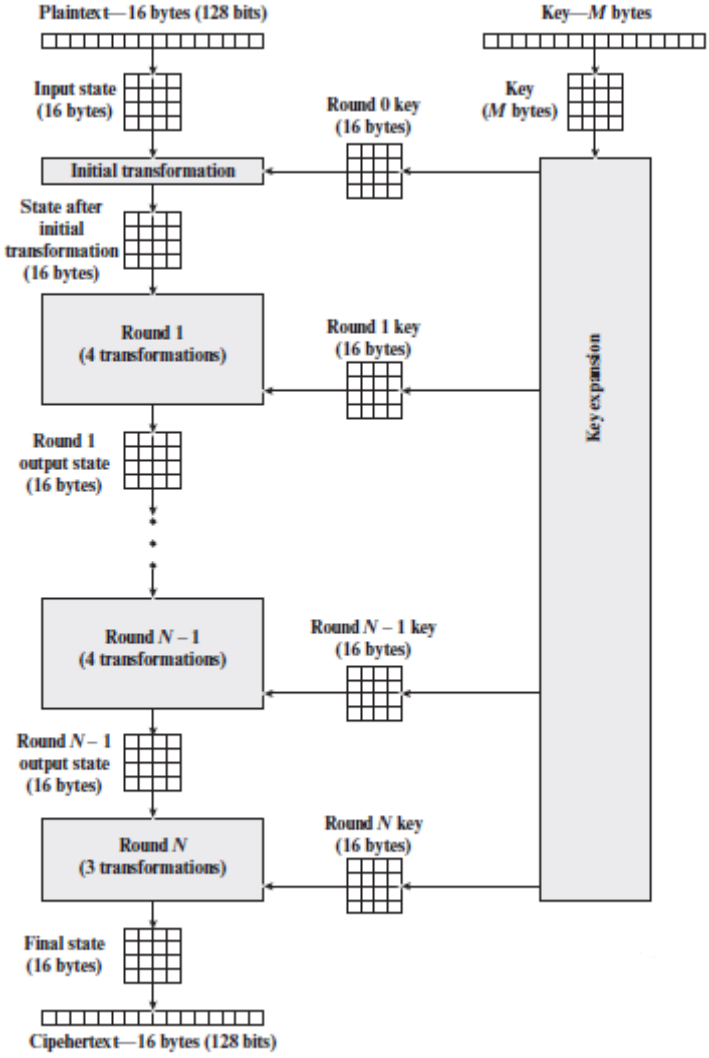
2.7 AES

Merupakan standard yang dikeluarkan oleh National Institute of Standards and Technology (NIST) pada tahun 2001 yang masuk dalam kategori Standard Keamanan Komputer subkategori Kriptografi. Algoritma yang digunakan dalam AES adalah block cipher simetris yang dapat mengenkripsi dan mendekripsi informasi. Oleh karena itu algoritma ini juga bisa digunakan untuk melindungi data-data elektronik. Unit basis yang digunakan dalam algoritma AES adalah byte, sehingga rangkaian dari 8 bit akan diperlakukan sebagai entitas tunggal atau satu kesatuan [1], [15].

2.7.1 Struktur AES secara Umum

Input dan output pada proses enkripsi dan dekripsi yang digunakan dalam algoritma AES adalah rangkaian atau sekuen dari 128 bit (digit dengan nilai 1 atau 0), yang mana rangkaian bit itu kemudian disebut sebagai blok . Blok tersebut kemudian diolah dengan menggunakan kunci cipher dengan panjang 128, 192, atau 256 bit [16]. Sesuai dengan FIPS PUB 197 [17], blok tersebut

digambarkan sebagai matriks 4 x 4 dalam byte. Berikut merupakan struktur keseluruhan proses enkripsi AES.



Gambar 2.7 Cara kerja AES

Cipher terdiri dari N round, yang mana jumlah dari round bergantung pada panjang kunci yang digunakan. Terdapat tiga kunci dalam AES yakni 128, 192, dan 256 bit. Sedangkan round sendiri adalah rangakaian fungsi transformasi yang terjadi dalam proses enkripsi dan dekripsi AES. Terdapat empat fungsi transformasi berbeda yang terjadi dalam satu round, yakni SubBytes, ShiftRows, MixColumns, dan AddRoundKeys. Namun pada round terkahir hanya dilakukan tiga transformasi tanpa AddRoundKey, tapi sebenarnya AddRoundKey dilakukan pertama kali sebelum round 1 sehingga bisa disebut round 0. setiap transformasi menggunakan matriks 4x4 sebagai input dan menghasilkan matriks 4x4 sebagai input [1].

Tabel 2.1 Parameter AES

	AES-128	AES-192	AES-256
Key size	4 word / 16 byte	6 word / 24 byte	8 word / 32 byte
Plaintext block size	4 word / 16 byte	4 word / 16 byte	4 word / 16 byte
Number of round	10	12	14
Round key size	4 word / 16 byte	4 word / 16 byte	4 word / 16 byte
Expanded key size	44 word / 176 byte	52 word / 208 byte	60 word / 240 byte

2.7.2 Detail Struktur AES

Seperti yang telah disebutkan sebelumnya, AES cipher memiliki empat tahapan dalam round, satu permutasi dan tiga substitusi. Masing-masing dari tahapan tersebut akan dijelaskan pada pembahasan berikut.

2.7.2.1 Subtitute Bytes (SubByte)

Merupakan substitusi byte non-linear yang beroperasi secara independen di setiap bytenya dengan menggunakan tabel substitusi (S-box). Dimana S-box sendiri adalah kumpulan nilai-nilai byte yang didefinisikan oleh AES sebagai matriks 16x16. Di dalam S-box terdapat permutasi dari semua 256 kemungkinan nilai 8-bit. Sehingga, setiap input yang masuk akan digantikan nilainya sesuai dengan yang ada pada S-Box.

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Gambar 2.8 S-Box

Gambar di atas merupakan S-Box beserta nilainya yang telah disediakan oleh standar AES untuk melakukan proses SubByte. Nilai pada baris dan kolom berfungsi sebagai indeks untuk memilih output 8 bit yang bernilai unik. Misalnya, heksadesimal dengan nilai {95} merujuk pada baris 9, kolom 5 dari S-Box, yang mana mengandung nilai {2A}. Maka hasil SubByte dari nilai {95} memiliki output nilai {2A}. Selain S-Box, juga terdapat Inverse S-Box yang memiliki nilai kebalikan dari S-Box. Dimana inpit yang bernilai {2A} akan memiliki nilai output {95}, seperti yang ditampilkan pada gambar berikut ini.

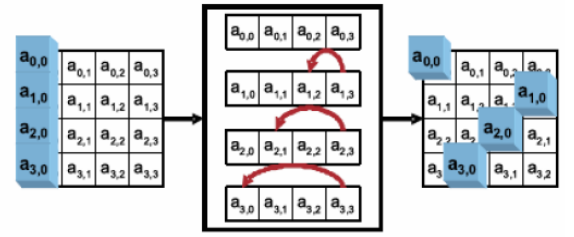
		y														
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Gambar 2.9 Inverse S-Box

Cara mudah untuk melakukan operasi SubBytes adalah dengan memerlakukan S-box sebagai tabel look-up, sehingga untuk state matriks 4x4 atau input data akan disubstitusikan dengan nilai yang sesuai dengan yang ada pada S-box yang kemudian akan berlaku sebagai output dari SubByte. Setelah operasi SubByte selesai, operasi selanjutnya adalah Shiftrow.

2.7.2.2 Shift Rows

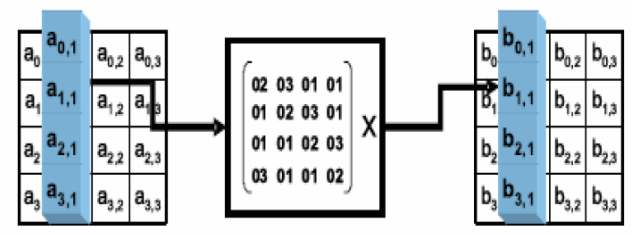
Dalam operasi Shiftrow, proses operasinya terjadi pada baris-baris yang berbeda dari setiap matriks input. Cara kerjanya adalah dengan melakukan rotasi sederhana dengan rentang yang berbeda di setiap barisnya. Pada baris kedua dari 4x4 byte input digeser satu byte ke kiri dari posisi aslinya, baris ketiga digeser dua byte ke kiri, dan baris keempat digeser 3 byte ke kiri dari posisi awal. Sedangkan untuk baris pertama tetap dibiarkan begitu saja. Proses shiftrow lebih jelasnya dapat dilihat pada gambar 2.10. Setelah melewati operasi Shifrow, state selanjutnya masuk kedalam operasi Mixcolumn.



Gambar 2.10 Proses kerja ShiftRow

2.7.2.3 Mix Columns

Berbeda dengan operasi sebelumnya ynag bekerja pada baris dalam State matriks 4x4, Mixcolumn beroperasi pada bagian kolomnya. Cara kerjanya, semua kolom yang ada di dalam State diambil dan digabungkan (XOR) dengan data yang independen satu sama lain untuk membentuk suatu kolom baru [16].



Gambar 2.11 Mix Column

2.7.2.4 AddRoundKey

Pada operasi Addroundkey, setiap byte input yang berhubungan di XOR dengan kunci ekspansi. 128 bit dari state input di XOR dengan 128 bit milik round key. Dari gambar dibawah, dapat dilihat proses transformasi AddRoundKey dimana matriks pertama adalah State dan input kedua adalah round key.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

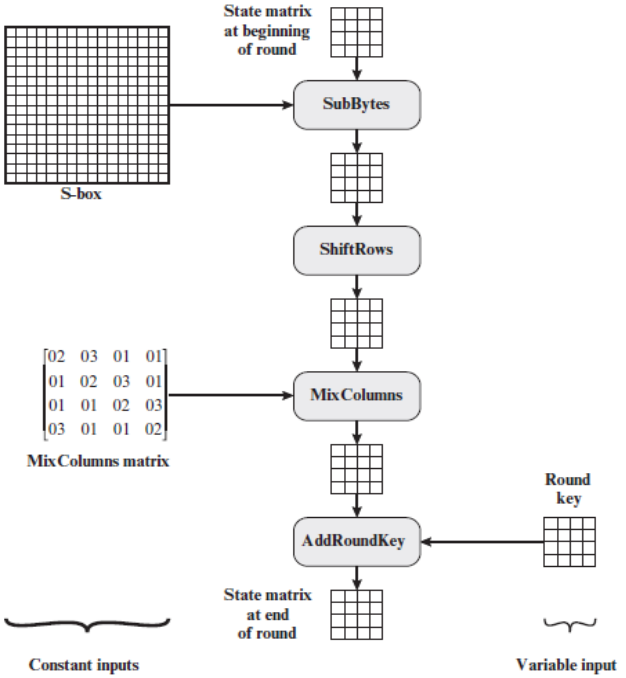
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

Gambar 2.12 Proses AddRound Key

Dari beberapa proses yang terjadi dalam algoritma AES, dapat disimpulkan bahwa dalam sekali putaran pada input dilakukan 4 proses berbeda yakni, SubByte, ShiftRows, MixColumn, dan AddRoundKey yang bisa digambarkan sebagai berikut.



Gambar 2.13 Proses dalam satu round AES

2.7.3 Mode Operasi AES

Di dalam AES, terdapat lima mode operasi sering digunakan ketika diaplikasikan pada blok cipher dalam berbagai aplikasi. Beberapa blok cipher tersebut adalah [17] :

a. *Electronic Codebook Mode* (ECB) : mode ini biasanya digunakan untuk data yang lebih pendek dari panjang blok. Apabila data lebih panjang, maka harus dibagi menjadi beberapa blok sesuai panjang yang dibutuhkan, bisa juga menambahkan padding pada blok terakhir bila perlu.
b. *Cipher Block Chaining Mode* (CBC) : pada mode ini, *plaintext* di-XOR dengan blok *ciphertext* yang sebelumnya sebelum dienkripsi. Setelah dienkripsi, hasil *ciphertext* disimpan di dalam feedback register untuk kemudian digunakan untuk XOR blok *plaintext* berikutnya hingga akhir.
c. *Cipher Feedback Mode* (CFB) : tipe ini mengizinkan konversi blok cipher menjadi stream cipher. CFB menghilangkan kebutuhan padding seluruh data dan menjadikannya sebagi nomor integral dari blok yang telah digunakan dalam proses. Bit yang paling kiri di-XOR dengan segmen pertama dari *plaintext* untuk mendapatkan *ciphertext*.
d. *Output Feedback Mode* (OFB) : hampir sama dengan CFB, namun disini OFB menghilangkan hasil blok *plaintext* dengan blok *ciphertext* yang sama dengan menggunakan mekanisme internal feedback yang mana independen terhadap string *plaintext* dan *ciphertext*.
e. *Counter Mode* (CTR) : dalam mode ini, nilai counter harus berbeda untuk setiap blok yang dienkripsi. Counter dienkripsi dan di-XOR dengan *plaintext* agar mendapatkan blok *ciphertext* tanpa chaining. Untuk mendapatkan *plaintext* kembali tinggal melakukan proses secara terbalik dengan counter yang sama lalu di-XOR.

2.7.4 Kekuatan Algoritma AES-128

Algoritma AES-128 dipilih dalam tugas akhir ini dikarenakan oleh beberapa alasan, antara lain adalah karena metode ini merupakan metode yang paling sering dipakai oleh instansi pemerintah Amerika dan merupakan standar bagi keamanan data sampai level *top secret.* Jika dilihat dari panjang kunci, algoritma AES-128 termasuk dalam kategori *computationally infeasible* atau tidak mudah dipecahkan secara komputasi. Hal ini dikarenakan, dengan panjang kunci 128 bit, jika ingin menemukan kombinasi kunci dengan brute force, maka terdapat 2^{128} kunci alternatif lain.

Seandainya seseorang bisa menghasilkan 1 kunci dalam waktu 1 µs, maka dia akan berhasil menemukan kunci yang benar dalam waktu 5.4 x 10^{24} tahun. Dan bila seseorang bisa menghasilkan sejuta kunci dalam waktu 1 µs, maka dia bisa memecahkan kunci dalam waktu 5.4 x 10^{18} tahun.

Tabel 2.2 Kekuatan AES dengan brute force attack

Key Size (bits)	Number of Alternative Keys	Time required at 1 encryption / µs	Time required at 10^6 encryption / µs
32	2^{32}= 4.3 x 10^9	2^{32} µs= 35.8 minutes	2.15 miliseconds
56	2^{56}= 7.2 x 1016	2^{56} µs = 1142 years	10.01 hours
128	21^{28} = 3.4 x 10^{38}	2^{128} µs = 5.4 x 10^{24} years	5.4 x 10^{18} years
168	21^{68} = 3.7 x 10^{50}	2^{168} µs = 5.9 x 10^{36} years	5.9 x 10^{30} years

2.8 Phyton

Python merupakan bahasa pemrograman level tinggi yang dapat melakukan eksekusi sejumla instruksi secara langsung (interpretatif) dengan metode orientasi objek (Object Oriented Programming) serta menggunakan semantik dinamis untuk memberikan tingkat keterbacaan syntax.

Syntax python dapat dijalankan dan ditulis untuk membangun aplikasi di berbagai sistem operasi, antara lain Linux/Unix, Windows, Mac OS, Android, Jawa Virtual Machine, Raspbian, dll. beberapa fitur yang dimiliki python adalah sebagai berikut :

a. Memiliki pustaka yang luas; dalam distribusi Python telah disediakan modul-modul 'siap pakai' untuk berbagai keperluan.
b. Memiliki tata bahasa yang jernih dan mudah dipelajari.
c. Memiliki aturan *layout* kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber.
d. Berorientasi objek.
e. Memiliki sistem pengelolaan memori otomatis (*garbage collection*, seperti java)
f. Modular, mudah dikembangkan dengan menciptakan modul-modul baru. modul-modul tersebut dapat dibangun dengan bahasa python maupun c/c++.
g. Memiliki fasilitas pengumpulan sampah otomatis, seperti halnya pada bahasa pemrograman java, python memiliki fasilitas pengaturan penggunaan memori komputer sehingga para pemrogram tidak perlu melakukan pengaturan memori komputer secara langsung.
h. Memiliki banyak faslitas pendukung sehingga mudah dalam pengoprasiannya.

2.9 Fernet

Fernet adalah implementasi dari metode enkripsi simetrik (symmetric enryption) yang memastikan bahwa pesan yang telah terenkripsi tidak dapat dimanipulasi ataupun diubah tanpa kunci

tertentu. Metode enkripsi yang digunakan dalam fernet adalah AES-128-CBC dan menggunakan PKCS7 untuk padding. Fernet juga menggunakan HMAC dengan SHA-256 untuk autentikasinya [18].

Modul Fernet tersedia di dalam cryptography package yang ada di environment pemrograman milik python. Dalam cryptography sendiri sebenarnya dibagi menjadi dua level. Yang pertama, memiliki resep kriptografi yang aman dan hanya membutuhkan sedikit konfigurasi. Oleh karena itu, level ini mudah dan aman untuk digunakan serta developer tidak butuh untuk membuat banyak keputusan. Dan modul Fernet berada di dalamnya.

Level yang lainnya yaitu primitif kriptografi tingkat rendah. Level ini memiliki karakteristik yang lebih berbahaya dan terkadang bisa digunakan dengan cara yang salah. Selain itu juga membutuhkan keputusan yang tepat serta pengetahuan yang mendalam tentang konsep kriptografi. Karena adanya potensi bahaya, level ini sering disebut dengan 'hazmat layer' dan berada dalam package cryptography.hazmat [18]. Maka dari itu, untuk penggunaannya, lebih direkomendasikan untuk menggunakan pada level aman pada para developer.

Beberapa hal yang dapat ditemukan di layer aman antara lain adalah Fernet, untuk melakukan enkripsi simetrik, dan X.509 untuk bagian enkripsi asimetrik (public key infrastructure). Sedangkan di layer hazmat, kita bisa menggunakan algoritma-algoritma primitif seperti:

a. Enkripsi yang terautentikasi
b. Algoritma asimetrik
c. Enkripsi simetrik
d. Fungsi waktu konstan
e. Fungsi *key derivative*
f. Key warpping
g. *Message authentication codes* (MAC)
h. *Message digest* (Hashing)
i. Symmetric padding
j. Two-factor authentication
k. Random number generation

2.10 SQL

SQL adalah singkatan dari Structured Query Language, merupakan bahasa pemrograman yang digunakan untuk mengakses dan mengelola database. Sedangkan MySQL adalah program yang dapat mengerti SQL. SQL menunjukkan pernyataan atau statement yang menunjukkan informasi yang terkandung dalam database beserta tabel-tabelnya. Beberapa al yang dapat dilakukan dengan SQL antara lain adalah :

- Memasukkan, memperbarui, atau menghapus catatan dalam database.
- Membuat database dan tabel baru, prosedur penyimpanan, dan peninjauan.
- Mengeluarkan data dari database, dll.

Database sendiri adalah kumpulan dari data yang diatur sedemikian hingga mempermudah memberikan efisiensi dalam akses, manajemen, dan update. Database terdiri dari kumpulan tabel yang memiliki informasi yang relevan satu sama lain.

2.11 PHPMyAdmin

PHPMyAdmin adalah satu dari sekian banyak perangkat lunak yang dapat digunakan untuk mengelola database dalam MySQL. Dengan ini, membuat, mengisi, memperbarui, dan mengelola tabel dapat dilakukan dengan mudah tanpa harus menghafal semua perintah atau query yang dibutuhkan. Hal ini disebabkan karena interface yang digunakan sangat user-friendly sehingga mudah digunakan.

2.12 Quality of Services

Terdapat 3 level Quality of Service yang disediakan pada protokol MQTT untuk menjaga realibilitas data dalam protokol MQTT yang digolongkan sesuai dengan jenis atau tipe data yang sedang ditransmisikan. Berikut adalah tipe-tipe QoS dalam protokol MQTT.

Tabel 2.3 Pembagian QoS pada MQTT

QoS level	Message Guarantee	Behavior
QoS 0	At most once	Pesan dikirim ke semua subscriber sekali, tidak ada pengiriman kembali, dan tidak ada konfirmasi (ACK) dari penerima
QoS 1	At least once	Pesan dikirim ke semua subscriber paling tidak sekali dan ada konfirmasi dari penerima
QoS 2	Exactly once	Pesan dikirim ke seluruh subscriber sekali, tanpa duplikasi, tanpa pesan konfirmasi tambahan untuk menghindari duplikasi pesan, dan menjamin pengiriman pesan

Pada QoS level 0, data dikirim bergantung pada kekuatan jaringan. Data akan dikirim hanya sekali atau bahkan tidak sama sekali dan tidak dibutuhkan konfirmasi penerimaan (*acknowledgment*) dari subscriber. Pada QoS level 1, data akan dikirim paling tidak sekali dan memungkinkan untuk terjadi duplikasi pada pesan. Publisher akan mengirim pesan dan menunggu *acknowledgment* (PUBACK) dari subscriber. Dan apabila tidak menerima acknowledgment, pesan akan dikirim kembali dengan flag duplikat *(DUP flag set)* QoS level 2 merupakan level QoS tertinggi, yang mana protokol tambahan dibutuhkan untuk memastikan data benar-benar hanya dikirim satu kali [11].

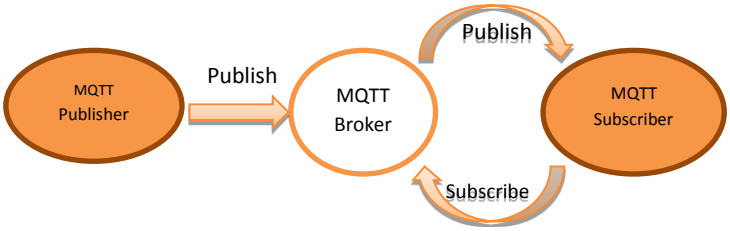
Halaman ini mengandung kekosongan

BAB 3 PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bagian ini akan ditunjukkan bagaimana sistem keamanan akan dibangun dalam proyek tugas akhir ini dan pengimplementasiannya pada jaringan komunikasi ITS yang telah dibangun.

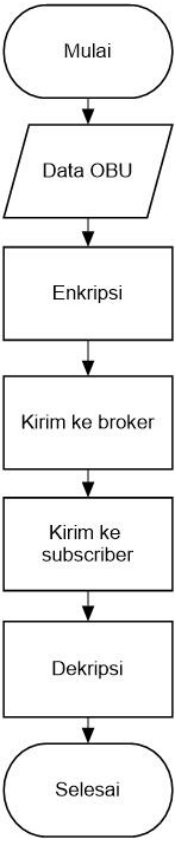
3.1 Perancangan Sistem Keamanan secara Global

Sistem komunikasi yang terjadi pada sistem ITS yang dibangun, dilakukan dengan standar protokol MQTT yang mana protokol tersebut berjalan di layer TCP/IP. Terdapat tiga komponen utama dalam komunikasi MQTT yakni broker, subscriber, dan publisher. Secara global, data yang didapatkan dari publisher akan dikirim menuju broker, di broker data akan disusun dan dipisahkan menurut topik masing-masing, baru setelah itu dilanjutkan menuju subscriber sesuai dengan topik ynag dipilih. Berikut adalah gambaran mengenai arsitektur jaringan pada MQTT.



Gambar 3.1 Arsitektur sederhana MQTT

Untuk mengamankan jaringan pada MQTT dapat dilakukan dengan dua cara, yakni dengan menggunakan TLS/SSL atau dengan mengenkripsi payload yang dikirimkan melalui jaringan [11]. Pada tugas akhir ini akan dilakukan proses enkripsi dan dekripsi untuk melindungi data-data yang dikirimkan melalui jaringan. Untuk melakukan hal tersebut maka dibutuhkan konfigurasi langsung di bagian subscriber dan publisher, sehingga data-data yang terkirim tidak akan bisa dibaca saat melewati broker.



Gambar 3.2 Alur proses enkripsi data

Pada sistem ITS, perangkat OBU dan Server TMC sama-sama berperan menjadi publisher serta subscriber yang akan mengirimkan dan menerima data. Broker yang digunakan dalam jaringan MQTT ini diberi nama vps2.lawanghosting.pw.

Data-data yang diperoleh oleh OBU seperti GPS, data ticketing, dan lainnya akan melewati proses enkripsi terlebih dahulu sebelum akhirnya dikirimkan menuju broker. Di broker, data yang

diterima akan terlihat tidak beraturan karena diterima sebagai *ciphertext*.

3.2 Konfigurasi MQTT

Konfigurasi akan dilakukan pada semua komponen dalam MQTT, yakni broker, subscriber, dan juga publisher. Konfigurasi ini bertujuan untuk memastikan bahwa publisher telah bisa mengirimkan data-datanya ke subscriber melalui broker yang telah ditentukan.

3.2.1 Instalasi dan Konfigurasi MQTT Broker

MQTT broker yang akan digunakan pada tugas akhir ini adalah Mosquitto. Mosquitto sendiri merupakan salah satu MQTT server/broker yang paling populer dan mudah untuk diinstal serta di konfigurasi. Ubuntu 16.04 sudah memiliki versi Mosquitto terbaru dalam default software repositorinya. Cukup dengan log-in menggunakan non-root user dan install mosquitto dengan apt-get.

```
$ sudo apt-get install mosquitto
```

Untuk melakukan instalasi paho-mqtt dapat langsung dilakukan dengan menjalankan kode berikut :

```
$ Pip install paho-mqtt
```

Apabila menggunakan Windows, mosquitto.exe dapat langsung diunduh di situs mosquitto.org. Namun saat ingin menginstall, kita akan mendapatkan peringatan untuk mengunduh beberapa dependansinya seperti OpenSSL dan pathreads. Setelah berhasil mengunduh dua hal tersebut dan memindahkan file-file yang diperlukan ke dalam modul mosquitto, barulah aplikasi mosquitto dapat diinstall dengan aman.

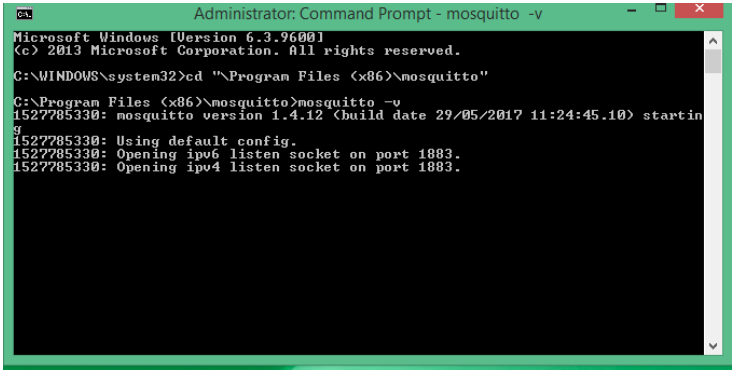
Paho-mqtt berada dalam environment pemrograman python dan tersedia dalam Python Package Indeks (PyPi). Jika di komputer

belum memiliki python, maka disarankan untuk memasangnya terlebih dahulu. Dan apabila tidak bisa menjalankan pip, disarankan untuk mengunduh package pip terlebih dahulu baru kemudian mengunduh paho-mqtt. Setelah berhasil di download, maka konfigurasi berikutnya akan mudah dilakukan.

Untuk memeriksa apakah mosquitto sudah dapat digunakan, bisa dengan dilakukan dengan menuliskan perintah

```
$ mosquitto -v
```

Dengan begitu akan terlihat bahwa mosquitto sudah bekerja dan telah berhasil mendengarkan socket pada port 1883, yang mana port 1883 merupakan default port untuk MQTT bekerja. Mosquitto yang telah berfungsi bisa dilihat pada gambar berikut.



Gambar 3.3 MQTT yang suda berjalan

3.2.2 Konfigurasi MQTT Subscriber

Di bagian subscriber, akan dilakukan konfigurasi untuk menentukan client, membuat koneksi dengan broker, serta menentukan topik yang akan disubscribe. Konfigurasi bisa dilakukan dengan menggunakan gedit. Pertama, membuat file .py untuk

meletakkan semua konfigurasi subscriber. Setelah itu dilanjutkan dengan menuliskan kode program berikut.

Kode program pertama yang dituliskan dalam gedit adalah untuk memasukkan mqtt.client sebagai paho, serta memasukkan fungsi waktu. Kode ini berfungsi untuk memanggil package paho mqtt yang telah di download sebelumnya dan agar bisa digunakan untuk membuat koneksi MQTT.

```
import paho.mqtt.subscribe as subscribe
```

Setelah itu, menentukan fungsi callback on_connect untuk saat ketika menerima respon CONNACK dari server. Fungsi ini bersifat opsional dan tidak harus dicantumkan dalam program.

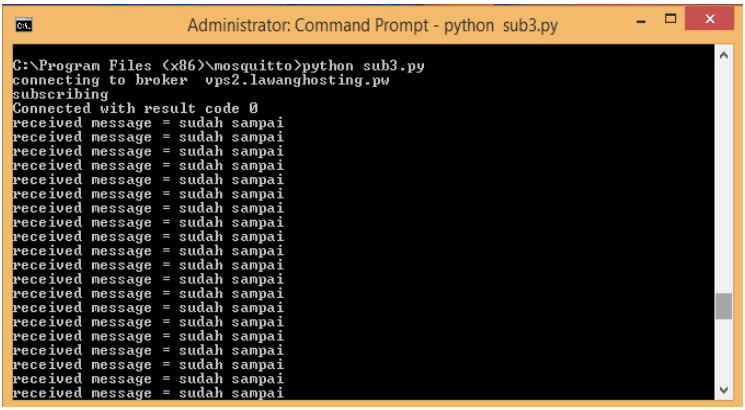
```
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
```

Fungsi Callback on_message ketika pesan PUBLISH diterima dari server menggunakan. Fungsi ini akan mengambil dan menampilkann isi message yang diterima dari publisher ketika dijalankan sesuai dengan topik dan brokernya. Dalam percobaan digunakan topik “paho/test1” dan broker dari laptop yang digunakan.

```
def on_message(client, userdata, message):
    print("%s %s" % (message.topic, message.payload))

subscribe.callback (on_message, "paho/test1", hostname= "192.168.1.5")
```

Hasil dari konfigurasi publisher dan subscriber yang berhasil dijalankan akan terlihat seperti Gambar 3.4 berikut, dimana pesan yang telah dikirimkan oleh sisi publisher berhasil diterima dan ditampilkan di bagian subscriber.



Gambar 3.4 Pesan yang diterima pada subscriber

3.2.3 Konfigurasi MQTT Publisher

Hampir sama dengan konfigurasi pada subscriber, di bagian pyblisher akan ditentukan pesan yang akan dikirimkan dalam suatu topik. Setelah membuat file .py untuk meletakkan kode konfigurasi publisher, kemudian menuliskan kode program sebagai berikut.

```
import time
import paho.mqtt.client as paho
```

Seperti pada subscriber, kode pertama yang harus ditulis adalah untuk memasukkan package mqtt.client sebagai paho dan memasukkan fungsi waktu. Kemudian menentukan broker yang akan dihubungkan dan menuliskan perintah berikut.

```
broker = “vps2.lawanghosting.pw”
client= paho.Client("pub-local")

message = open("original.txt","r")
msg = message.read()
```

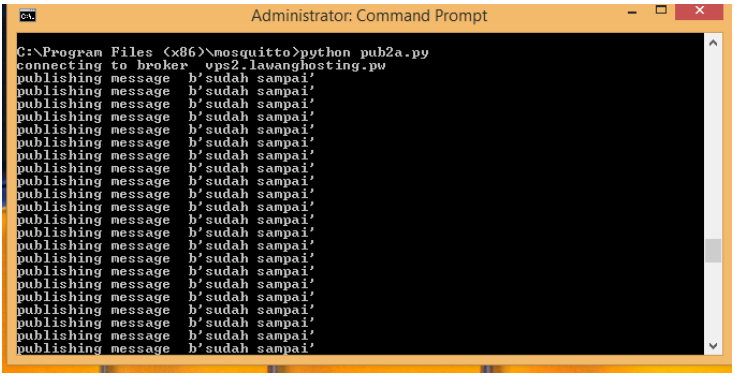
Perintah tersebut akan membuat sebuah klien bernama “pub-local” serta mendeklarasikan pesan yang akan dikirim menuju broker. Setelah itu membuat koneksi dengan broker menggunakan

```
Print(“connecting to broker”, broker)
Client.connect(broker)
Print(“publishing message”)
```

Selanjutnya data akan dipublish sesuai dengan topiknya dengan time sleep yang telah ditentukan. Time sleep akan membuat pesan terkirim sesuai dengan jeda waktu yang ditentukan.

```
While 1:
        Client.publish(“paho/test1”, out_message)
        Time.sleep(1)
Client.loop_forever()
```

Dengan begitu, data akan terus dipublish apabila tidak dihentikan. Berikut adalah hasil dari konfigurasi publisher yang sedang mengirimkan pesan.



Gambar 3.5 Pesan yang dikirim dari publisher

3.3 Konfigurasi Enkripsi Data

Untuk melindungi data yang dikirim dari serangan pihak luar dilakukan dengan mengenkripsi data. Enkripsi data pada MQTT dapat dilakukan dengan menggunakan modul fernet yang sudah tersedia di dalam package cryptography. Untuk menggunakan fernet, dimulai dengan menuliskan perintah untuk mengimport fernet seperti berikut ini.

```
From cryptography.fernet import Fernet
```

Selanjutnya menentukan kunci simetris yang akan digunakan di publisher dan subscriber. Kunci ini berfungsi untuk melakukan enkripsi dan dekripsi terhadap pesan yang dikirimkan. Karena bersifat simetris, maka kunci untuk dekripsi harus sama dengan kunci untuk enkripsi.

```
Key = Fernet.generate_key()
```

Perintah tersebut akan menghasilkan kunci fernet secara random yang dihasilkan dari os.urandom(), yang mana merupakan fasilitas dari sistem operasi untuk menghasilkan bilangan acak. Kunci yang dihasilkan harus disimpan dengan aman dan tidak boleh hilang. Karena menggunakan kunci simetris, maka kunci yang didapatkan harus diletakkan di subscriber dan di publisher. Berikut adalah contoh hasil dari kunci fernet yang dihasilkan oleh fernet key generator.

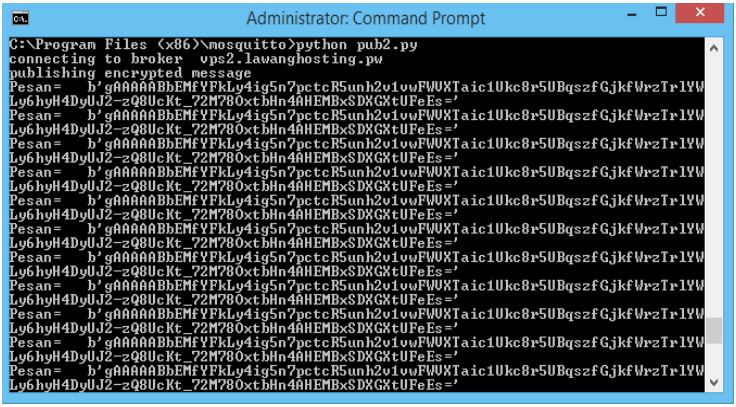
```
cipher_key= b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
```

Di bagian publisher, setelah didapatkan kunci selanjutnya adalah mengenkripsi data yang akan dikirim. Untuk mendapatkan pesan terenkripsi dapat dilakukan dengan memanggil fungsi fernet.

```
cipher = Fernet(cipher_key)
encrypted_message = cipher.encrypt(message)
out_message=encrypted_message.decode()
```

Hasil dari proses tersebut disebut sebagai “Fernet Token” dan memiliki jaminan privasi dan autentikasi yang kuat. Hal ini dikarenakan di dalam fernet, proses enkripsi data dilakukan dengan metode AES-128-CBC. Ditambah lagi, sebelum dienkripsi, data telah diberi tanda (sign) dengan menggunakan HMAC SHA-256.

Setelah itu, data yang telah dienkripsi siap untuk dipublish. Berikut ini adalah hasil enkripsi data yang diperoleh di sisi publisher.



Gambar 3.6 Hasil publikadi pesan yang terenkripsi

Pada sisi Subscriber juga harus dikonfigurasikan script untuk mendekripsi data yang diterima dari broker akan mendapatkan data yang sebenarnya. Untuk itu, pada script subcriber juga harus menambahkan modul fernet.

```
From cryptography.fernet import Fernet
```

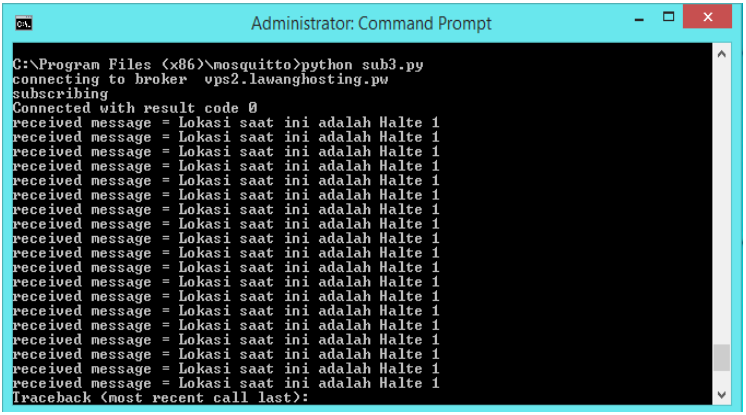
Setelah itu, tidak lupa menambahkan kunci yang sama dengan publisher yang telah dihasilkan dengan generate_key(). Dalam hal ini, kunci yang digunakan adalah sebagai berikut.

```
cipher_key= b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
cipher = Fernet(cipher_key)
```

Untuk mendapatkan *plaintext* dapat dilakukan dengan mendekripsi payload dari data yang dikirimkan

```
decrypted_message = cipher.decrypt(message.payload)
print("received message = ",
str(decrypted_message.decode("utf-8")))
```

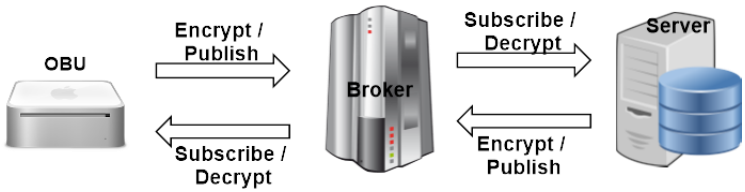
Dengan memanggil fungsi dekripsi yang ada dalam modul fernet dan kunci yang sama dengan saat melakukan enkripsi, proses dekripsi disini akan menghasilkan *plaintext* sesuai dengan data yang dikirimkan.



Gambar 3.7 Hasil Dekripsi pada subscriber

3.4 Implementasi Program pada Sistem ITS

Setelah kode program ditulis dan berhasil dijalankan, bagian berikutnya adalah mengimplementasikan pada sistem ITS yang sebenarnya. Apabila yang tadi adalah percobaan dalam pengiriman data yang sudah terenkripsi antara publisher dan subscriber biasa, sekarang akan dilanjutkan dengan melakukan percobaan pengiriman data pada sistem komunikasi ITS yang telah dibangun.



Gambar 3.8 Implementasi pada sistem komunikasi ITS

Pada ITS, data yang diolah didapat dari sensor-sensor yang berada di OBU. Data tersebut berupa data lokasi dari GPS serta nomor seri RFID untuk ticketing. Disini OBU dan Server sama-sama berperan sebagai publisher dan subscriber karena keduanya saling mengirim dan menerima.

Data yang diterima OBU akan di enkripsi dan ditandai terlebih dahulu menggunakan metode AES-128-CBC dan HMAC SHA-256 dengan modul Fernet yang ada di dalam package cryptography. Setelah itu data berupa *ciphertext* akan dikirim (publish) sesuai dengan topiknya menuju broker dan diteruskan menuju server. Sebagai subscriber, Server akan menerima data sesuai dengan topik yang ikuti. Karena data yang diterima masih berupa *ciphertext*, maka subscriber akan mendekripsi *ciphertext* terlebih dahulu dan menghasilkan *plaintext* (teks asli). Setelah data diolah oleh server, kemudian akan dikirim kembali menuju OBU untuk ditampilkan ke penumpang yang mana saat publish dan subscribe juga dilakukan proses enkripsi dekripsi yang sama. Proses ini dapat dilihat pada Gambar 3.8.

Sekali lagi akan dilakukan konfigurasi. Kali ini, konfigurasi akan dilakukan pada bagian OBU sebagai publisher, dimana OBU akan mendapatkan data berupa data lokasi dari GPS yang di-tap pada RFID reader. Seperti sebelumnya, hal pertama yang harus dilakukan adalah mengimpor modul-modul yang dibutuhkan, ditambah dengan modul serial agar bisa membaca nomor serial dari GPS.

```
import paho.mqtt.client as mqtt
import time
import serial
from cryptography.fernet import Fernet
```

Selanjutnya menentukan data GPS yang akan dikirim, membuat koneksi dengan broker, memasukkan kunci private untuk enkripsi, kemudian dipublish dengan topik “OBU/GPS”

```
serialStream = serial.Serial("/dev/ttyAMA0", 9600, timeout=0.5)
mqttc = mqtt.Client()
mqttc.connect("vps2.lawanghosting.pw", 1883, 60)
mqttc.loop_start()
key = b'8I7nMN_-ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
cipher = Fernet(key)

try:
   while True:
      sentence = serialStream.readline()
      if sentence.find('GGA') > 0:
         encrypt = cipher.encrypt(sentence.encode('utf-8'))
         print sentence
         mqttc.publish("OBU/GPS", encrypt)
except KeyboardInterrupt:
   streamer.close()
```

Setelah program publish dibuat, maka dibuatlah program untuk subscribe topik "OBU/GPS" yang kemudian akan mengirimkan data-data GPS yang diterima ke database untuk melakukan perekaman. Selain disimpan di database, data lokasi dari GPS juga akan diteruskan menuju program TMC agar bisa dilakukan monitoring lokasi armada secara *real time* melalui video wall. Berikut ini adalah program untuk subscribe GPS yang diteruskan menuju database.

```
import paho.mqtt.client as mqtt
from cryptography.fernet import Fernet
import com_sensor
#
def on_connect(client, userdata, flags, rc):
        print("Connected with result code "+str(rc))
        client.subscribe(topic="OBU/GPS")
        client.subscribe(topic="OBU/#")

#
def on_message(client, userdata, msg):
        key = b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
        cipher = Fernet(key)
        decrypted = cipher.decrypt(msg.payload)
        pay = decrypted.decode("utf-8")
        pay_out=u"'"+pay+"'"
        print("on_message:topic=" + msg.topic+" ,pay="
+str(pay_out))
        if len(pay_out)>10 and len(pay_out)<256:
                clsSen=com_sensor.sensorClass()
                clsSen.saveSensor( msg.topic, pay_out)
```

```
if __name__ == "__main__":
    client = mqtt.Client(client_id="spam")
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(host="185.201.9.130", port=1883)
    client.loop_forever()
```

Untuk bisa mengakses database yang akan digunakan untuk menyimpan data GPS, harus dilakukan setting terlebih dahulu dan menentukan database yang akan digunakan.

```
# -*- coding: utf-8 -*-

#com_appConst
class appConstClass:
    def __init__(self):
        print ""

    mHost = "localhost"
    mDB_NAME= "db_ITS_2018"
    mUser ="root"
    mPass ="aj404its!@#$%"
    mOK_CODE=1
    mNG_CODE=0

    def test(self, sId):
            print "test"
```

Setelah itu juga membuat program konfigurasi dan SQL untuk menghubungkan MQTT dengan database agar bisa memasukkan data-data lokasi yang diterima ke dalam tabel yang ada di dalam database yang dibuat.

```
# -*- coding: utf-8 -*-
import MySQLdb
import com_appConst

#com_sensor
class sensorClass:

   def __init__(self):
      print ""
      def saveSensor(self, topic, payload):
            ret=False
            clsConst = com_appConst.appConstClass()
            connection = MySQLdb.connect(host=clsConst.mHost,
db=clsConst.mDB_NAME, user=clsConst.mUser,
passwd=clsConst.mPass, charset="utf8")
            cursor = connection.cursor()
            sSql=u"INSERT INTO GPS_Log (serID, topic,
payload,created_at)"
            sSql=sSql+" values ("
            sSql=sSql+"serID"+","
            sSql=sSql+"'"+topic +"'"
            sSql=sSql+","+ payload
            sSql=sSql+",now() );"
            #print sSql
            cursor.execute(sSql)
            connection.commit()
            cursor.close()
            connection.close()
            ret=True
            return ret
```

Untuk mengirimkan database ke aplikasi TMC, maka terlebih dahulu data lokasi dari GPS di salurkan melalui UDP agar sampai pada aplikasi TMC.

```
import paho.mqtt.client as mqtt
import socket
import time
from cryptography.fernet import Fernet

UDP_IP = "127.0.0.1"
UDP_PORT = "50000"
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

def on_message(client, userdata, message):
   key = b'8I7nMN_-ShTaSBOzZL4DWC66slvlYL9xUzeDiNIQ3Gg='
   cipher = Fernet(key)
   decrypted_gps = cipher.decrypt(message.payload)
   str1 = str(decrypted_gps.decode('utf-8'))
   print "msg>" , str1
   sock.sendto(str1+' -$ $', (UDP_IP, UDP_PORT))
subscribe.callback (on_message, "OBU/GPS", hostname =
"185.201.9.130")
```

3.5 Pengujian Sistem

Pada bagian ini akan dibahas mengenai data yang akan digunakan dalam pengujian sistem keamanan jaringan antara OBU dan TMC server (berbasis protokol MQTT) yang telah dirancang serta skenario pengujiannya. Pengujian akan dibagi menjadi tiga bagian, yakni : pengujian program enkripsi-dekripsi, pengujian program enkripsi tanpa dekripsi, dan pengujian jaringan. Masing-masingnya akan dijelaskan sebagai berikut.

3.5.1 Data Pengujian

Pada pengujian ini, terdapat tiga jenis data yang akan digunakan. Data-data tersebut kemudian akan disebut sebagai Data 1, Data 2, dan Data 3 yang dibedakan menurut jumlah bit dari isi pesannya. Isi pesan dalam data dapat dilihat pada Tabel 3.1. Data-data tersebut kemudian akan digunakan sebanyak masing-masing 3 kali sesuai dengan skenario yang akan dijelaskan pada subbab 3.5.2.

Tabel 3.1Isi Pesan Pengujian

No.	Data	Isi Pesan	Jumlah bit
1	Data 1	Dear Mr Shopaholic, please order a typewriter. Regards Honest John	**640**
2	Data 2	Starting example for the CrypTool version family 1.x (CT1) CrypTool 1 (CT1) is a comprehensive and free educational program about cryptography and cryptanalysis offering extensive online help and many visualizations. This text file was created in order to help you to make your first steps with CT1. 1) The starting page of the online help offers the best oversight of CT1's capacity. From the starting page you can reach all essential functions via links. The starting page of the online help can be accessed via the menu "Help -> Starting Page" at the top right of the main window or by using the search keyword "Starting page" within the index of the online help. Press F1 to start the online help everywhere in CT1. 2) A possible next step would be to encrypt a file with the Caesar algorithm. This can be done via the menu "Crypt/Decrypt ->	**12.872**

No.	Data	Isi Pesan	Jumlah bit
		Symmetric (classic)". 3) There are several examples (tutorials) within the online help which provide an easy way to gain an understanding of cryptology. These examples can be found via the menu "Help -> Scenarios (Tutorials)".	
3	Data 3	%% Plotting Sensitivities of a Portfolio of Options % This example plots gamma as a function of price and time for a portfolio % of 10 Black-Scholes options. % The plot in this example shows a three-dimensional surface. For each % point on the surface, the height (_z_-value) represents the sum of the % gammas for each option in the portfolio weighted by the amount of each % option. The _x_-axis represents changing price, and the _y_-axis represents % time. The plot adds a fourth dimension by showing delta as surface color. % This example has applications in hedging. First set up the portfolio with arbitrary % data. Current prices range from $20 to $90 for each option. Then, set the corresponding % exercise prices for each option. % Copyright 2015 The MathWorks, Inc. Range = 20:90; PLen = length(Range); ExPrice = [75 70 50 55 75 50 40 75 60 35]; % Set all risk-free interest rates to 10%, and set times to maturity in % days. Set all volatilities to 0.35. Set the number of options of each % instrument, and allocate space for matrices. Rate = 0.1*ones(10,1);	**21.488**

No.	Data	Isi Pesan	Jumlah bit
		Time = [36 36 36 27 18 18 18 9 9 9]; Sigma = 0.35*ones(10,1); NumOpt = 1000*[4 8 3 5 5.5 2 4.8 3 4.8 2.5]; ZVal = zeros(36, PLen); Color = zeros(36, PLen);	

Adapun parameter yang akan diukur dalam tugas akhir ini adalah delay pengiriman pesan, throughput, dan tingkat keamanan informasi yang dikirimkan yang selanjutnya akan dianalisis dan diambil kesimpulan.

3.5.2 Skenario Pengujian

Pengujian akan dibagi menjadi tiga bagian, yakni : pengujian pesan original, pengujian enkripsi tanpa dekripsi, dan pengujian enkripsi – dekripsi, masing-masing akan dilihat delay, throughput, dan keamanan pengiriman pesan.

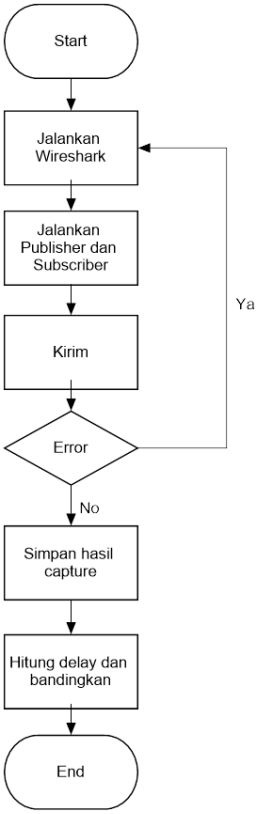
Tabel 3.2 Skenario pengujian dan parameter pengukuran

No	Keterangan	Parameter
1	Skenario Pengujian	Pengujian pesan original
		Pengujian enkripsi tanpa dekripsi
		pengujian enkripsi - dekripsi
2	Parameter pengukuram	Delay pengiriman
		Throughput
		Tingkat Keamanan

3.5.2.1 Pengujian Pesan Original

Pengujian pertama adalah pengujian dengan mengirimkan pesan original dari publisher menuju subscriber. Data-data dikirimkan dari publisher menuju subscriber dan akan diamati karakteristik dari parameter-parameter pengukuran yang diinginkan menggunakan aplikasi Wireshark..

Alur pengujian data pesan dapat dilihat pada Gambar 3.9, yang mana pengujian tersebut akan diulang-ulang sebanyak 3 kali sesuai dengan data pada Tabel 3.1 dan akan dilakukan pengukuran selama 1 menit dengan selang waktu data pengiriman setiap 5 detik.



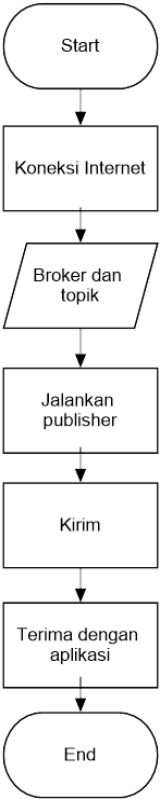
Gambar 3.9 Skenario Pengujian pesan original

3.5.2.2 Pengujian Enkripsi Tanpa Dekripsi

Tujuan dari pengujian ini adalah untuk melihat apakah program enkripsi benar-benar berjalan dan publisher hanya mengirim *ciphertext*. Untuk itu dilakukan pengujian dengan

menjalankan publisher yang telah ditambahkan script enkripsi dan subscriber yang tidak memiliki script dekripsi.

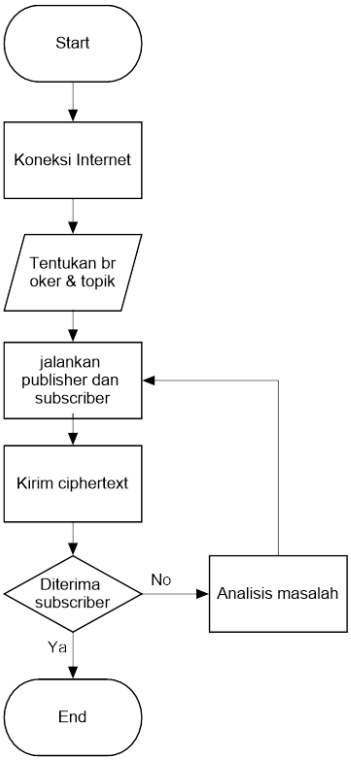
Alur pengujian data pesan dapat dilihat pada Gambar 3.10, yang mana sama seperti pengujian sebelumnya, pengujian kali ini juga akan diulang-ulang sebanyak 3 kali sesuai dengan data pada Tabel 3.1 dan akan dilakukan pengukuran selama 1 menit dengan selang waktu data pengiriman setiap 5 detik.



Gambar 3.10 Skenario Pengujian enkripsi tanpa dekripsi

3.5.2.3 Pengujian Enkripsi - Dekripsi

Bentuk pengujian ketiga adalah dengan mengirimkan pesan yang melalui proses enkripsi pada publisher dan diikuti proses dekripsi di sisi subscriber, dimana akan dibuktikan apakah program enkripsi berhasil dijalankan serta apakah program dekripsi berhasil mendapatkan kembali *plaintext* / pesan original dari *ciphertext* yang diterima. Untuk itu, digunakan publisher yang telah ditambahkan script enkripsi agar mengirim data pada subscriber yang juga memiliki script dekripsi. Parameter pengujian lainnya juga akan didapatkan dengan menjalankan aplikasi wireshark selama proses pengiriman pesan.



Gambar 3.11 Skenario Pengujian Delay

Alur pengujian data pesan dapat dilihat pada Gambar 3.11, yang mana sama seperti pengujian sebelumnya, pengujian tersebut akan diulang-ulang sebanyak 3 kali sesuai dengan data yang ada pada Tabel 3.1 serta pengukuran yang akan dilakukan selama 1 menit dengan selang waktu data pengiriman setiap 5 detik.

Halaman ini mengandung kekosongan

BAB 4 PENGUJIAN DAN ANALISIS DATA

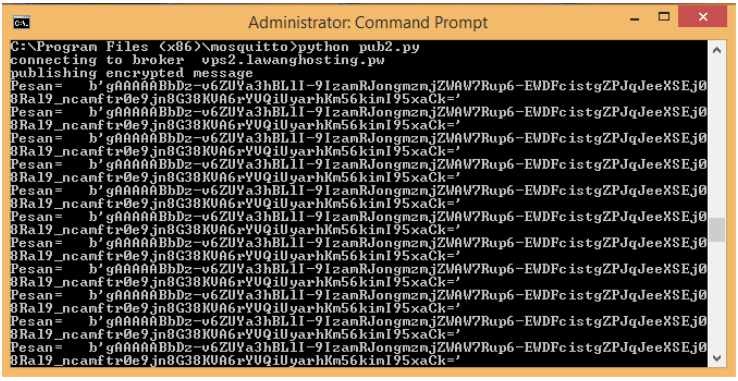
Pada bab ini akan diperlihatkan hasil pengujian dan analisis dari sistem keamanan yang telah dirancang untuk jaringan komunikasi ITS sebagaimana dijelaskan pada bab sebelumnya.

4.1 Hasil Pengujian Keberhasilan Sistem

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah sistem keamanan yang sudah dibangun sudah berjalan dengan benar serta memastikan bahwa data yang dikirim tidak bisa dibaca oleh pihak yang tidak memiliki otoritas akses.

4.1.1 Hasil Enkripsi Sisi Publisher

Pengujian pertama dilakukan dengan menjalankan program enkripsi pada bagian publisher dan mempublish data. Script program yang ada di sisi publisher akan membuat data terenkripsi dahulu sebelum dikirim menuju subscriber. Data dienkripsi dengan algoritma AES-128-CBC menggunakan modul fernet yang dituliskan pada script publisher sebelum dikirimkan menuju broker.

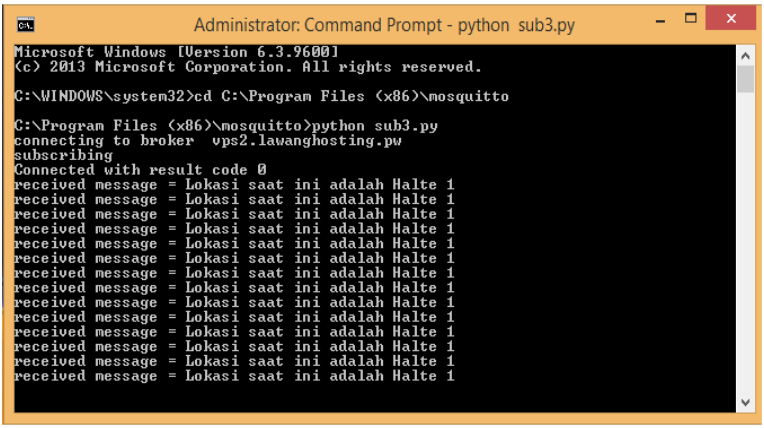


Gambar 4.1 Hasil enkripsi pada publisher

Seperti yang terlihat pada Gambar 4.1, program enkripsi yang dijalankan telah berhasil mengubah pesan "Lokasi saat ini adala Halte1" yang akan dikirim menjadi *ciphertext* sehingga tidak bisa dibaca secara langsung.

4.1.2 Hasil Dekripsi Sisi Subscriber

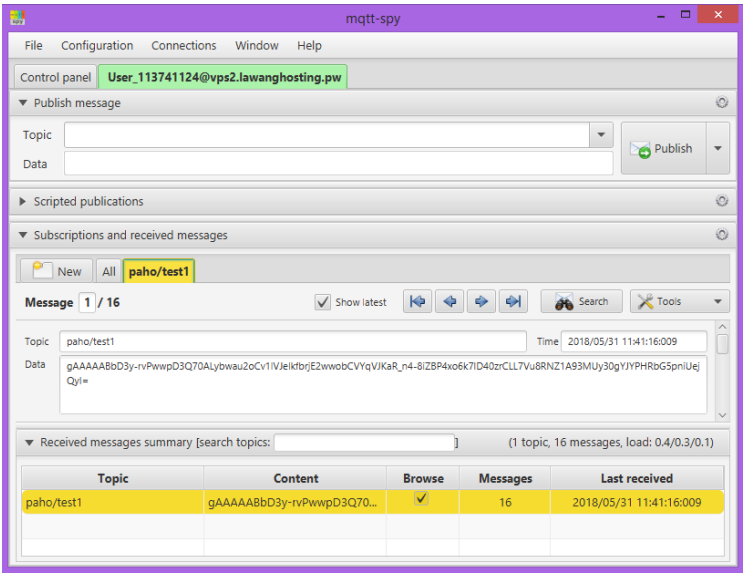
Di sisi subscriber yang telah ditambahkan program untuk dekripsi akan menampilkan kembali data *plaintext* yang dikirim oleh publisher. Pesan didekripsi juga menggunakan algoritma AES-128-CBC, agar bisa menampilkan kembali pesan yang telah dikirimkan. Adapun contoh pesan yang dicoba adalah "Lokasi saat ini adalahh Halte 1" dan hasilnya dapat dilihat pada Gambar 4.2 .



Gambar 4.2 Hasil Dekripsi di sisi subscriber

4.1.3 Hasil Tanpa Dekripsi Sisi Subscriber

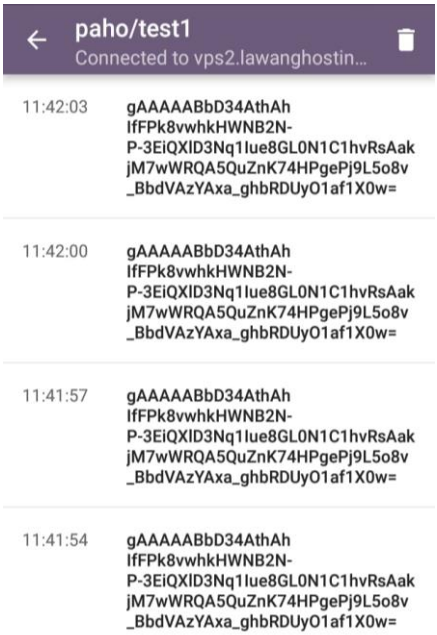
Untuk memeriksa bahwa data yang terkirim telah terenkripsi, maka dilakukan pengujian dengan mengirimkan pesan yang sama seperti sebelumnya ke aplikasi yang tidak didukung dengan program dekripsi. Dua aplikasi yang digunakan untuk pengujian ini adalah mqtt spy yang ada di komputer dan MQTT Dashboard yang diinstall di smartphone.



Gambar 4.3 tampilan pesan pada mqt spy masih berbentuk ciphertext.

Dari Gambar 4.3 diatas, dapat dilihat bahwa data yang diterima oleh mqtt spy bersifat acak karena merupakan *ciphertext* dari data asli sehingga tidak bisa dimengerti. Hal ini dikarenakan mqtt spy tidak dilengkapi dengan kunci simetris yang bisa digunakan untuk mendekripsi pesan sebagaimana yang digunakan untuk mengenkripsi pesan.

Gambar 4.4 berikut merupakan hasil tangkapan layar dari data kiriman publisher yang berhasil diterima oleh aplikasi MQTT Dashboard yang ada pada smartphone. Karena pada smartphone tidak dilengkapi dengan program dekripsi, maka data yang diterima merupakan cipertext dan tidak bisa dibaca. Namun masih bisa dilihat kalau *ciphertext* yang diterima kedua aplikasi masih sama dengan *ciphertext* yang dihasilkan publisher.

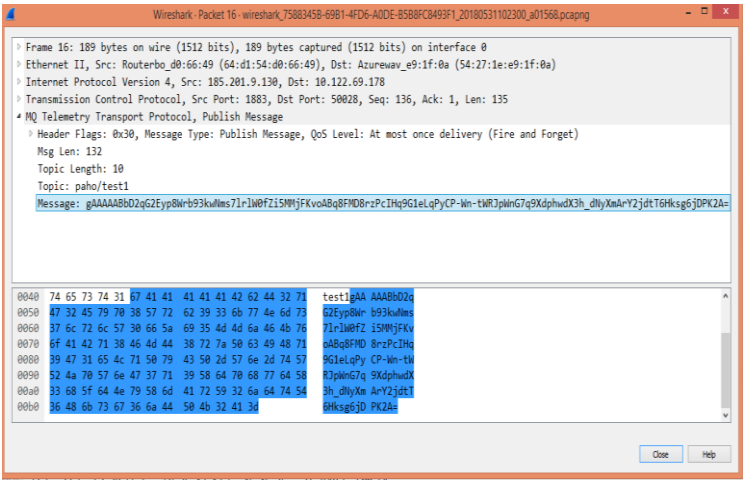


Gambar 4.4 Tampilan pesan pada aplikasi MQTTDashboard

Dari hasil tangkapan dua aplikasi di atas, dapat disimpulkan bahwa program enkripsi yang digunakan sudah bekerja sesuai dengan ekspektasi, dimana program atau aplikasi yang tidak memiliki program dekripsi dan kunci simetris tidak akan bisa membaca pesan asli yang dikirimkan.

4.1.4 Hasil Capture Pesan dengan Wireshark

Untuk melihat apakah data yang dikirim benar-benar terenkripsi dan tidak bisa terbaca saat melewati jaringan, maka dilakukan pengujian dengan aplikasi wireshark. Pengujian ini juga menggunakan pesan seperti pengujian sebelumnya dimana wireshark akan menangkap dan menampilkan pesan atau paket-paket yang sedang dikirimkan melaui jaringan.



Gambar 4.5 Hasil tangkapan pesan dengan Wireshark

Setelah wireshark dijalankan, dapat dilihat dari hasil capture bahwa data yang terkirim sudah terenkripsi dan tidak bisa terbaca dari wireshar. Hal ini dapat dilihat dari hasil pesan yang tidak beraturan.

4.2 Hasil Pengujian Performansi Sistem

Setelah sistem keamanan pada ITS bisa dijalankan dan selesai diuji, selanjutnya adalah melakukan pengujian sesuai dengan skenario-skenario yang telah dijelaskan pada subbab 3.5.

4.2.1 Hasil Pengiriman Pesan Original

Tabel 4.1Hasil Performansi Pesan Original

No	Data	Delay (s)	Throughput (bps)	Tkt. Keamanan
1	Data 1	0.00457717	1110	0%
2	Data 2	0.0072815	1661	0%
3	Data 3	0.007801	124100	0%
Rata-rata		**0.006553**	**11764.667**	**0%**

Setelah dilakukan pengujian dengan skenario 3.5.2.1 terhadap Data 1, Data 2, dan Data 3 menggunakan Wireshark, diperoleh hasil pengukuran yang berisikan waktu pengiriman dan throughput. Selanjutnya dari masing-masing hasil tersebut dihitung nilai rata-rata untuk setiap data, yang kemudian dari Data 1,2,dan 3 dihitung lagi rata-ratanya untuk mengetahui Delay pengiriman, throughput, dan tingkat keamanan secara umum. Hasilnya dapat dilihat pada tabel diatas. Untuk lebih lengkapnya, data hasil pengukuran skenario 3.5.2.1 dapat dilihat pada Lampiran C1, C2, dan C3.

4.2.2 Hasil Pengiriman Enkripsi tanpa Dekripsi

Tabel 4.2Hasil Performansi Enkripsi tanpa Dekripsi

No	Data	Delay (s)	Throughput (bps)	Tkt. Keamanan
1	Data 1	0.0062193	3429	100%
2	Data 2	0.0076249	19592	100%
3	Data 3	0.0082176	27608	100%
Rata-rata		**0.007354**	**16876.33**	**100%**

Setelah dilakukan pengujian dengan skenario 3.5.2.2 terhadap Data 1, Data 2, dan Data 3 menggunakan Wireshark, diperoleh hasil pengukuran yang berisikan waktu pengiriman dan throughput. Selanjutnya dari masing-masing hasil tersebut dihitung nilai rata-rata untuk setiap data, yang kemudian dari Data 1,2,dan 3 dihitung lagi rata-ratanya untuk mengetahui Delay pengiriman, throughput, dan tingkat keamanan secara umum. Hasilnya dapat dilihat pada tabel diatas. Untuk lebih lengkapnya, data hasil pengukuran skenario 3.5.2.2 dapat dilihat pada Lampiran C4, C5, dan C6.

4.2.3 Hasil Pengiriman Enkripsi - Dekripsi

Tabel 4.3 Hasil Performansi Enkripsi-Dekripsi

No	Data	Delay (s)	Throughput (bps)	Tkt. Keamanan
1	Data 1	0.006638	5599	100%

2	Data 2	0.0092338	19616	100%
3	Data 3	0.0098668	31976	100%
Rata-rata		**0.00858**	**19063.667**	**100%**

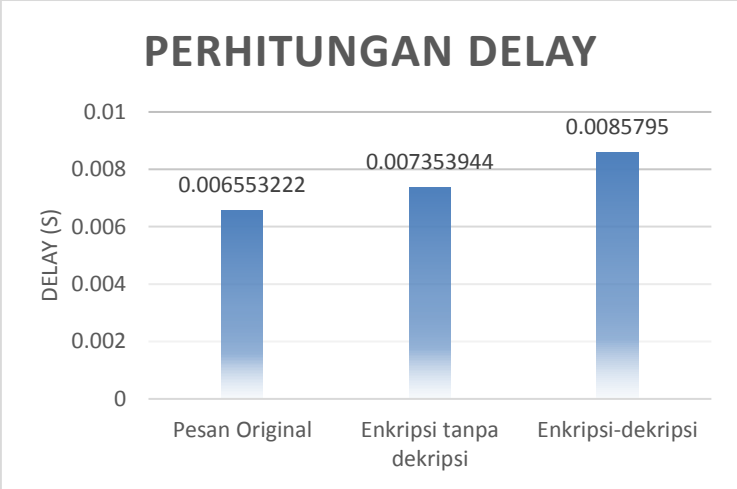
Setelah dilakukan pengujian dengan skenario 3.5.2.3 terhadap Data 1, Data 2, dan Data 3 menggunakan Wireshark, diperoleh hasil pengukuran yang berisikan waktu pengiriman dan throughput. Selanjutnya dari masing-masing hasil tersebut dihitung nilai rata-rata untuk setiap data, yang kemudian dari Data 1,2,dan 3 dihitung lagi rata-ratanya untuk mengetahui Delay pengiriman, throughput, dan tingkat keamanan secara umum. Hasilnya dapat dilihat pada tabel diatas. Untuk lebih lengkapnya, data hasil pengukuran skenario 3.5.2.3 dapat dilihat pada Lampiran C7, C8, dan C9.

4.3 Analisis dan Pembahasan

Setelah data-data didapatkan dari pengujian yang dilakukan, disini dianalisis serta dibahas mengenai perilaku atau karakteristik data terhadap parameter-parameter yang diukur.

4.3.1 Pengaruh Enkripsi - Dekripsi terhadap Delay

Dari data-data yang diperole pada bagian hasil pengujian di subbab 4.2, dapat dilihat bahwa untuk delay pada pengiriman pesan original sebesar 0.006553 s, untuk pengiriman Enkripsi tanpa Dekripsi sebesar 0.007354 s, dan untuk pengiriman dengan enkripsi dan enkripsi sebesar 00858 s. Maka dari itu dapat dilihat bahwa pesan yang dikirimkan melalui proses enkripsi dan diikuti dengan dekripsi memiliki delay pengiriman paling besar. Dengan lebih jelasnya dapat dilihat pada grafik hasil perbandingan berikut pada gambar 4.6 .

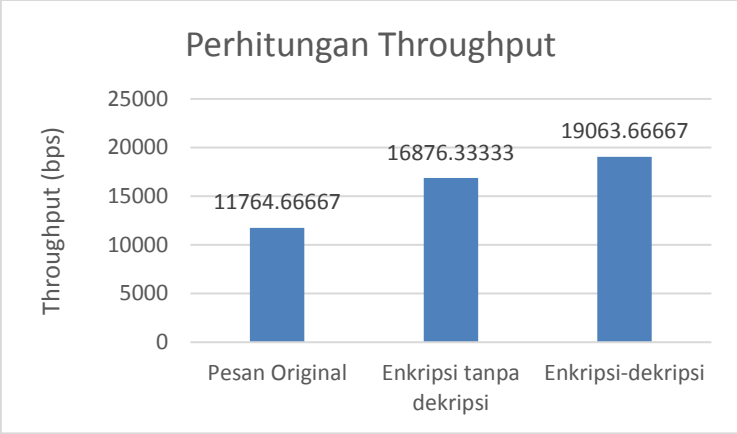


Gambar 4.6 Grafik Perhitungan delay

Hal tersebut disebabkan karena pesan yang dikirimkan mendapatkan tambahan bit-bit yang diperoleh dari prosen enkripsi dengan AES-128-CBC dan HMAC SHA 256, sehingga pesan terkirim menjadi lebih panjang dari pesan aslinya.

4.3.2 Pengaruh Enkripsi - Dekripsi terhadap Throughput

Dari data-data yang diperole pada bagian hasil pengujian di subbab 4.2, dapat dilihat bahwa untuk throughput pengiriman pesan original sebesar 11764.67 bps, untuk pengiriman Enkripsi tanpa Dekripsi sebesar 16876.33 bps, dan untuk pengiriman dengan enkripsi dan enkripsi sebesar 19063.667 bps. Maka dari itu dapat dilihat bahwa pesan yang dikirimkan melalui proses enkripsi dan diikuti dengan dekripsi memiliki throughput paling besar. Untuk lebih jelasnya dapat dilihat pada grafik hasil perhitungan Throughput berikut pada gambar 4.7.



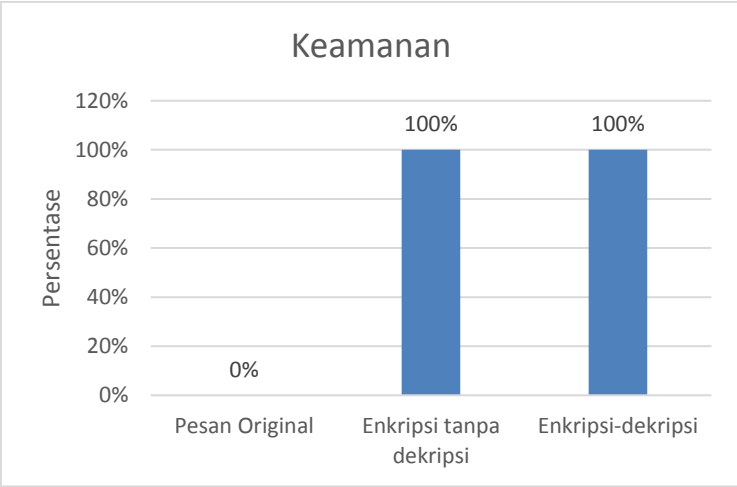
Gambar 4.7 Grafik Perhitungan Throughput

Hal tersebut disebabkan karena pesan yang dikirimkan mendapatkan tambahan bit-bit dari proses enkripsi dengan AES-128-CBC dan HMAC SHA 256, sehingga banyaknya pesan yang terkirim menjadi lebih besar dari pesan aslinya.

4.3.3 Pengaruh Enkripsi - Dekripsi Keamanan Pesan

Dari data-data yang diperole pada bagian hasil pengujian di subbab 4.2, dapat dilihat bahwa tingkat keamanan pengiriman pesan original sebesar 0%, untuk pengiriman Enkripsi tanpa Dekripsi sebesar 100%, dan untuk pengiriman dengan enkripsi dan enkripsi sebesar 100%. Maka dari itu dapat dilihat bahwa pesan yang dikirimkan melalui proses enkripsi memiliki tingkat keamanan 100%. Untuk lebih jelasnya dapat dilihat pada grafik hasil perhitungan Throughput berikut pada gambar 4.8 .

Namun terdapat perbedaan pada tingkat keamanan pada pengiriman Enkripsi tanpa Dekripsi dan pengiriman Enkripsi – Dekripsi, yang mana pada pengiriman pesan dengan Enkripsi namun tidak diikuti dengan Dekripsi, informasi yang diterima di sisi subscriber sudah aman namun tidak akan bisa dibaca karena masih berupa *ciphertext.*



Gambar 4.8 Persentase Keamanan

4.3.4 Potensi Ancaman Keamanan Pesan

Setelah sistem berhasil diamankan, maka disini saya melakukan analisis dengan membandingkan kondisi-kondisi sebelum dan sesudah dilakukan pengamanan dari segi ancaman kejahatan dan aspek keamanan yang ada.

Tabel 4.4 Analisis teradap keamanan sistem

No.	Aspek Keamanan	Ancaman	Hasil
1	*Confidentiality*	*Eavesdropping*	Dengan adanya enkripsi AES-128-CBC, attacker sudah tidak bisa lagi membaca data-data yang dikirimkan dari subscriber

No.	Aspek Keamanan	Ancaman	Hasil
		Brute force	Dengan panjang kunci 128 bit, akan membutukan waktu 5.4 x 10^{18} tahun untuk menemukan kunci yang benar
2	*Integrity*	Modifikasi data	Dengan HMAC SHA-256, perubahan pada data akan terdeteksi dan tidak diterima oleh subscriber
		Collision (bentrok)	Hampir mustahil untuk menemukan nilai hash yang sama dengan SHA-256.

Halaman ini mengandung kekosongan

BAB 5 PENUTUP

5.1 Kesimpulan

Setelah dilakukan perancangan dan pengujian sistem keamanan pada jaringan komunikasi ITS Surabaya, maka secara keseluruhan dapat diambil kesimpulan sebagai berikut:

1. Semakin panjang bit pesan yang dikirimkan berpengaruh terhadap delay pengiriman dan throughput yang juga semakin besar.
2. Dengan adanya proses enkripsi dan/atau dekripsi akan berdampak pada penambahan delay dan throughput yang semakin besar.
3. Dengan penggunaan enkripsi dan dekripsi AES-128-CBC dan HMAC SHA256, akan meningkatkan keamanan pengiriman informasi dibandingkan dengan 'pesan original' yang mana telah terlindungi dari potensi serangan terhadap faktor *confidentiality* (kerahasiaan) dan *integrity*.
4. Dari keseluruhan pengujian yang dilakukan dalam tugas akhir ini, dapat disimpulkan bahwa untuk masing-masing skenario pengukuran didapatkan delay rata-rata adalah 0.006553 s, 0,007354, dan 0,00858 s. Hasil tersebut tergolong sangat bagus karena jauh dibawah ketentuan maksimal yang bernilai 150 ms.
5. Dengan mengirimkan pesan dengan panjang 640 bit, 12.872 bit, dan 21.488 bit, diperoleh nilai throughput berturut-turut adalah 11764.67 bps, 16876.33 bps, dan 19063.667 bps.
6. Dengan skenario pengujian yang dilakukan, didapatkan persentase tingkat keamanannya adalah 0%, 100%, 100%. Namun pada pengiriman pesan dengan enkripsi saja tanpa proses dekripsi, pesan yang diterima di sisi subscriber tidak bisa dibaca karena berupa *ciphertext*.

5.2 Saran

Adapun hal-hal yang masih bisa dikembangkan lagi dari tugas akhir ini untuk penelitian berikutnya antara lain adalah:

1. Untuk meningkatkan keamanan pada jaringan komunikasi ITS, bisa dilakukan dengan mengunakan metode lain seperti memasang TLS/SSL.
2. Enkripsi pada data-data yang akan dikirim bisa dicoba menggunakan algoritma lain selain AES-128-CBC seperti RSA atau *Digital Signature.*

DAFTAR PUSTAKA

[1] W. Stallings, *C Ryptography and*. 2017.

[2] M. A. Yasir, "Rancang Bangun Sistem Perpesanan OBU(on Board Unit) Untuk Intelligent Transport System Di Surabaya," 2018.

[3] Information Resources Management Association (IRMA), *Green Technologies: Concepts, Methodologies, Tools and Applications ... - Google Buku*. New York: IGI Global, 2011.

[4] P. Yuliantoro, "Rancang bangun protokol e-ticketing," 2015.

[5] A. Affandi, S. D. Rahardjo, and I. Pratomo, "PENGEMBANGAN PERANGKAT ON BOARD UNIT DAN FITUR VARIABLE MESSAGING SYSTEM UNTUK MENDUKUNG IMPLEMENTASI SISTEM TRANSPORTASI CERDAS," 2017.

[6] I. K. E. Purnama, A. Affandi, G. Kusrahardjo, and M. Ardita, "Pengembangan Tesbed Traffic Management Center (TMC) untuk Pemantaunan dan Pengaturan Transportasi Publik di Pusat Studi TIK dan Robotika," 2017.

[7] K. Finkenzeller, *RFID Handbook*, Third. Wiltshire: Wiley, 2010.

[8] M. Bani Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, "Internet of Things: Survey and open issues of MQTT Protocol," *Ieee*, pp. 1–6, 2017.

[9] J. Velez, R. Trafford, M. Pierce, B. Thomson, E. Jastrzebski, and B. Lau, "IEEE 1451-1-6 : Providing Common Network Services over MQTT," 2018.

[10] S. N. Firdous, Z. Baig, C. Valli, and A. Ibrahim, "Modelling and Evaluation of Malicious Attacks against the IoT MQTT Protocol," *2017 IEEE Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data*, pp. 748–755, 2017.

[11] OASIS, "MQTT Version 3.1.1," *OASIS Stand.*, no. October, p. 81, 2014.

[12] "How to Install and Secure the Mosquitto MQTT Messaging Broker on Ubuntu 16.04 | DigitalOcean." [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-the-mosquitto-mqtt-messaging-broker-on-

ubuntu-16-04#step-1-—-installing-mosquitto. [Accessed: 18-May-2018].

[13] OASIS, "MQTT and the NIST Cybersecurity Framework Version 1.0," *OASIS Comm. Note 01*, no. May, 2014.

[14] "MQTT Security Fundamentals: Payload Encryption." [Online]. Available: https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption. [Accessed: 01-Jul-2018].

[15] R. O. N. Ross and R. Mcquaid, "Draft SP 800-160 Vol. 2, Systems Security Engineering: Cyber Resiliency Considerations for the Engineering of Trustworthy Secure Systems," vol. 2.

[16] G. R. Karsanbhai and M. G. Shajan, "AES Algorithm for Secured Wireless Communication," no. May, 2011.

[17] N. Fips, "197: Announcing the advanced encryption standard (AES)," *... Technol. Lab. Natl. Inst. Stand. ...*, vol. 2009, pp. 8–12, 2001.

[18] "Fernet (symmetric encryption) — Cryptography 2.3.dev1 documentation." [Online]. Available: https://cryptography.io/en/latest/fernet/. [Accessed: 01-Jul-2018].

LAMPIRAN A

PROPOSAL TUGAS AKHIR

Departemen Teknik Elektro
Fakultas Teknologi Elektro - ITS

TE 141599 TUGAS AKHIR – 4 SKS

09 FEB 2018

Nama Mahasiswa	: Diana Musabbihah
Nomor Pokok	: 07111440007004
Bidang Studi	: Telekomunikasi Multimedia
Tugas Diberikan	: Semester Genap 2017/ 2018
Dosen Pembimbing	: 1. Dr. Ir. Achmad Affandi, DEA 2. Ir. Djoko Suprajitno Rahardjo, MT.
Judul Tugas Akhir	: **Perencanaan Sistem Keamanan pada Jaringan Komunikasi ITS (Intelligent Transportation System) Antara OBU dan TMC Server.** ***(Development of Security System for Wireless Communication of ITS (Intelligent Transportation System Between OBU and TMC Server)***

Uraian Tugas Akhir :

Kemajuan teknologi informasi dan komunikasi (TIK) sangatlah berperan penting dalam perkembangan *Intelligent Transport System* (ITS) atau yang juga disebut dengan Sistem Transportasi Cerdas. Banyak negara yang sedang melakukan penelitian untuk meningkatkan performansi sistem transportasi. Salah satu bentuk penelitian yang telah dikembangkan untuk meningkatkan performansi tersebut adalah dengan dibuatnya komponen *on-board unit* (OBU) sebagai pusat pengambilan informasi di armada yang kemudian akan diolah oleh *Traffic Managenment System* (TMC) serta dengan adanya fitur *Variable Messaging System* (VMS) yang memungkinkan terjadinya perpesanan dari sisi *client* dan sisi server. Karena komunikasi yang terjadi di antara kedua sisi dilakukan pada jalur internet publik, maka timbul permasalahan-permasalahan ditinjau dari segi keamanan. Diantaranya adalah bagaimana menjaga integritas dari data-data yang dikirim serta menjamin ketersediaan sistem keamanan tersebut agar selalu berjalan setiap saat akibat dari mobilitas OBU yang berada di dalam armada. Oleh karena itu dibutuhkan sebuah protokol keamanan untuk melindungi semua data dan informasi vital yang dikirimkan dari sisi OBU maupun sisi TMC server kapanpun dan di manapun armada berada, serta dapat menjamin availabilitas dari sistem keamanan yang dibangun sehingga semua transaksi dapat berjalan dengan lancar dan aman.

Dosen Pembimbing I,

Dr. Ir. Achmad Affandi, DEA
NIP. 196510141990021001

Dosen Pembimbing II,

Ir. Djoko Suprajitno Rahardjo, M.T.
NIP. 195506221987011001

Mengetahui,
Ketua Program Studi S1

Dedet C. Riawan, ST., M.Eng., Ph.D
NIP. 197311192000031001

Menyetujui,
Kepala Laboratorium Jaringan Telekomunikasi

Dr. Ir. Achmad Affandi, DEA
NIP. 196510141990021001

USULAN TUGAS AKHIR

A. JUDUL TUGAS AKHIR

Perencanaan Sistem Keamanan pada Jaringan Komunikasi ITS (*Intelligent Transportation System*) Antara OBU dan TMC Server.

B. RUANG LINGKUP

1. *Security Policy*
2. Enkapsulasi paket data
3. *Tunneling*
4. Enkripsi Data

C. LATAR BELAKANG

Untuk menangani kemacetan yang terjadi di kota-kota besar, pemerintah saat ini sangat gencar dalam membangun fasilitas transportasi publik yang memadai. Pada era digital saat ini, sistem transportasi modern yang dikenal dengan sistem transportasi cerdas atau *Intelligent Transportation System* (ITS) sangat membutuhkan penerapan teknologi informasi dan telekomunikasi, di antaranya adalah dalam sistem komunikasi data yang digunakan pada jaringan ITS. Setiap data yang digunakan dalam transaksi pada ITS seperti data diri penumpang, lokasi armada, status armada hingga data saldo dari penumpang yang dikirim dari sisi OBU ke server TMC harus melewati jaringan internet untuk kemudian diolah.

Salah satu aspek penting yang harus diperhatikan dalam sistem komunikasi data ini adalah dari segi keamanannya. Demi efektivitas infrastruktur jaringan, pada sistem ITS digunakan jaringan internet publik untuk transportasi data yang dikirim dari OBU ke TMC server. Dengan internet publik yang bisa diakses oleh siapapun, setiap orang yang terhubung dengannya akan dapat mengakses data-data dan trafik yang melintas termasuk data dari transaksi ITS.

Oleh karena itu diperlukan suatu protokol keamanan yang dapat menjaga integritas data yang dikirimkan dari sisi OBU menuju TMC server agar tidak ada pihak yang dapat melihat ataupun menginterferensi data yang melintas dalam jaringan internet publik. Aspek lain dalam network security yang perlu diperhatikan pada jaringan ITS ini adalah availability, yakni ketersediaan informasi dimana data atau infomasi yang dikirim oleh OBU dapat tersampaikan ke TMC server. Maka dari itu, pada tugas akhir ini akan dilakukan perancangan protokol keamanan pada jaringan ITS yang akan menjamin integritas data yang dikirim dari sisi *client* melalui internet publik menuju server dan sebaliknya, serta dapat memastikan bahwa data yang terkirim benar-benar telah sampai di TMC server sehingga data dapat dikelola dengan aman.

D. RUMUSAN MASALAH

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimanakah sistem keamanan yang tepat untuk digunakan dalam jaringan ITS pada jaringan internet publik.
2. Bagaimana cara menjaga agar sistem keamanan antara OBU dan TMC server tetap berjalan meskipun terjadi perubahan IP adrress akibat mobilitas OBU di dalam armada.
3. Bagaimana *Security Parameter Index* yang tepat sebagai autentikasi paket data yang dikirimkan dalam jaringan ITS.

E. BATASAN MASALAH

Batasan masalah dari Tugas Akhir ini adalah sebagai berikut.

1. Fokus pada keamanan jaringan IP publik
2. Digunakan pada jaringan komunikasi antara OBU dan server TMC

F. TUJUAN TUGAS AKHIR

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Terdapat sebuah protokol keamanan jaringan yang dapat menjamin keamanan data transaksi antara sisi OBU dan TMC server
2. Menjamin availabilitas sistem keamanan pada jaringan ITS akibat mobilitas OBU di dalam armada
3. Didapatkan *Security Parameter Indeks* yang tepat untuk menjamin pengiriman paket-paket data.

G. TINJAUAN PUSTAKA DAN DASAR TEORI

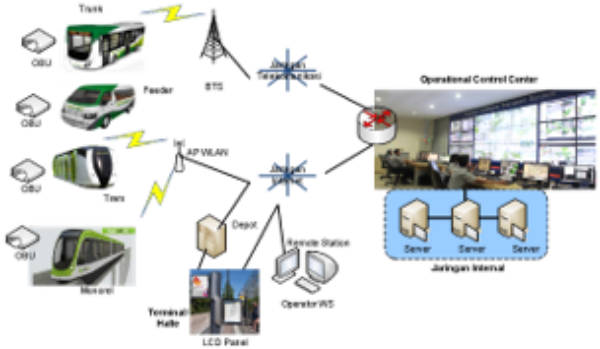
1. *Intelligent Transport System (ITS)*

Intelligent Transport System atau dalam bahasa Indonesia disebut dengan Sistem Transportasi cerdas, merupakan suatu sistem transportasi hasil dari pemanfaatan teknologi informasi dan telekomunikasi [1].

Terdapat tiga bidang dalam sistem transportasi cerdas, yakni [1]:

- *Intelligent Infrastructure* (Infrastruktur Cerdas)
- *Smart Vehicle* (Kendaraan Cerdas)
- *Information Services* (Layanan Informasi)

Untuk mewujudkan sistem transportasi cerdas yang dapat berjalan dengan baik, penelitian yang dilakukan oleh laboratorium ITS telah menghasilkan *on-board unit* (OBU) yang diletakkan dalam armada untuk mengumpulkan informasi terkait, variable messaging system untuk komunikasi antara OBU dan TMC, serta system *Traffic Management Center* (TMC) sebagai pusat pengelolaan seluruh transaksi yang terjadi.

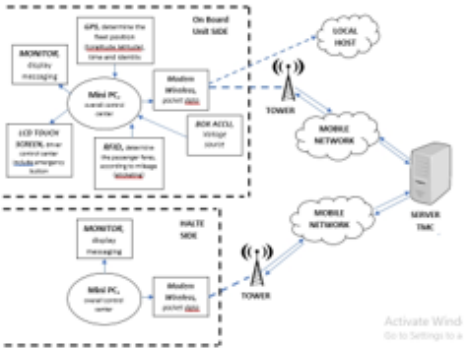


Gambar 1. Jaringan OBU-VMS dan TMC

1.1 OBU dan VMS

Dalam sistem transportasi cerdas, terdapat dua komponen utama yang harus tersedia yakni OBU dan VMS. *On-board Unit* (OBU) merupakan suatu komponen yang diletakkan dalam armada, berfungsi sebagai sistem pengendali masukan dan keluaran yang terkait dengan fungsional manajemen armada, pendapatan (tiket), lalu

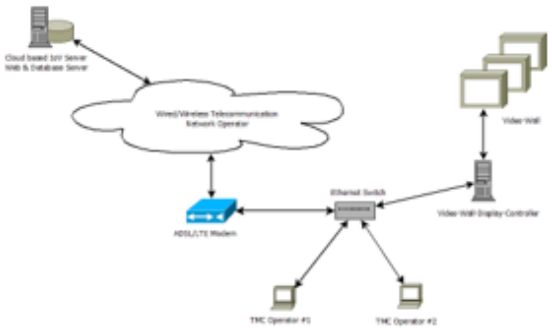
lintas, dan sistem darurat [2]. Sedangkan *Variable Messaging Unit* (VMS) bertugas untuk memberikan informasi kepada pengendara dan (calon) penumpang selama perjalanan armada. Secara global, sistem OBU dan VMS akan terbagi menjadi dua macam yaitu *OBU side* dan *Halte Side* [2]. Sistem OBU dan VMS dapat dilihat pada gambar 2.



Gambar 2. Blok Fungsional Sistem Global OBU dan VMS

1.2 TMC

TMC (*Traffic Management Center*) dalam ITS merupakan pusat kendali dan pemantauan pergerakan armada transportasi publik. TMC dibangun dengan tujuan untuk mewujudkan suatu sistem manajemen armada dan manajemen revenue yang dapat berjalan dengan baik serta dapat digunakan untuk memantau kinerja secara real time [3]. Konsep kerja dari TMC dimulai dari perangkat OBU yang mengirimkan data berupa data koordinat, data ticketing, dan data kondisi dalam bus maupun luar bus, kemudian seluruh data tersebut disimpan dalam server TMC dan ditampilkan ke wall-display [3]. Arsitektur jaringan TMC dapat dilihat pada gambar 3.



Gambar 3. Arsitektur Jaringan TMC

2. *Security Policy* (Kebijakan Keamanan)

Banyak sekali bidang-bidang yang dapat dibicarakan apabila membahas tentang kebijakan keamanan. Keamanan dapat digolongkan dalam beberapa bidang antara lain keamanan informasi, keamanan komputer, keamanan jaringan, keamanan multimedia, dan masih banyak lagi. Hal yang dapat diunggulkan dalam bidang keamanan ketika banyak pihak menawarkan sebuah sistem keamanan adalah dari segi kekuatan, seberapa kuat dan seberapa lama sistem keamanan tersebut dalam menahan serangan. Terdapat tiga aspek utama yang tidak bisa diabaikan dalam bidang security adalah [4]:

- *Confidentiality*, memastikan bahwa tidak ada yang mengetahui isi pesan selain pengirim dan penerima.
- *Integrity*, dapat membuktikan keaslian dari informasi, dimana pesan yang diterima merupakan pesan yang dikirim.
- *Authentication*, validasi identitas dari pihak pengirim dan penerima sehingga kedua belah pihak memang merupakan pihak yang diinginkan.

Selain tiga aspek yang telah disebutkan diatas, masih banyak lagi persyaratan atau aspek dalam layanan security, antara lain privacy, non-repudiation atau keaslian pengiriman, obliviousness (keraguan), information flow, availability authorization, verifiability, unforgetability, distinguishability, dan detectability [4].

2.1 *Network Security*

Network Security atau keamanan jaringan dilakukan untuk memantau dan mencegah akses yang tidak sah, penyalahgunaan, modifikasi, ataupun kegiatan lain yang tidak diinginkan terhadap suatu jaringan. Beberapa jenis *network security* yang ada saat ini antara lain adalah [5]:

- *Active Devices*, yakni perangkat yang dapat memblokir traffic yang berlebihan. Beberapa contoh dari perangkat *active devices* yaitu *firewall, antivirus scanning devices,* dan *content filtering devices.*
- *Passive Devices*, merupakan perangkat yang dapat mengidentifikasi dan melaporkan traffic yang tidak diinginkan, misalnya *intrussion detection appliances.*
- *Preventative Devices*, perangkat ini dapat memindai jaringan dan mengidentifikasi potensi dari masalah keamanan. Misalnya, *penetration testing devices* dan *vulnerability assesment appliances..*

3. Koneksi VPN (*Virtual Private Network*)

VPN atau *Virtual Private Network* merupakan sebuah teknologi yang digunakan untuk membangun koneksi private melalui koneksi publik, yang artinya jaringan hanya bersifat virtual dan tidak semua orang bisa mengakses jaringan tersebut [6]. Bisa dibilang bahwa dengan membuat VPN berarti membuat jaringan di dalam jaringan. Membangun koneksi VPN dapat dilakukan dengan dua cara. Pertama, dengan menggunakan IPsec (Internet Protocol Security) sebagai autentikasi dan enkripsi di antara *endpoint*. Dan yang kedua dengan menggunakan mekasisme tunneling. Beberapa teknologi tunneling hasil dari perkembangan VPN yang sering digunakan antara lain adalah PPTP (Point-to-Point Tunneling Protocol), L2TP (Layer 2 Tunneling Protocol), dan GRE (Generic Routing Encapsulation).

Terdapat dua tipe dalam koneksi VPN, Remote Access VPN dan Site-to-site VPN. Yang mana fitur-fitur dan karakteristik dari Remote Access VPN adalah sebagai berikut [7]:

- Menghubungkan user dengan jaringan perusahaan (corporate network)
- Internet Protocol Security
- Skema client-server
- Point-to-Point Tunneling Protocol
- Layer 2 Tunneling Protocol

Sedangkan fitur-fitur dan karakteristik dari Site-to-site VPN adalah:

- Menghubungkan antar network
- Internet Protocol Security
- Komunikasi host melalui VPN gateway
- Generic Routing Encapsulation
- Multi-Protocol Label Switching

Dengan penerapan VPN, hal-hal yang dapat diperoleh dari segi security service adalah privasi, integritas data, autentikasi, serta anti-replay.

3.1 *IPsec (Internet Protocol Security)*

IPsec menyediakan autentikasi dasar, integritas data, dan layanan enkripsi untuk melindungi dari akses tidak sah dan modifikasi data.

3.2 *ESP (Encapsulated Security Payload)*

ESP mencakup sisi header dan trailer untuk mendukung enkripsi dan autentikasi secara opsional. Ini juga menyediakan mode tunnel and transport yang sekarang banyak digunakan [8].

3.3 *Security Parameter Indeks (SPI)*

Security parameter indeks (SPI) adalah 32-bit number yang dihasilkan secara random yang digunakan untuk mengidentifikasi SA tertentu secara unique pada suatu paket data yang datang di sisi penerima [8]. Dimana SA (Security Association) merupaka n indikator dari parameter-parameter yang diketahui oleh sisi pengirim dan penerima [8]. SPI disediakan untuk memetakan paket yang datang pada sebuah SA di tujuan.

4. Enkripsi Data

Dalam IPsec protokol, dilakukan proses authentication yang mana digunakan untuk memverifikasi pengirim dan penerima yang tepat [9]. Dalam proses ini dapat dilakukan enkripsi data untuk merahasiakan data asli yang dikirim dari pihak ketiga atau attacker. Terdapat banyak sekali metode yang dapat digunakan untuk mengenkripsi suatu data, beberapa diantaranya yang paling sering digunakan antara lain adalah [9]:

- DES (Data Encryption Standard)
- AES (Advanced Encryption Standard)
- One Way Function (Hash)
- SHA
- RSA

H. METODOLOGI

Metodologi pada tugas akhir ini adalah sebagai berikut.

Gambar 4. Metodologi Penelitian

Dari gambar metodologi penelitian pada gambar 4, dapat dilihat tahapan-tahapan yang akan dilalui dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. **Studi literatur**

   Mengumpulkan dan memelajari buku serta referensi yang dapat membantu mewujudkan tujuan dari tugas akhir ini antara lain, *Intelligent Transportation System*, *Network Security*, Protokol Telekomunikasi, *Ethical Hacking*, *Internet Protocol Security (IPsec)*, *Tunneling*, Enkripsi data, serta beberapa bahasa pemrograman yang diperlukan seperti Delphi, C, dll.

2. **Perancangan dan Implementasi Sistem Keamanan Jaringan ITS**

   Setelah mengetahui dan mendapatkan semua bahan-bahan yang diperlukan dalam tugas akhir ini, kemudian dibuatlah suatu rancangan sistem keamanan yang kemudian diimplementasikan pada jaringan ITS.

3. **Simulasi dan Pengujian Sistem**

   Apabila sistem sudah berhasil diimplementasikan, maka selanjutnya adalah melakukan simulasi dan pengujian sistem dengan melakukan penetrasi untuk melihat ketahanan dari sistem keamanan yang telah dirancang.

4. **Pengambilan Data dan Analisis**

   Data diperoleh dari hasil simulasi dan pengujian sistem keamanan terhadap jaringan ITS. Kemudian dilakukan analisis untuk mengetahui tingkat performansi keamanan dari jaringan. Berikut merupakan parameter yang harus

5. **Kesimpulan**

   Kesimpulan dibuat setelah data yang telah diambil sudah dianalisis dengan matang.

6. **Penyusunan laporan**

Penerapan metodologi yang telah tertulis di atas kemudian disusun berbentuk laporan tugas akhir sebagai bentuk pertanggungjawaban penulis setelah melakukan kegiatan tersebut di atas.

I. RELEVANSI

Hasil Dari tugas akhir ini diharapkan dapat diimplementasikan pada jaringan Komunikasi ITS untuk memberikan keamanan paket data yang dikirim dari sisi OBU kepada sisi TMC server sehingga transaksi dapat diolah dengan aman dan lancar.

J. JADWAL KEGIATAN

Tabel Jadwal Kegiatan

Kegiatan	Bulan Februari 2018				Bulan Maret 2018				Bulan April 2018				Bulan Mei 2018				Bulan Juni 2018	
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
Mempelajari Literatur																		
Perancangan Sistem																		
Pembuatan Simulasi Sistem																		
Analisis																		
Bimbingan																		
Penyusunan Laporan																		
Ujian Tugas Akhir																		

LAMPIRAN B

DAFTAR KODE PROGRAM

1. Konfigurasi pada Publisher dengan Enkripsi

```
import time
import paho.mqtt.client as paho
from cryptography.fernet import Fernet

broker="192.168.1.5"
client= paho.Client("pub2")

cipher_key=b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
cipher = Fernet(cipher_key)
message = open("original.txt","r")
msg = message.read()
encrypted_message = cipher.encrypt(msg.encode('utf-8'))

print("connecting to broker ",broker)
client.connect(broker)#connect
print("publishing encrypted message ")

while 1:
    client.publish("paho/test1",encrypted_message)
    print ("Publishing Message =  ")
    time.sleep(1)

client.loop_forever()
```

2. Konfigurasi pada Subscriber dengan Dekripsi

```
import paho.mqtt.subscribe as subscribe
from cryptography.fernet import Fernet

def on_message(client, userdata, message):
  cipher_key                                    =b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
```

```
    cipher = Fernet(cipher_key)
    decrypted_message = cipher.decrypt(message.payload)
    out = decrypted_message.decode("utf-8")
    print (out)

subscribe.callback (on_message, "paho/test1", hostname =
"192.168.1.5")
```

3. Konfigurasi Publisher tanpa Enkripsi

```
import time
import paho.mqtt.client as paho

broker="192.168.1.5"
client= paho.Client("pub-local")

message = open("original.txt","r")
msg = message.read()

print("connecting to broker ",broker)
client.connect(broker)#connect

while 1:
    client.publish("paho/test1", msg)
    print("publishing message ")
    time.sleep(1)

client.loop_forever()
```

4. Konfigurasi Subscriber tanpa Enkripsi

```
import paho.mqtt.subscribe as subscribe


def on_message(client, userdata, message):
   print("%s %s" % (message.topic, message.payload))

subscribe.callback(on_message,                    "paho/test1",
hostname="localhost")
```

5. Konfigurasi pada Publisher GPS

```
import paho.mqtt.client as mqtt
import serial
import time
from cryptography.fernet import Fernet

serialStream = serial.Serial("/dev/ttyAMA0", 9600, timeout=0.5)
mqttc = mqtt.Client()
mqttc.connect("vps2.lawanghosting.pw", 1883, 60)
mqttc.loop_start()
key = b'8I7nMN_-ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
cipher = Fernet(key)

try:
   while True:
      sentence = serialStream.readline()
      if sentence.find('GGA') > 0:
         encrypt = cipher.encrypt(sentence.encode('utf-8'))
         print sentence
         mqttc.publish("OBU/GPS", encrypt)
except KeyboardInterrupt:
   streamer.close()
```

6. Subscriber GPS – Database

```
# -*- coding: utf-8 -*-

import paho.mqtt.client as mqtt
from cryptography.fernet import Fernet
import com_sensor
#
def on_connect(client, userdata, flags, rc):
      print("Connected with result code "+str(rc))
      client.subscribe(topic="OBU/GPS")
      client.subscribe(topic="OBU/#")

#
def on_message(client, userdata, msg):
```

```
        key = b'8I7nMN_-ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
        cipher = Fernet(key)
        decrypted = cipher.decrypt(msg.payload)
        pay = decrypted.decode("utf-8")
        pay_out=u"""+pay+"""
        print("on_message:topic=" + msg.topic+" ,pay="+str(pay_out))
        if len(pay_out)>10 and len(pay_out)<256:
                clsSen=com_sensor.sensorClass()
                clsSen.saveSensor( msg.topic, pay_out)

if __name__ == "__main__":
    client = mqtt.Client(client_id="spam")
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(host="185.201.9.130", port=1883)
    client.loop_forever()
```

7. Fungsi Setting Database

```
# -*- coding: utf-8 -*-

#com_appConst
class appConstClass:
    def __init__(self):
        print ""

    mHost = "localhost"
    mDB_NAME= "db_ITS_2018"
    mUser ="root"
    mPass ="aj404its!@#$%"
    mOK_CODE=1
    mNG_CODE=0

    def test(self, sId):
        print "test"
```

8. Konfigurasi SQL Database

```
# -*- coding: utf-8 -*-
import MySQLdb
import com_appConst

#com_sensor
class sensorClass:

    def __init__(self):
        print ""

    def saveSensor(self, topic, payload):
        ret=False
        clsConst = com_appConst.appConstClass()
        connection = MySQLdb.connect(host=clsConst.mHost,
db=clsConst.mDB_NAME, user=clsConst.mUser,
passwd=clsConst.mPass, charset="utf8")
        cursor = connection.cursor()
        sSql=u"INSERT INTO GPS_Log (serID, topic,
payload,created_at)"
        sSql=sSql+" values ("
        sSql=sSql+"serID"+","
        sSql=sSql+"'"+topic +"'"
        sSql=sSql+","+ payload
        sSql=sSql+",now() );"
        #print sSql
        cursor.execute(sSql)
        connection.commit()
        cursor.close()
        connection.close()
        ret=True
    return ret
```

9. Konfigurasi pada Publisher RFID

```
#!/usr/bin/env python
import paho.mqtt.client as mqtt
```

```
import time
import serial
from cryptography.fernet import Fernet

ser = serial.Serial(
      port='dev/ttyS0',
      baudrate = 9600,
      parity = serial.PARITY_NONE,
      stopbits= serial.STOPBITS_ONE,
      bytesize= serial.EIGHTBITS,
      timeout= 0.010,
      rtscts = True,
      dsrdtr = False

      )
cipher_key=b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
cipher = Fernet(cipher_key)
mqttc = mqtt.Client()
mqttc.connect("vps2.lawanghosting.pw", 1883, 60)
mqttc.loop_start()

while 1:
      data_rfid = ser.readline()
      encrypted_message = cipher.encrypt(data_rfid)
      out_message=encrypted_message.decode()
      if len (data_rfid) > 3:
                  print (data_rfid)
                  mqttc.publish("OBU/rfid", encrypted_message)
```

10. Konfigurasi pada Subscriber RFID

```
import paho.mqtt.subscribe as subscribe
from cryptography.fernet import Fernet

def on_message(client, userdata, message):
   cipher_key                                    =b'8I7nMN_-
ShTaSBOzZL4DWC66slvIYL9xUzeDiNIQ3Gg='
   cipher = Fernet(cipher_key)
```

```
    decrypted_message = cipher.decrypt(message.payload)
    out = decrypted_message.decode("utf-8")
    print (out)

subscribe.callback (on_message, "OBU/rfid", hostname =
"192.168.1.5")
```

Halaman ini mengandung kekosongan

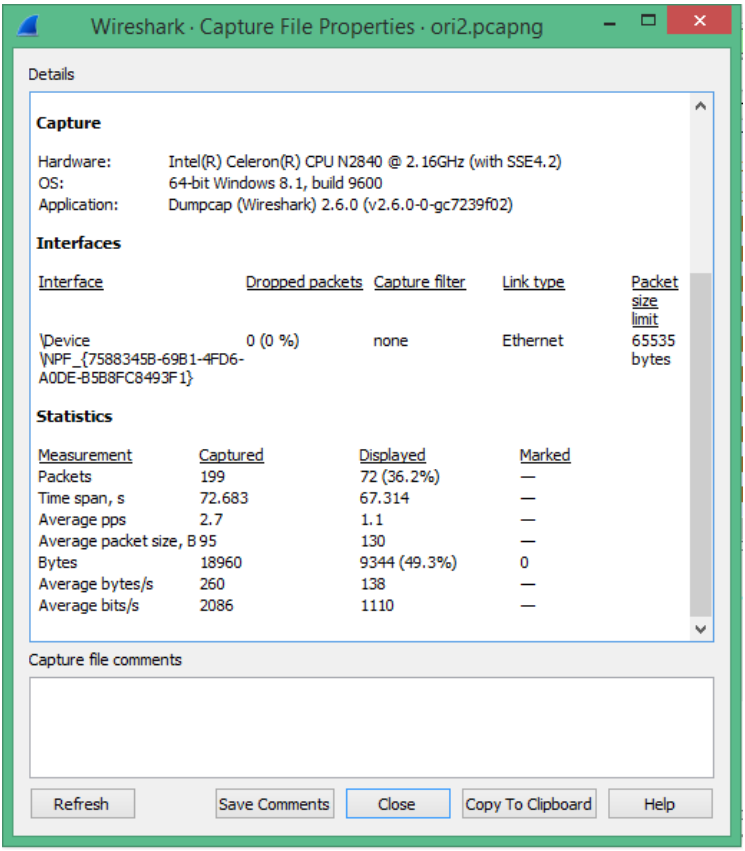
LAMPIRAN C

DATA HASIL PENGUJIAN

C.1 Hasil Pengujian Pesan Original (Data 1)

Tabel 1. Tabel Performansi Pengiriman Pesan Original (Data 1)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.008737	Terbaca	Tidak Aman
10	0.003197	Terbaca	Tidak Aman
15	0.005754	Terbaca	Tidak Aman
20	0.006931	Terbaca	Tidak Aman
25	0.00239	Terbaca	Tidak Aman
30	0.004312	Terbaca	Tidak Aman
35	0.003628	Terbaca	Tidak Aman
40	0.003618	Terbaca	Tidak Aman
45	0.001688	Terbaca	Tidak Aman
50	0.002895	Terbaca	Tidak Aman
55	0.007366	Terbaca	Tidak Aman
60	0.00441	Terbaca	Tidak Aman
Rata-rata = 0.0045772		% Keamanan = 0 %	



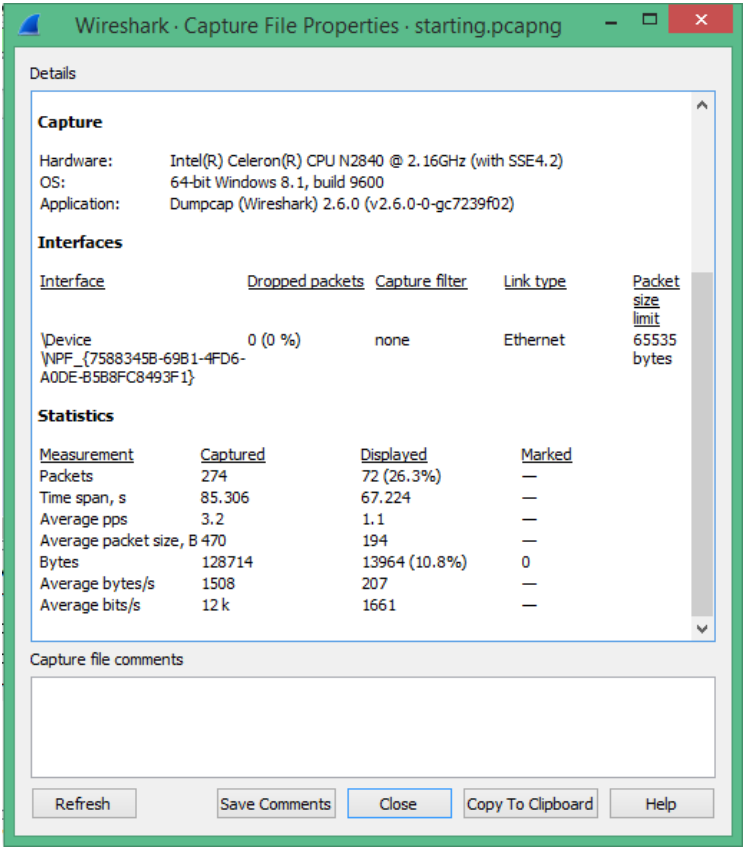
Gambar 1. Capture Wireshark Enkripsi tanpa Dekripsi (Data 1)

C.2 Hasil Pengujian Pesan Original (Data 2)

Tabel 2. Tabel Performansi Pengiriman Pesan Original (Data 2)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.01896	Terbaca	Tidak Aman
10	0.003473	Terbaca	Tidak Aman
15	0.005004	Terbaca	Tidak Aman
20	0.013694	Terbaca	Tidak Aman
25	0.007086	Terbaca	Tidak Aman

30	0.00411	Terbaca	Tidak Aman
35	0.019681	Terbaca	Tidak Aman
40	0.00352	Terbaca	Tidak Aman
45	0.002821	Terbaca	Tidak Aman
50	0.003647	Terbaca	Tidak Aman
55	0.004321	Terbaca	Tidak Aman
60	0.001061	Terbaca	Tidak Aman
Rata-rata = **0.007282**		% Keamanan = 0 %	

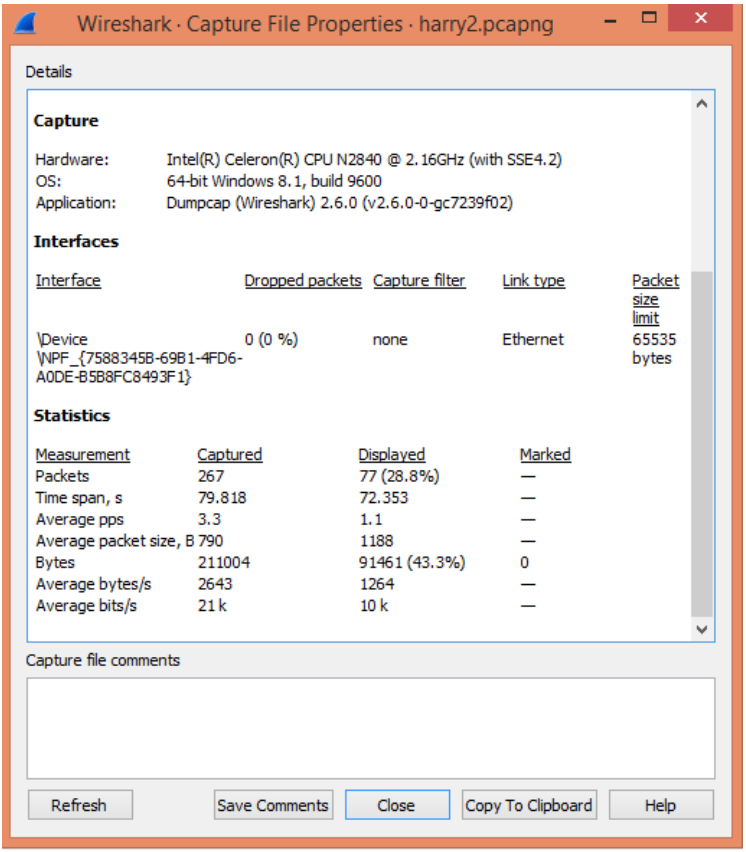


Gambar 2. Capture Wireshark Enkripsi tanpa Dekripsi (Data 2)

C.3 Hasil Pengujian Pesan Original (Data 3)

Tabel 3. Tabel Performansi Pengiriman Pesan Original (Data 3)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.00709	Terbaca	Tidak Aman
10	0.00392	Terbaca	Tidak Aman
15	0.006894	Terbaca	Tidak Aman
20	0.009588	Terbaca	Tidak Aman
25	0.004682	Terbaca	Tidak Aman
30	0.01678	Terbaca	Tidak Aman
35	0.007002	Terbaca	Tidak Aman
40	0.007339	Terbaca	Tidak Aman
45	0.004521	Terbaca	Tidak Aman
50	0.012863	Terbaca	Tidak Aman
55	0.007442	Terbaca	Tidak Aman
60	0.005491	Terbaca	Tidak Aman
Rata-rata = **0.007801**		% Keamanan = 0 %	



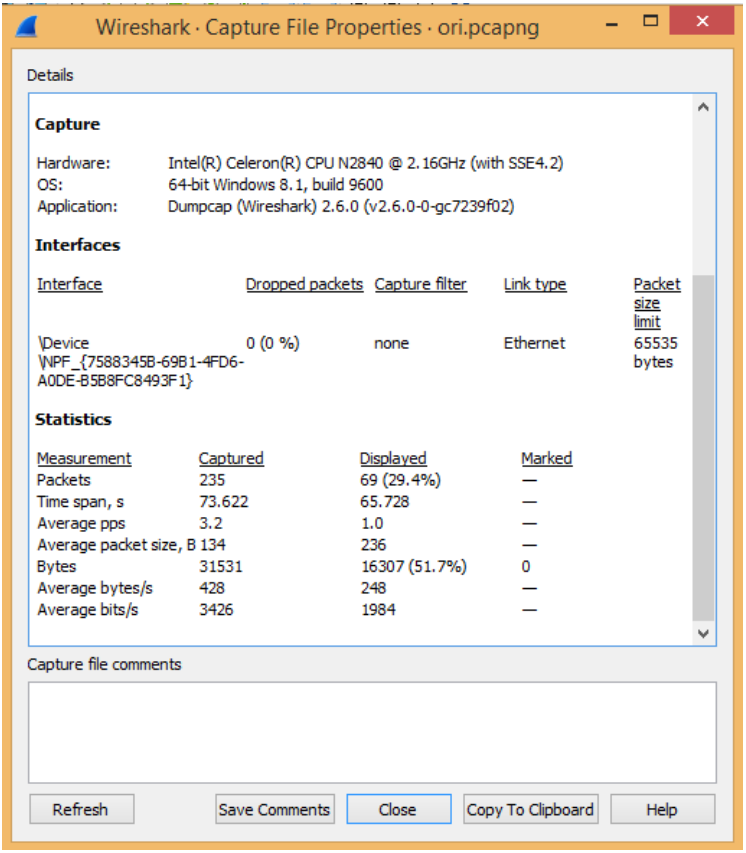
Gambar 3. Capture Wireshark Enkripsi tanpa Dekripsi (Data 3)

C.4 Hasil Pengujian Enkripsi tanpa Dekripsi (Data 1)

Tabel 4. Tabel Performansi Enkripsi tanpa Dekripsi (Data 1)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.008282	Tidak	Aman
10	0.003863	Tidak	Aman
15	0.005633	Tidak	Aman
20	0.014763	Tidak	Aman
25	0.003089	Tidak	Aman

30	0.004217	Tidak	Aman
35	0.004117	Tidak	Aman
40	0.014914	Tidak	Aman
45	0.003025	Tidak	Aman
50	0.004339	Tidak	Aman
55	0.003279	Tidak	Aman
60	0.005111	Tidak	Aman
Rata-rata = **0.006219**		% Keamanan = 100 %	

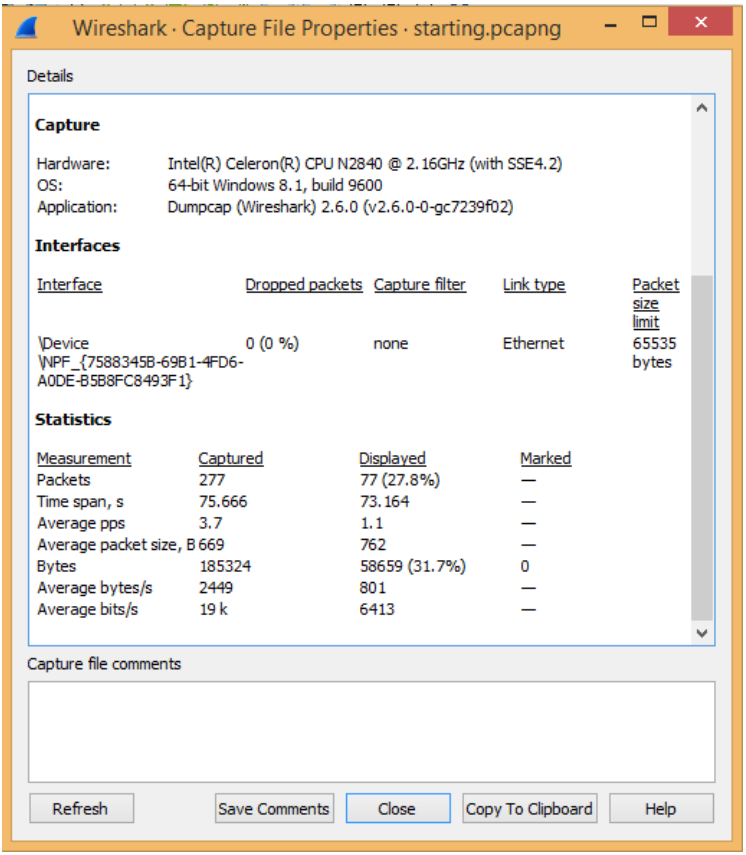


Gambar 4. Capture Wireshark Enkripsi tanpa Dekripsi (Data 1)

C.5 Hasil Pengujian Enkripsi tanpa Dekripsi (Data 2)

Tabel 5. Tabel Performansi Enkripsi tanpa Dekripsi (Data 2)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.005201	Tidak	Aman
10	0.003809	Tidak	Aman
15	0.002506	Tidak	Aman
20	0.005176	Tidak	Aman
25	0.005576	Tidak	Aman
30	0.009306	Tidak	Aman
35	0.006001	Tidak	Aman
40	0.003052	Tidak	Aman
45	0.003363	Tidak	Aman
50	0.012787	Tidak	Aman
55	0.016201	Tidak	Aman
60	0.018521	Tidak	Aman
Rata-rata = **0.007625**		% Keamanan = 100 %	



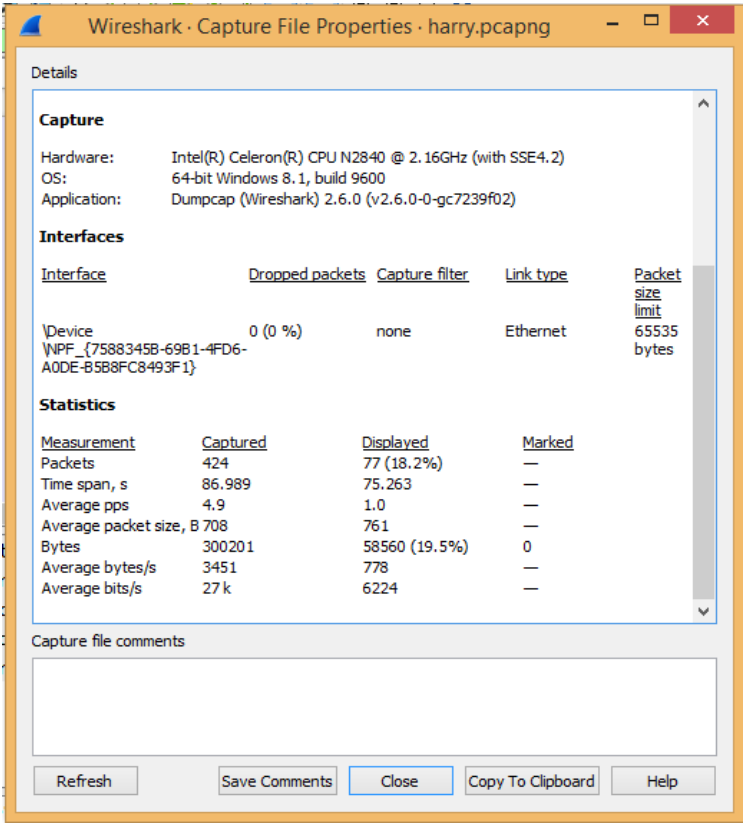
Gambar 5. Capture Wireshark Enkripsi tanpa Dekripsi (Data 2)

C.6 Hasil Pengujian Enkripsi tanpa Dekripsi (Data 3)

Tabel 6. Tabel Performansi Enkripsi tanpa Dekripsi (Data 3)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5		Tidak	Aman
10		Tidak	Aman
15		Tidak	Aman
20		Tidak	Aman
25		Tidak	Aman

30		Tidak	Aman
35		Tidak	Aman
40		Tidak	Aman
45		Tidak	Aman
50		Tidak	Aman
55		Tidak	Aman
60		Tidak	Aman
Rata-rata =		% Keamanan = 100 %	

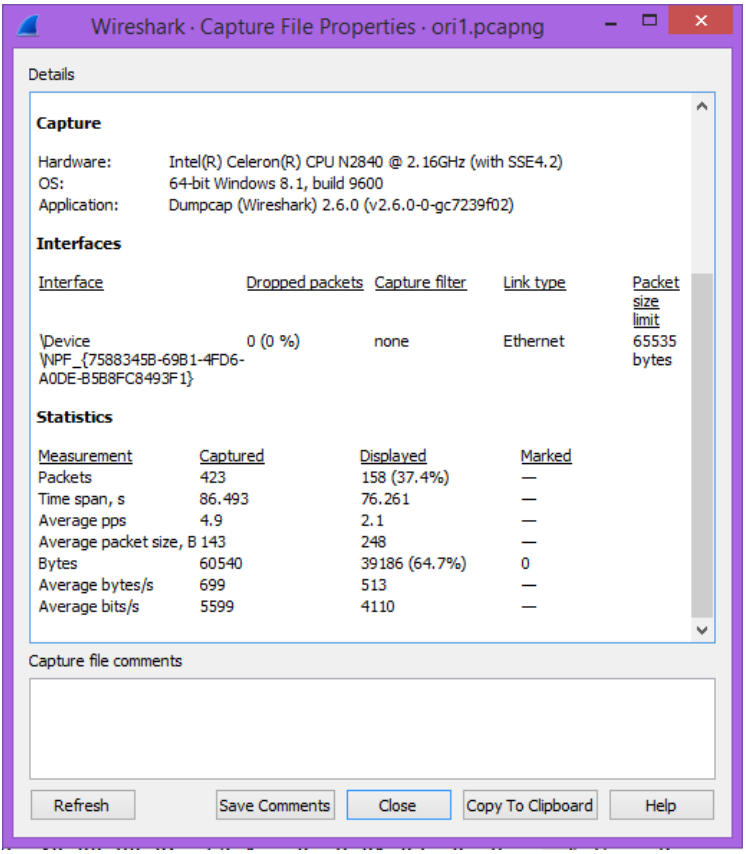


Gambar 6. Capture Wireshark Enkripsi tanpa Dekripsi (Data 3)

C.7 Hasil Pengujian Enkripsi - Dekripsi (Data 1)

Tabel 7. Tabel Performansi Enkripsi - Dekripsi (Data 1)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.009988	Tidak	Aman
10	0.012194	Tidak	Aman
15	0.005134	Tidak	Aman
20	0.003112	Tidak	Aman
25	0.015075	Tidak	Aman
30	0.003531	Tidak	Aman
35	0.005089	Tidak	Aman
40	0.004178	Tidak	Aman
45	0.001504	Tidak	Aman
50	0.008392	Tidak	Aman
55	0.00887	Tidak	Aman
60	0.002589	Tidak	Aman
Rata-rata = **0.006638**		% Keamanan = 100 %	



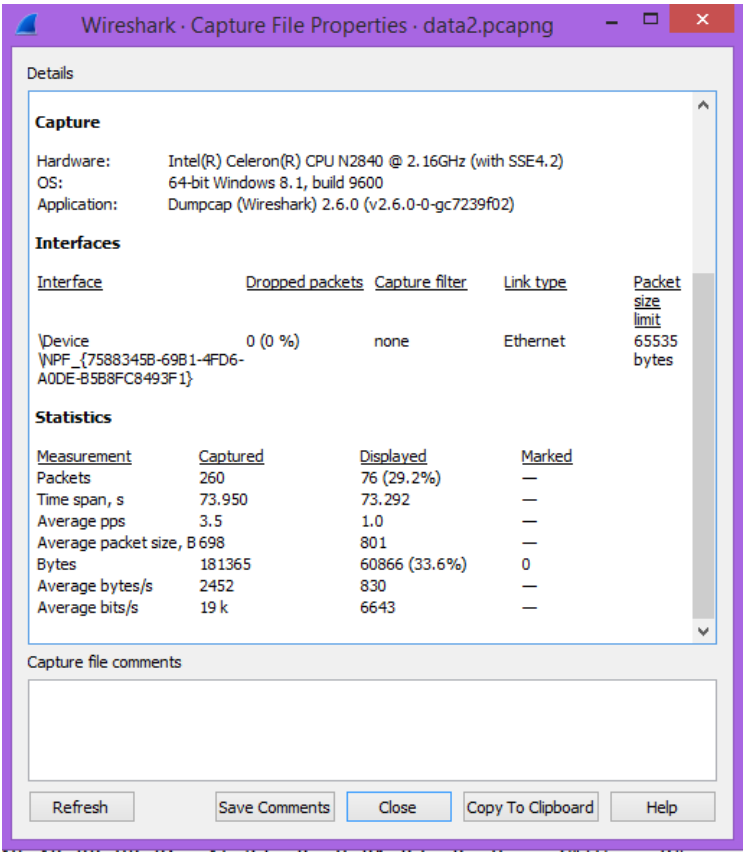
Gambar 7. Capture Wireshark Enkripsi - Dekripsi (Data 1)

C.8 Hasil Pengujian Enkripsi - Dekripsi (Data 2)

Tabel 8. Tabel Performansi Enkripsi - Dekripsi (Data 2)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.003476	Tidak	Aman
10	0.005213	Tidak	Aman
15	0.009871	Tidak	Aman
20	0.003905	Tidak	Aman
25	0.003858	Tidak	Aman

30	0.004872	Tidak	Aman
35	0.014676	Tidak	Aman
40	0.004285	Tidak	Aman
45	0.010765	Tidak	Aman
50	0.002964	Tidak	Aman
55	0.033185	Tidak	Aman
60	0.013735	Tidak	Aman
Rata-rata = **0.009234**		% Keamanan = 100 %	

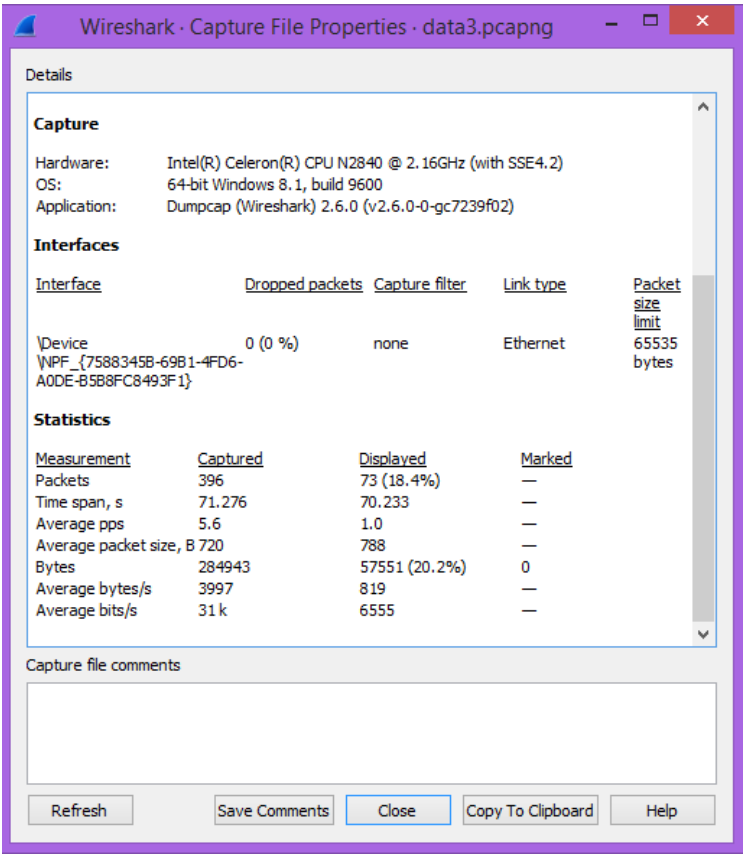


Gambar 8. Capture Wireshark Enkripsi - Dekripsi (Data 2)

C.9 Hasil Pengujian Enkripsi - Dekripsi (Data 3)

Tabel 9. Tabel Performansi Enkripsi - Dekripsi (Data 3)

Detik	Delay (s)	Pesan terbaca / tidak	Ket.
5	0.00437	Tidak	Aman
10	0.007986	Tidak	Aman
15	0.008878	Tidak	Aman
20	0.019716	Tidak	Aman
25	0.012914	Tidak	Aman
30	0.011058	Tidak	Aman
35	0.012192	Tidak	Aman
40	0.009648	Tidak	Aman
45	0.008421	Tidak	Aman
50	0.006785	Tidak	Aman
55	0.006376	Tidak	Aman
60	0.010057	Tidak	Aman
Rata-rata = **0.009867**		% Keamanan = 100 %	



Gambar 9. Capture Wireshark Enkripsi - Dekripsi (Data 3)

RIWAYAT HIDUP PENULIS

Penulis buku ini memiliki nama lengkap Diana Musabbihah dengan nama panggilan Ica atau Diana untuk lebih formalnya. Lahir dan besar di kabupaten Pasuruan, tepatnya kecamatan Pandaan, pada hari Jumat tanggal 05 September 1997. Hanya memiliki satu saudara perempuan yang lebih tua darinya. Pendidikan pertama yang Ia ditempuh adalah di Play Group Al-Hikmah, dilanjutkan dengan TK pada tahun 2001-2003 , lalu masuk MI Mamba'ul Ulum dan lulus pada tahun 2009. Setelah itu Ia pergi meninggalkan kotanya untuk mengenyam pendidikan lebih lanjut di Pondok Pesantren yang ada di kota Mojokerto selama 5 tahun dengan rincian, MTs Unggulan Amanatul Ummah pada tahun 2009 hingga tahun 2011 untuk jenjang sekolah menengah pertama dan MBI Amanatul Ummah dari tahun 2011 hingga tahun 2014 untuk jejang sekolah menengah atas. Setelah lulus, Ia berhasil diterima di Departemen Teknik Elektro ITS (S1) pada tahun 2014. Jika ingin menghubungi penulis dapat melalui alamat email : musabbihah14@gmail.com.