ANALISIS DAN IMPLEMENTASI PENGAMANAN DISTRIBUSI KUNCI (*KEY DISTRIBUTION*) DAN ENKRIPSI PADA APLIKASI PESAN INSTAN MENGGUNAKAN KRIPTOGRAFI KUNCI PUBLIK DAN SIMETRI

Oleh

JOHN MATRIO YAHYA

T3109204

SKRIPSI

Untuk memenuhi salah satu syarat ujian
guna memperoleh gelar Sarjana

PROGRAM SARJANA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS ICHSAN GORONTALO
GORONTALO
2017

HALAMAN PERSETUJUAN

ANALISIS DAN IMPLEMENTASI PENGAMANAN DISTRIBUSI KUNCI (*KEY DISTRIBUTION*) DAN ENKRIPSI PADA APLIKASI PESAN INSTAN MENGGUNAKAN KRIPTOGRAFI KUNCI PUBLIK DAN SIMETRI

Oleh

JOHN MATRIO YAHYA

T3109204

SKRIPSI

Untuk memenuhi salah satu syarat ujian
guna memperoleh gelar Sarjana
Program Studi Teknik Informatika, ini
telah disetujui oleh Tim Pembimbing.

Gorontalo, April 2017

Pembimbing Utama	Pembimbing Pendamping
Budy Santoso, S.Kom, M.Eng	**Warid Yunus, S.Kom, M.Kom**
NIDN. 0908048403	**NIDN. 0914059001**

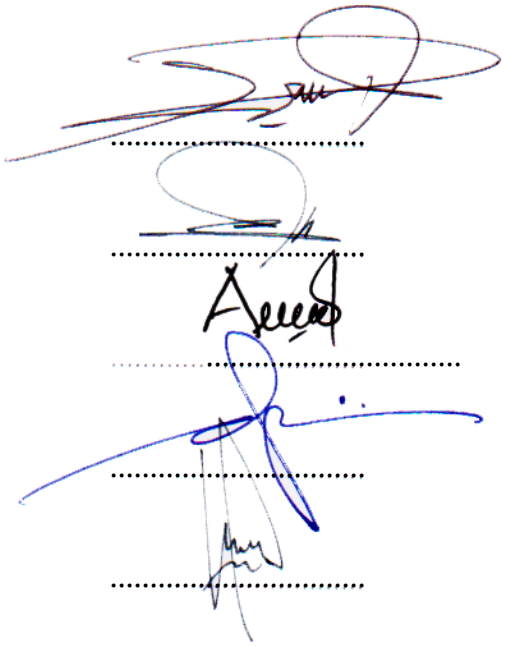
HALAMAN PENGESAHAN

ANALISIS DAN IMPLEMENTASI PENGAMANAN DISTRIBUSI KUNCI (*KEY DISTRIBUTION*) DAN ENKRIPSI PADA APLIKASI PESAN INSTAN MENGGUNAKAN KRIPTOGRAFI KUNCI PUBLIK DAN SIMETRI

OLEH
JOHN MATRIO YAHYA
T3109204

Diperiksa Oleh Panitia Ujian Skripsi Strata Satu (S1)
Universitas Ichsan Gorontalo

1. Ketua
   Irvan Abraham Salihi, M.Kom .....
2. Anggota
   Sunarto Taliki, M.Kom .....
3. Anggota
   Andi Nirfah, M.Kom .....
4. Anggota
   Budy Santoso, S.Kom, M.Eng .....
5. Anggota
   Warid Yunus, M.Kom .....



HALAMAN PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Karya tulis saya (Skripsi) ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (Sarjana) baik di universitas ichsan Gorontalo maupun di Perguruan Tinggi lainnya.
2. Karya tulis ini adalah murni gagasan, rumusan dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan dari Tim Pembimbing.
3. Dalam karya tulis ini tidak terdapat karya atau pendapat yang telah dipublikasikan orang lain, kecuali secara tertulis dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan tinggi ini.

Gorontalo, April 2017

Yang Membuat Pernyataan,

John Matrio Yahya

ABSTRACT

Ease era of communication these days is experiencing rapid development, one of which is the development of instant messaging applications (Instant Messenger). One concern is about the security of messages sent over a public network. Several methods have been developed to improve the security aspects of data communications. This study aims to establish the method of securing an instant messaging are quick but safe.

In this final project created software to test simultaneously analyze a combination of two encryption algorithms in terms of both computational speed and security of a message that is not easily read by parties who have no interest. The combination of RSA and AES encryption algorithm used to secure the distribution of symmetric key algorithms and message encryption.

The results showed that the application of the method of the combination of RSA and AES algorithm to establish the method of securing the highly efficient instant messaging. This is evidenced by the small computing time and system resource usage slightly. Therefore, the method of securing the combination of RSA and AES algorithm is very suitable for instant messaging communications with small resource.

Keywords: Securing instant messaging, combination of RSA and AES Encryption Message

ABSTRAK

Era kemudahan komunikasi akhir-akhir ini mengalami perkembangan pesat, salah satunya perkembangan aplikasi pesan instan (*Instan Messenger*). Salah satu yang menjadi perhatian adalah tentang keamanan pesan yang dikirim melalui jaringan publik. Beberapa metode telah dikembangkan untuk meningkatkan aspek keamanan komunikasi data. Penelitian ini bertujuan membangun metode pengamanan pengiriman pesan instan yang ringkas namun aman.

Pada tugas akhir ini dibuatkan perangkat lunak untuk menguji sekaligus menganalisa kombinasi dua algoritma enkripsi baik dari segi kecepatan komputasi dan keamanan pesan sehingga tidak mudah dibaca oleh pihak yang tidak memiliki kepentingan. Kombinasi algoritma enkripsi RSA dan AES dipakai untuk melakukan pengamanan distribusi kunci algoritma simetris dan enkripsi pesan.

Hasil penelitian menunjukkan bahwa penerapan metode kombinasi algoritma RSA dan AES untuk membangun metode pengamanan pada pesan instan sangat efisien. Hal ini dibuktikan dengan waktu komputasi yang kecil serta penggunaan resource sistem yang sedikit. Oleh karena itu metode pengamanan dengan kombinasi algoritma RSA dan AES sangat cocok untuk komunikasi pesan instan dengan resource yang kecil.

Kata Kunci: Pengamanan pesan instan, Kombinasi RSA dan AES, Enkripsi Pesan

KATA PENGANTAR

Puji syukur penulis panjatkan kehadiran Allah SWT, karena atas berkat dan rahmat-Nya penulis dapat menyelesaikan Skripsi ini dengan judul, **Analisis dan implementasi pengamanan distribusi kunci (*key distribution*) dan enkripsi pada aplikasi pesan instan menggunakan kriptografi kunci publik dan simetri**, sesuai dengan yang direncanakan. Skripsi ini dibuat untuk memenuhi salah satu syarat untuk memperoleh gelar Sarjana Komputer. Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, Skripsi ini tidak dapat penulis selesaikan. Oleh karena itu penulis menyampaikan terima kasih kepada:

1. Dra. Hj. Juriko Abdussamad, M.Si Selaku Ketua Yayasan Pengembangan Ilmu Pengetahuan dan Teknologi (YPIPT) Ichsan Gorontalo.
2. Bapak Dr. Abdul Gaffar La Tjokke, M.Si, selaku Rektor Universitas Ichsan Gorontalo.
3. Ibu Zohrahayati, S.Kom, M.Kom , selaku Dekan Fakultas Ilmu Komputer
4. Ibu Asmaul Husnah Nasrullah, S.Kom, M.Kom selaku Pembantu Dekan I Bidang Akademik.
5. Ibu Irma Surya Kumala, S.Kom, M.Kom, selaku Pembantu Dekan II Bidang Administrasi Umum dan Keuangan.
6. Bapak Yasin Aril Mustofa, S.Kom, M.Kom, selaku Pembantu Dekan III Bidang Kemahasiswaan.
7. Bapak Irvan Salihi, S.Kom, M.Kom, selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer.
8. Bapak Budy Santoso, S.Kom, M.Eng, selaku Pembimbing Utama, yang telah membimbing penulis selama mengerjakan skripsi ini.
9. Bapak Warid Yunus, S.Kom, M.Kom selaku Pembimbing Pendamping, yang telah membimbing penulis selama mengerjakan skripsi ini.
10. Bapak dan Ibu Dosen yang telah mendidik dan membimbing penulis dalam memahami materi kuliahan khususnya masalah kriptografi dan keamanan komputer maupun jaringan.

11. Kedua orang tua dan keluarga yang telah membantu/mendukung penulisan skripsi ini.
12. Istri dan anak yang telah memberi dorongan dan menemani dalam penulisan skripsi ini.
13. Ucapan terima kasih kepada Semua pihak yang telah membantu penulis dalam penyelesaian skripsi ini.

Semoga beliau-beliau di atas mendapatkan imbalan yang lebih besar dari Allah SWT, melebihi apa yang beliau-beliau berikan kepada penulis.

Gorontalo, April 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pesan instan atau *Instan messenger* (IM) adalah sebuah alat komunikasi digital, banyak digunakan untuk berbagai keperluan. Namun hal ini dibarengi dengan adanya tindakan kejahatan seperti *sniffing* ataupun *eavasdropping* memungkinkan data yang dikirimkan oleh pengguna dapat direkam dan dimodifikasi, tindakan ini umumnya dilakukan untuk mencuri data yang dikirim melalui jaringan tanpa dienkripsi terlebih dahulu.

Adanya hal hal tersebut membuat pengembangan IM menjadi sangat pesat menghasilkan berbagai model pengamanan sistem yang dianggap lebih baik. Salah satu dari model pengamanan yaitu model *Spam Detection* dan *Filtering* model ini dilakukan dengan menerapkan daftar *Black/White* dimana paket data yang dikirim akan diperiksa terlebih dahulu sumber dan tujuannya untuk memeriksa validitas serta melakukan penyaringan pesan yang melewati sistem. Kelemahan sistem ini karena *Filtering* hanya bekerja pada sisi penerima pesan, sehingga masih dapat di sadap karena semua pesan dikirim secara broadcast ke jaringan IM.

Selanjutnya model pengamanan di level sistem. Model pengamanan ini dilakukan dengan mengenkripsi semua proses sebelum data ditransmisikan, bayak kombinasi yang dipakai untuk melakukan enkripsi pesan, metode kombinasi RSA dan TripleDES pada penenelitian Wenpin Guo dkk. Metode pengamanan ini menjadi lebih aman karena telah menerapkan enkripsi sebelum pesan di

transmisikan, meskipun demikian permasaalahan muncul ketika terjadi pertukaran data yang makin besar yang menyebabkan kecepatan komputasi menjadi lambat.

Penelitian berikutnya oleh Yusof dkk. dengan metode penerapan fungsi *Secure Hash Algorithm* (SHA), pesan text dianggap lebih cocok menggunakan algoritma ini. Metode dilakukan dengan menetapkan fungsi F(M) dimana nilai M adalah panjang pesan yang bisa berubah-ubah. 2 variabel penting yaitu *checksum* dan *securehash.* Checksum merupakan fungsi dari pesan (M) yang pada dasarnya hanya memiliki satu *property* sehingga nilai F(M) tergantung pada *byte* yang ada pada M. dengan memanfaatkan metode ini pesan yang dikirim dianggap lebih aman, namun yang menjadi catatan adalah tipe pesan M pada sistem ini hanya terdiri dari M1 dan M2 dan tentu sangat riskan sebab kode yang dibangkitkan untuk pertama kalinya bisa terdeteksi dengan lebih mudah.

Metode berikutnya yaitu kombinasi Diffie Hellman dan AES serta SHA sebagai signaturenya diteliti oleh A. Nalawade dkk. dengan Diffie Hellman sebagai *key exchange* dari *Chiperkey* AES dan enkripsi pesan dengan AES serta SHA-1 sebagai signature validasi pesan. Metode ini sangat kuat untuk enkripsinya dan dianggap aman untuk komunikasi data pada pesan instan, masalahnya adalah 3 kombinasi ini membutuhkan proses yang panjang karena masing masing algoritma menjalankan proses enkripsinya sebelum pesan tersebut di transmisikan ke jaringan IM sehingga kurang effisien.

Ada puluhan metode pengamanan pada IM telah di perkenalkan pada penelitian penelitian sebelumnya, beberapa diantaranya menggunakan metode kombinasi dari beberapa algoritma enkripsi namun terlalu banyak kombinasi dan

algoritma yang digunakan justru membuat proses pengamanan pesan menjadi kurang effisien terhadap masalah komputasi untuk perangkat-perangkat dengan *resource* kecil.

Untuk menanggulangi permasalahan tersebut, dalam penelitian kali ini kami mencoba menggunakan metode kombinasi dua algoritma saja untuk di terapkan pada aplikasi pesan instan (IM). Algoritma AES dan RSA menjadi kandidat yang akan digunakan untuk melakukan proses enkripsi, *key exchange* dan *signaturing* dimana kedua algoritma ini merupakan algoritma terbaik dan terpopuler di masing-masing kategori oleh NIST (*National Institutes of Standards and Technology*) pada tahun 2001, dengan menerapkan dua algoritma serta metode pengamanan pada penelitian ini kami anggap mampu menangani permasalahan komputasi yang panjang, dengan kapabilitas yang akan ditingkatkan dari pada penelitian sebelumnya baik itu proses pengamanan pendistribusian kunci, pengamanan isi pesan hingga pemberian *signature* untuk validasi isi pesan yang otentik dengan peningkatan waktu komputasi, sehingga metode ini bisa di terapkan pada perangkat keras dengan resource sistem yang kecil.

1.2 Rumusan Masaalah

Berdasarkan latar belakang diatas, yang menjadi permasaalahan dan perlu dikaji yaitu:

1. Bagaimana mengamankan pertukaran kunci enkripsi dari algoritma simetri dengan metode yang ringkas.
2. Bagaimana membuat metode enkripsi dan signaturing dengan efisien.

1.3 Batasan Masaalah

Dalam penyusunan proposal penelitian ini, pembahasan ditekankan pada:

1. Penelitian dibatasi pada aplikasi *instant messenger* dengan layanan pesan text saja.
2. Penelitian dan implementasi dikhususkan pada metode kombinasi algoritma kunci simetri AES dan algoritma kunci asimetri RSA yang diterapkan untuk membuat metode baru dari penelitian sebelumnya.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai melalui penelitian ini adalah membangun metode pengamanan pengiriman pesan instan yang memenuhi aspek *confidentiality*, *non-repudiation* serta *authentication* dengan proses yang ringkas namun aman, sehingga penerapannya cocok pada pengguna IM dengan resource kecil.

1.5 Kegunaan / Manfaat Penelitian

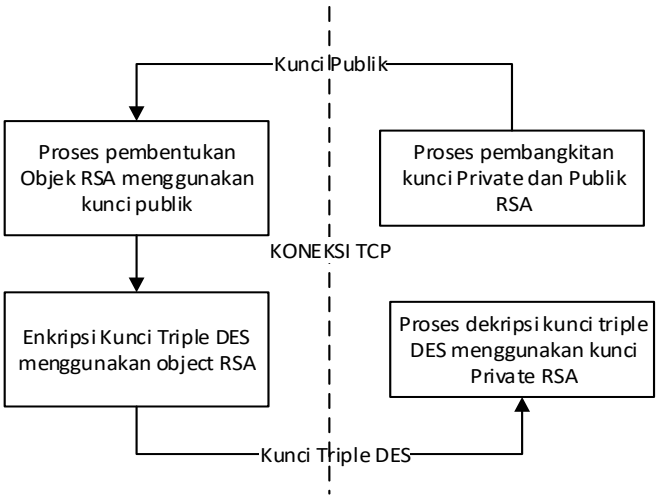
Dengan adanya penelitian dalam bidang enkripsi komunikasi pada aplikasi instant masenger ini diharapakan:

1. Penelitian ini dapat memberikan informasi tambahan bagi developer aplikasi IM tentang manajemen kunci yang memegang peranan yang sangat penting dalam aplikasi kriptografi, cara mendistribusikan kunci yang aman dengan mengkombinasikan enkripsi simetris dan asimetris, penggunaan signature untuk aspek otentikasi serta enkripsi pesan untuk kerahasiaan data.
2. Bagi penulis, hasil penelitian ini dapat dijadikan sebagai bahan kajian penelitian selanjutnya yang memiliki topik yang serupa dengan penelitian ini.

BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Studi

Penggunaan kriptografi dalam metode pengamanan *Instant Mesenger* telah diteliti dan berkembang. Beberapa penelitian sebelumnya dilakukan untuk mengembangkan metode pengamanan *Instant Mesenger* (IM) antara lain: model yang di usulkan untuk meningkatkan keamanan IM dijumpai dalam penelitian (Guo et al. 2009) dimana dia mengembangkan metode pengamanan IM dengan menenkripsi semua pesan sebelum data ditransmisikan. Ketika *client* menerima data *chippertext* maka data akan di dekripsikan menggunakan kombinasi algoritma RSA dan Triple DES. Model pengamanan berbasis RSA dan Triple DES ini menjadi lebih aman karena pesan telah mengalami proses enkripsi sebelum di transmisikan melalui internet. Model pengamanan pesan menggunakan algoritma RSA dan Triple DES ini diilustrasikan pada gambar dibawah ini.



Gambar 2.1 Komunikasi IM berdasarkan model Wenping Guo et al (2009).

Di 2004 beberapa protocol keamanan telah di perkenalkan dan salah satunya adalah protocol *Off the Record* (OTR). Protocol ini dikembangkan oleh (Borisov et al. 2004) sebagai perbaikan dan peningkatan dari OpenPGP dan system S/MIME pada "*Workshop on Privacy in the Electronic Society"* (WPES). Off-the-Record Messaging (OTR) adalah protokol kriptografi yang menyediakan enkripsi untuk percakapan IM. OTR menggunakan kombinasi AES symmetric-key algoritma dengan 128 bit panjang kunci, Diffie-Hellman pertukaran kunci dengan 1.536 ukuran kelompok bit, dan fungsi hash SHA-1. Selain otentikasi dan enkripsi, OTR memberikan peningkatan kerahasiaan dan enkripsi yang dapat dibentuk.

Banyak penelitian bidang keamanan pesan IM yang sudah dilakukan, beberapa diantara metode yang digunakan belum sepenuhnya memfokuskan pada efisiensi lama waktu proses dengan berbagai kombinasi. Oleh karena itu, penelitian ini akan membahas metode pengamanan pesan IM dengan metode kombinasi 2 Algorima enkripsi dengan fokus pada peningkatan waktu proses pengamanan pendistribusian kunci, kerahasiaan pesan, nirpenyangkalan serta otentikasi yang dikirimkan melalui jaringan *unsecure*.

Tabel berikut ini merangkum beberapa penelitian terkait pengamanan pada aplikasi *instant messenger*.

Tabel 2.1 Rangkuman penelitian terkait

1	Wenping Guo et al (2009)	Mengembangkan metode pengamanan IM dengan mengkripsi semua pesan sebelum data ditransmisikan. Ketika client menerima data chippertext maka data akan didekripsikan menggunakan kombinasi algoritma RSA dan Triple DES. Model pengamanan berbasis RSA dan Triple DES ini menjadi lebih aman karena pesan telah

		mengalami proses enkripsi sebelum ditransmisikan melalui internet.
2	Yusof et al (2011)	Mengembangkan metode pengamanan IM berbasis Message Digest. Pengembangan metode pengamanan pesan dilakukan dengan menerapan fungsi Hash pada data yang akan ditransmisikan melalui IM dimana pesan teks dianggap lebih cocok untuk menggunakan enkripsi tipe Secure Hash Algoritm (SHA). Dengan memanfaatkan metode ini pesan yang dikirim melalui jalur internet dianggap lebih aman.
3	Mohammed H. et al (2011)	Mengembangkan metode pengamanan IM berbasis protokol. Protokol yang terbentuk dari kombinasi ElGamal cryptosystem, algoritma RSA, and Chinese Remainder Theorem (CRT). Pada protokol ini bagian CRT lebih berperan pada pembaharuan private key pengguna Di dalam membentuk protokol IM ini CRT dianggap sebagai pasangan bilangan prima yang bernilai relatif dan akan dilakukan proses konkruen secara bersamaan. Pengembangan teknik pengamanan pada penelitian ini menggunakan metode ElGamal Cryptosystem.
4	A. Nalawade et al (2014)	Mengusulkan model protokol Off The Record (OTR) untuk penerapan IM. Pendekatan melalui protokol Off The Record (OTR). Protokol ini bekerja dengan memanfaatkan enkripsi yang cukup kuat yaitu dengan algoritma kunci simetri AES, pertukaran kunci menggunakan konsep Diffie Helman, menggunakan fungsi Hash SHA-1. Proses pengamanan dimulai sejak user mengaktifkan OTR. Setelah OTR diaktifkan, pengguna IM dapat menggunakan saluran komunikasi khusus yang terenkripsi.

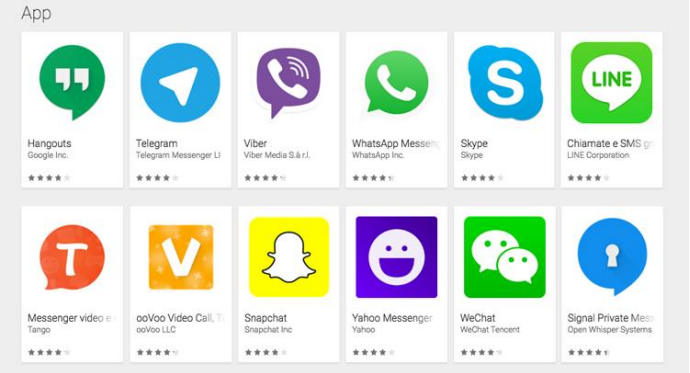
5	Penelitian Saya	Penelitian ini berfokus pada metode kombinasi 2 Algorimta kriptographi. Menggunakan enkripsi algoritma AES yang cukup kuat, pengamanan distribusi kunci menggunakan RSA sekaligus penggunaan digital signature untuk otentikasi.

2.2 Tinjauan Teori

Pada penelitian ini penulis menggunakan literatur-literatur yang di jadikan sebagai sumber kepustakaan meliputi: Konsep *Instan Mesenger* (IM), definisi *client- server, P2P,* Kriptografi, algoritma kunci simetris dan asimetris, protocol pertukaran kunci beserta algorimanya.

2.2.1 Instan Mesenger

Instant messenger merupakan salah satu layanan aplikasi yang sangat banyak digunakan saat ini untuk bertukar pesan melaui jaringan publik. Pesan instant yang ditransmisikan melalui jaringan internet memiliki risiko jika datanya bersifat rahasia (*private*). Saat ini IM semakin mengarah ke perangkat *mobile* yang sudah memiliki kemampuan komputasi yang tinggi. Berbagai macam aplikasi IM yang berkembang saat ini dapat dilihat pada Gambar 2.2.

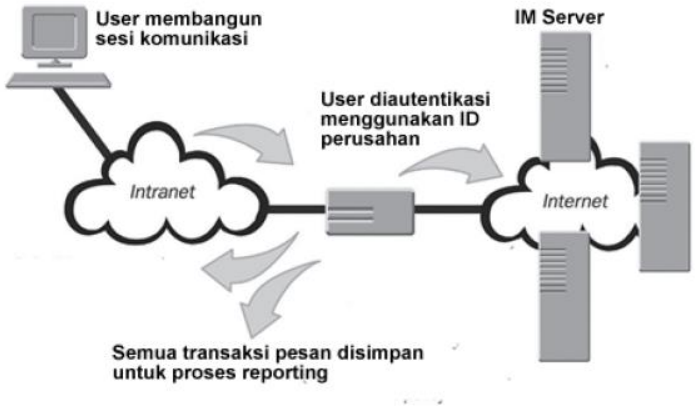


Gambar 2.2 Macam ragam IM

Perkembangan teknologi *computer* saat ini menjadikan aplikasi berkembang pesat. Instant messenger paling banyak dikembangkan dengan arsitektur yang meliputi:

1. *Client Server*

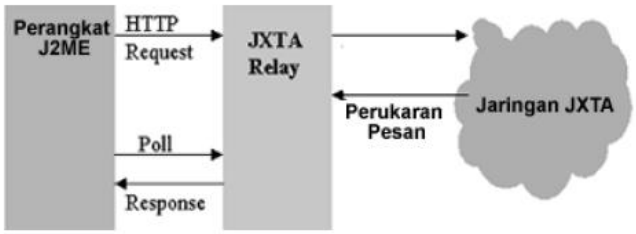
   Menurut (Casey 2007) *Instant Messenger* yang diterapkan pada sebuah lingkungan korporasi harus bersifat aman dengan menerapkan standar keamanan pada sistem yang berbasis *client server*. Selain itu, metode pengamanan seperti melakukan *packet sniffing* harus dilakukan untuk memantau lalu lintas jaringan pada sistem IM. Pada proses awal komunikasi, user IM harus membangun sebuah *session* dan sistem akan melakukan autentikasi berdasarkan pada identitas user yang sudah terdaftar pada IM Server. Pada sistem ini semua pesan hasil komunikasi akan disimpan ditempat khusus pada server IM. Sistem IM di atas dapat diilustrasikan pada Gambar 2.3.



Gambar 2.3 Lingkungan sistem IM model Donal C. (2007)

2. *Peer-to-Peer*

Pada penelitian (Tahsin, Choudhury, and Rahman 2008) komunikas mobile berbasis *peer-to-peer* (P2P) yang diterapakan pada limgkungan protokol JXTA/JXME. Pada model yang dibangun, sebuah perangkat (*device*) yang ingin melakukan komunikasi harus mengirimkan HTTP Request dan melacak relay yang akan mendukung komunikasi pada jaringan. Sedangkan *relay peer* berfungsi untuk menghubungkan antara jaringan JXTA dan *peers*. Pertukaran pesa yang dikirimkan oleh client harus melalui sebuah *relay peer* sebagai representasi JXME *client* saat berkomunikasi pada jaringan JXTA. Model ini dapat diilutrasikan pada Gambar 2.4



Gambar 2.4 Lingkungan sistem IM Model Tahmina Tahsin (2008)

Selain pengembangan yang berorientasi arsitektur, beberapa aplikasi IM mulai mengarah pada aspek keamanan data yang dikirikan melalui jaringan publik. Pada (Techcrunch.com and Russell 2014) membahas mengenai telegram yang merupakan salah satu aplikasi IM yang memiliki fitur keamanan pesan (*secure chat*) selama komunikasi. Telegram disebut lebih aman daripada IM yang banyak digunakan saat ini seperti WhatsApp and LINE karena percakapannya dienkripsi menggunakan AES-256 dan menggunakan protokol Mtproto yang dikembangkan

sendiri. Selain itu data hasil percakapan pada telegram akan dihapus secara otomatis pada aplikasi setelah percakapan selesai.

2.2.2 Kriptografi

Menurut (Schneier 1996), Kriptografi (*Cryptography*) merupakan istilah kata yang berasal dari dua buah kata dalam bahasa Yunani yaitu *crypto* dan *graphia* yang berarti penulisan rahasia atau ilmu yang mempelajari tentang penulisan secara rahasia. Kriptografi merupakan teknik yang digunakan untuk merubah informasi yang awalnya berbentuk teks asli (*plaintext*) menjadi kode tertentu yang bersifat rahasia (*chipertext*). Teknik yang digunakan untuk mengubah pesan asli menjadi kode teretntu pada sebuah informasi disebut dengan enkripsi (*encryption*), sedangkan teknik yang digunakan untuk mengembalikan *chipertext* menjadi bentuk teks asli (*plaintext*) disebut dengan dekripsi (*decryption*).

Sedangkan menurut (Menezes et al, 1996) kriptografi merupakan sekumpulan teknik matematika yang berhubungan dengan metode pengamanan data atau informasi yang meliputi kerahasiaan, data integrity, otentikasi pengirim/penerima data serta otentikasi data yang ditransaksikan. Dalam proses enkripsi dan dekripsi informasi, digunakan sebuah algoritma kriptografi yang disebut *cipher.* Algoritma kriptografi bekerja berdasarkan sebuah kunci (*key*). *Key* sangat penting dalam proses enkripsi dan dekripsi sebuah informasi, karena informasi yang dienkripsi menggunakan *key* yang berbeda akan menghasilkan c*iphertext* yang berbeda pula begitu juga dengan *ciphertext* yang didekripsi menggunakan kunci yang berbeda tidak akan menghasilkan teks yang sama.

Ada 4 (empat) tujuan mendasar dari ilmu kriptografi ini yang juga merupakan

aspek keamanan informasi yaitu:

1. Kerahasiaan (*confidentiality*), adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak. Di dalam kriptografi, layanan ini direalisasikan dengan menyandikan pesan menjadi cipherteks.
2. Integritas data (*data integrity*), adalah layanan yang menjamin bahwa pesan masih asli atau belum pernah dimanipulasi selama pengiriman. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan: "Apakah pesan yang diterima masih asli atau tidak mengalami perubahan?". Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubstitusian data lain ke dalam pesan yang sebenarnya.
3. Otentikasi (*authentication*), adalah layanan yang berhubungan dengan identifikasi, baik mengindentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication* atau *entity authentication*) maupun mengindentifikasi kebenaran sumber pesan (*data origin authencation*). Pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga dapat memastikan sumber pesan. Pesan yang dikirim melalui saluran komunikasi juga harus diotentikasi asalnya. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan: "Apakah pesan yang diterima benar-benar berasal dari pengirim yang benar?".
4. Nirpenyangkal (*non-repudiation*), adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan

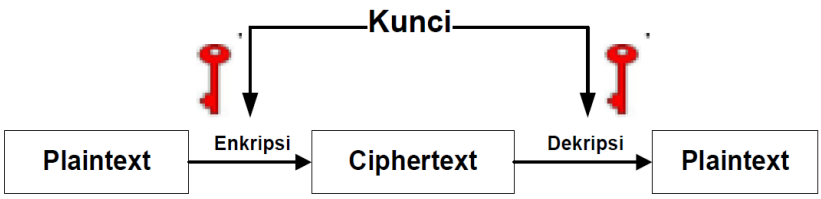
menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

2.2.3 Algoritma pada Kriptografi

Berdasarkan jenis kunci yang digunakan dalam proses enkripsi dan dekripsi pesan, algoritma kriptografi dibagi menjadi dua jenis yaitu algoritma simetri (*private key algorithm*) dan algoritma asimetri (*public key algorithm*).

2.2.3.1 Algoritma Simetri

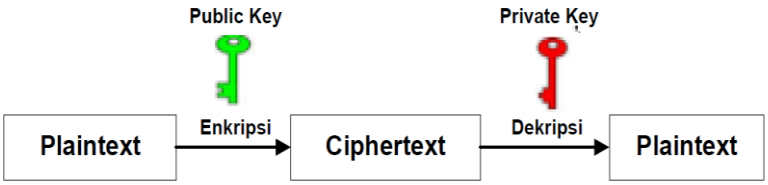
Algoritma simetri merupakan algoritma yang dalam proses enkripsi dan dekripsinya menggunakan kunci yang sama. Jenis algoritma ini disebut sebagai algoritma kriptografi konvensional dan sudah mengalami perkembangan yang sangat lama dalam dunia kriptografi. Algoritma kriptografi simetris dibagi menjadi 2 yaitu *Stream Chiper* dimana proses penyandian berorientasi pada satu *bit*/ satu *byte* data dan *Block Chiper* yang penyandiannya berorientasi pada sekumpulan *bit* atau *byte*. Contoh algoritma simetri antara lain: *Data Encryption Standard* (DES), Rijndael (AES), Blowfish dll. Cara kerja algoritma simetris diilustrasikan seperti pada Gambar 2.5.



Gambar 2.5 Cara kerja algoritma simetri

2.2.3.2 Algoritma Asimetri

Algoritma Asimetri merupakan algoritma kriptografi yang menggunakan kunci yang berbeda dalam proses enkripsi dan dekrispsi. Algoritma Asimetri biasanya dikenal dengan *public key algorithm* dimana kunci yang digunakan dalam proses enkripsi adalah kunci pubik (*public key)* yang disebarkan dan dapat diketahui oleh banyak orang, sedangkan proses dekripsi data akan menggunakan kunci pribadi *(private key)* yang hanya diketahui oleh orang yang memiliki kewenangan tertentu atau hanya diketahui oleh pemiliknya. Algoritma kunci publik juga telah mengalami perkembangan yang cukup pesat hingga saat ini. Beberapa algoritma asimetri yang banyak digunakan saat ini antara lain RSA (*Rivest-Shamir- Adlemen*) dan ECC (*Elliptic Curve Cryptography*). Cara kerja algoritma kriptografi asimetri diilutrasikan seperti Gambar 2.6.



Gambar 2.6 Cara kerja algoritma asimetri

Dalam implementasinya, algoritma asimetris dapat dibagi menjadi 3 (tiga) kategori, yaitu:

1. Kerahasiaan data. Seperti pada kriptografi kunci simetri, kriptografi kunci publik dapat digunakan untuk menjaga kerahasiaan data (*provide confidentiality*/*secrecy*) melalui mekanisme enkripsi dan dekripsi. Contoh algoritma untuk aplikasi ini adalah *RSA*, *Knapsack*, *Rabin*, *ElGamal*, *Elliptic Curve Cryptography* (*ECC*).

2. Tanda-tangan digital. Tanda-tangan digital (*digital signature*) dengan menggunakan algoritma kriptografi kunci publik dapat digunakan untuk membuktikan otentikasi pesan maupun otentikasi pengirim (*provide authentication*). Contoh algoritmanya untuk aplikasi ini adalah *RSA*, *DSA* dan *ElGamal*.
3. Pertukaran kunci (*key exchange*). Algoritma kriptografi kunci publik dapat digunakan untuk pengiriman kunci simetri (*session keys*). Contoh algoritmanya adalah *RSA* dan *Diffie-Hellman*.

2.2.3.3 Perbandingan Algoritma Simetri dan Asimetri

Algoritma simetri dan asimetri sama-sama mempunyai kelebihan dan kekurangan masing-masing dalam penerapan dalam aplikasi. Kelebihan kriptografi kunci simetri antara lain:

1. Algoritma kriptografi simetri dirancang sehingga proses enkripsi dan dekripsi membutuhkan waktu yang singkat.
2. Ukuran kunci simetri relatif pendek. Algoritma simetri dapat digunakan membangkitkan bilangan acak.
3. Algoritma kunci simetri dapat disusun untuk menghasilkan *cipher* yang lebih kuat.
4. Autentikasi pengiriman data langsung diketahui dari *ciphertext* yang diterima, karena kunci hanya diketahui pengirim dan penerima saja.

Selain memiliki kelebihan, algoritma simetri juga memiliki kekurangan yaitu:

1. Kunci simetri harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci.

2. Kunci harus sering diubah, mungkin setiap komunikasi.

Sedangkan algoritma asimetri yang bekerja dengan pasangan kunci yang berbeda yaitu kunci publik dan kunci privat memiliki kelebihan antara lain:

1. Hanya kunci privat yang perlu dijaga kerahasiaannya oleh setiap entitas yang berkomunikasi. Tidak ada kebutuhan untuk mengirim kunci privat sebagaimana kriptografi kunci simetri.
2. Pasangan kunci publik tidak perlu diubah, bahkan dalam periode waktu yang panjang.
3. Dapat digunakan dalam pengiriman kunci simetri.
4. Beberapa algoritma kunci publik dapat digunakan untuk member tanda tangan digital pada data.

Selain kelebihan yang telah disebutkan di atas, algoritma asimetri juga memiliki kekurangan antara lain:

1. Enkripsi dan dekripsi data umumnya lebih lambat daripada sistem kriptografi simetri, karena enkripsi dan dekripsi menggunakan bilangan yang besar dan melibatkan operasi perpangkatan yang besar.
2. Ukuran *ciphertext* lebih besar dari *plaintext*.
3. Ukuran kunci relatif lebih besar daripada ukuran kunci simetri.
4. Karena kunci publik diketahui secara luas dan dapat digunakan setiap orang maka *ciphertext* tidak memberikan informasi mengenai otentikasi pengirim.

2.2.4 Keaslian Data Digital

Pada proses komunikasi data, sebuah pesan atau dokumen lainnya bisa disebut otentik jika data tersebut asli dan berasal dari sumber yang sesungguhnya. Autentikasi merupakan suatu prosedur yang dilakukan untuk memverifikasi keaslian pesan yang diterima dengan menerapkan metode autentikasi tertentu misalnya dengan metode tanda tangan digital.

Pada proses komunikasi IM, transaksi data akan melibatkan dua pihak yaitu pihak pengirim (*sender*) dan penerima (*receiver*). Dalam sebuah proses transaksi data melalui jaringan publik, data yang dikirim adalah informasi atau pesan yang hanya boleh diketahui oleh kedua belah pihak. Namun, permasalahan akan timbul jika pihak ada pihak ketiga melakukan tindakan penyadapan dan melakukan perubahan pada pesan yang dikirim oleh pengirim atau berpura-pura sebagai pengirim asli agar dapat memperoleh penting dari pihak penerima pesan. Tindakan semacam itu lebih dikenal nama *Man in The Middle Attack* (MITM), tindakan ini akan sangat merugikan para pengguna IM tersebut. Sebuah layanan bisa dikatakan aman jika memenuhi beberapa unsur keamanan yang meliputi: kerahasiaan data, integritas data, otentikasi, Anti Penyangkalan (*Non Repudiation*) dan akses kontrol (A. Behrouz. 2008.) Di dalam penerapan tanda tangan digital, teknik pemberian tanda tangan pada pesan pesan dapat dilakukan dengan salah satu dari dua cara, yaitu:

1. Enkripsi pesan

   Mengenkripsi pesan dengan sendirinya juga menyediakan ukuran Autentikasi. Pesan yang terenkripsi juga sudah menyatakan bahwa pesan tersebut telah ditandatangani.

2. Tanda tangan digital dengan fungsi hash

   Tanda tangan digital dengan menggunakan fungsi hash. Pengirim pesan mula-mula menghitung *message digest* dari pesan. *Message digest* diperoleh dengan mentransformasikan pesan dengan menggunakan fungsi hash 1 arah. Selanjutnya *message digest* dienkripsi dengan algoritma kriptografi kunci publik dan menggunakan kunci privat pengirim. Hasil enkripsi inilah yang disebut dengan tanda tangan digital. Selanjutnya tandatangan digital dilekatkan dengan ke pesan dengan cara menyambung (*append*) lalu dikirim melalui saluran komunikasi. Kemudian di tempat penerima, tandatangan digital dibuktikan keotentikannya dengan melakukan verifikasi pesan.

2.2.5 Tanda Tangan Digital (*Digital Signature*)

Tanda tangan digital merupakan rangkaian *byte* yang ditambahkan pada paket (data) tertentu yang digunakan untuk melakukan autentikasi sebuah konten digital. Tanda tangan digital ini diciptakan melalui penggunaan pasangan kunci yaitu kunci privat *(private key)* dan kunci publik *(publik key)* dengan tujuan untuk memberikan aspek keamanan pada proses pertukaran data antara pengirim dan penerima (Menezes et al. 1996).

Tanda tangan digital merupakan cara yang tepat untuk mengetahui identitas seseorang pada sebuah pesan. Metode tanda tangan digital lebih *reliable*

dibandingkan dengan dengan metode tanda tangan biasa yang lebih mudah dipalsukan (Grabbe, 1998). Dalam penerapannya saat ini, skema tanda tangan digital dapat digunakan untuk hal-hal sebagai berikut:

a. Integritas Data (*Data integrity*)

Memastikan bahwa data tidak mengalami perubahan selama berada pada jalur pengiriman. Pengubahan data ini bisa dilakukan oleh orang yang tidak berhak untuk tujuan tertentu.

b. Autentikasi

Memastikan bahwa data bersumber dari pihak yang benar, sesuai dengan yang yang tertera pada sebuah pesan.

c. Anti Penyangkalan (*Non Repudiation*)

Memastikan bahwa salah satu pihak tidak dapat menyangkal bahwa pesan yang dikirm sebelumnya berasal darinya sesuai tanda tangan yang terkandung pada pesan. Aspek ini cukup penting dalam keamanan informasi (Menezes et al. 1996).

Peran utama dengan menerapkan metode ini ialah menyertakan tanda tangan pada sebuah pesan sehingga pesan tersebut dapat diketahui keaslian identitas pengirim sehingga data bisa dianggap memiliki integritas. Tanda tangan ini bisa disusun dari informasi-informasi tertentu yang dianggap unik dimana setiap orang akan memiliki tanda tangan yang berbeda sehingga dapat diketahui keaslian dari sebuah pesan yang dikirim

2.2.6 Algoritma RSA

Algoritma ini dipublikasikan pada tahun 1977 oleh Ron Rivest, Adi Shamir dan Len Adleman di MIT. RSA adalah singkatan dari nama para penemunya, yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman. RSA adalah salah satu algoritma penyandian yang paling banyak mengundang kontroversi, selain DES. Sejauh ini belum seorang pun yang berhasil menemukan lubang sekuriti pada DES dan RSA, tetapi tak seorang pun juga yang berhasil memberikan pembuktian ilmiah yang memuaskan dari keamanan kedua teknik sandi ini. Untuk menyandi informasi dan untuk menerjemahkan pesan tersandi sebuah algoritma penyandian memerlukan sebuah data biner yang disebut kunci. Tanpa kunci yang cocok orang tidak bisa mendapatkan kembali pesan asli dari pesan tersandi. Pada AES digunakan kunci yang sama untuk menyandi (enkripsi) maupun untuk menterjemahan (dekripsi), sedangkan RSA menggunakan dua kunci yang berbeda. Isitilahnya, AES disebut sistem sandi simetris sementara RSA disebut sistem sandi asimetris.

Sistem sandi simetris cenderung jauh lebih cepat sehingga lebih disukai oleh kalangan industri. Kejelekannya, pihak-pihak yang ingin berkomunikasi secara privat harus punya akses ke sebuah kunci AES bersama. Walaupun biasanya pihak-pihak yang terkait sudah saling percaya, skema ini memungkinkan satu pihak untuk memalsukan pernyataan dari pihak lainnya. RSA yang menggunakan algoritma asimetrik mempunyai dua kunci yang berbeda, disebut pasangan kunci (key pair) untuk proses enkripsi dan dekripsi. Kunci-kunci yang ada pada pasangan kunci mempunyai hubungan secara matematis, tetapi tidak dapat dilihat secara komputasi untuk mendeduksi kunci yang satu ke pasangannya. Algoritma ini disebut kunci

publik, karena kunci enkripsi dapat disebarkan. Orang-orang dapat menggunakan kunci publik ini, tapi hanya orang yang mempunyai kunci privat sajalah yang bisa mendekripsi data tersebut.

Tingkat keamanan algoritma penyandian RSA sangat bergantung pada ukuran kunci sandi tersebut (dalam bit), karena makin besar ukuran kunci, maka makin besar juga kemungkinan kombinasi kunci yang bisa dijebol dengan metode mengencek kombinasi satu persatu kunci atau lebih dikenal dengan istilah brute force attack. Jika dibuat suatu sandi RSA dengan panjang 256 bit, maka metode brute force attack akan menjadi tidak ekonomis dan sia-sia dimana para hacker pun tidak mau/sanggup untuk menjebol sandi tersebut.

2.2.6.1 Proses pembangkitan kunci

Pembangkitan kunci publik dan kunci privat pada skema tanda tangan RSA dapat dilakukan melalui langkah-langkah berikut ini:

a. Pilih dua bilangan prima p dan q secara acak , $p \neq q$. Bilangan ini harus cukup besar (minimal 100 digit).
b. Hitung $N = pq$. Bilangan N disebut *parameter sekuriti*.
c. Hitung $\varphi = (p-1)(q-1)$.
d. Pilih bilangan bulat (*integer*) antara satu dan φ ($1 < e < \varphi$) yang tidak mempunyai faktor pembagidari φ.
e. Hitung d hingga $d\,e \equiv 1 \pmod{\varphi}$.

Setelah melalu cara ini, maka kita akan mendapatkan kunci publik dan kunci privat. Kunci publik terdiri dari dua elemen, yaitu:

- N, merupakan modulus yang digunakan

- *e*, eksponen publik atau eksponen enkripsi

dan kunci privat, yang terdiri dari:

- *N*, merupakan modulus yang digunakan, sama seperti pada kunci publik
- d, eksponen pribadi atau eksponen deskripsi yang harus dijaga kerahasiaanya

Nilai *p* dan *q* sebaiknya dibuang atau dijaga kerahasiaannya, karena terdapat *N* dimana *p* dan *q* adalah faktor pembagi dari *N*. Walaupun bentuk ini memperbolehkan dekripsi secara cepat dan *signing* menggunakan *Chinese Remainder Theorem* (CRT), hal ini mejadi lebih tidak aman karena bentuk ini memperbolehkan *side channel attacks*. *Side channel attacks* adalah sebuah serangan yang berdasarkan informasi yang dikumpulkan dari implementasi fisik (atau kelemahan secara fisik) dari sebuah sistem kriptografi, dibanding dengan kelemahan teoritis dari algoritmanya sendiri. Sebagai contohnya, faktor-faktor kurun waktu dari informasi, konsumsi tenaga, bahkan suara yang ditimbulkan dapat membantu mempermudah informasi yang bisa diambil untuk menjebol sistem tersebut.

2.2.6.2 Proses Enkripsi Pesan

Misalkan pada suatu kasus si A ingin mengirim pesan *m* kepada si B. A mengubah *m* menjadi angka $n < N$, menggunakan protokol yang sebelumnya telah disepakati dan dikenal sebagai *padding scheme*. *Padding scheme* harus dibangun secara hati-hati sehingga tidak ada nilai dari *m* yang menyebabkan masalah keamanan. Contohnya, jika kita ambil contoh sederhana dari penampilan ASCII dari *m* dan menggabungkan bit-bit secara bersama-sama akan menghasilkan *n*,

kemudian pessan yang berisi ASCII tunggal karakter NULL (nilai numeris 0) akan menghasilkan $n = 0$, yang akan menghasilkan *ciphertext* 0 apapun itu nilai dari *e* dan *N* yang digunakan. Maka A mempunyai nilai *n* dan mengetahui *N* dan *e*, yang telah diumumkan oleh B. A kemudian menghitung *ciphertext c* yang terkait pada *n*:

$$\boldsymbol{c = n^e \bmod N}$$

Perhitungan tersebut dapat diselesaikan dengan menggunakan metode *exponentation by squaring*, yaitu sebuah algoritma yang dipakai untuk komputasi terhadap sejumlah nilai integer yang besar dengan cepat. Kemudian A mengirimkan nilai C kepada B.

2.2.6.3 Proses dekripsi pesan

B sudah menerima C dari A, dan mengetahui kunci privat yang digunakan B. B kemudian mengembalikan nilai *n* dari C dengan langkah-langkah sebagai berikut:

$$\boldsymbol{n = c^d \bmod N}$$

Perhitungan diatas akan menghasilkan *n*, dengan begitu B dapat mengembalikan pesan semula *m*. Prosedur dekripsi bekerja karena

$$\boldsymbol{c^d \equiv (n^e)^d \equiv n^{ed} \ (\bmod N)}$$

Kemudian, karena $ed \equiv 1 \pmod{p-1}$ dan $ed \equiv 1 \pmod{q-1}$, hasil dari *Fermat's little theorem*.

$$\boldsymbol{n^{ed} \equiv n \ (\bmod p)} \text{ dan } \boldsymbol{n^{ed} \equiv n \ (\bmod q)}$$

Karena *p* dan *q* merupakan bilangan prima yang berbeda, mengaplikasikan C*hinese remainder theorem* akan menghasilkan dua macam kongruen

$$\boldsymbol{n^{ed} \equiv n \ (\bmod pq) \text{ dan } c^d \equiv n \ (\bmod N)}$$

2.2.6.4 Keamanan RSA

Keamanan dari sistem kriptografi RSA adalah didasari oleh dua problem matematika:

- Problem dalam faktorisasi bilangan berjumlah banyak
- Problem RSA, yaitu mencari modulo akar *e*n

dari sebuah bilangan komposit *N* yang faktor-faktornya tidak diketahui Proses dekripsi penuh dari sebuah *ciphertext* RSA dianggap sesuatu hal yang tidak mudah karena kedua problem ini diasumsikan sulit.

2.2.6.5 Tanda Tangan Digital pada RSA

Proses pemberian tanda tangan digital pada algoritma RSA akan melalui langkah-langkah sebagai berikut:

1. Hitung *(~m) = R(m)*, sebuah integer pada kisaran *[0, n-1]*
2. Hitung *s = (~m)d mod n.*
3. Tanda tangan A untuk pesan m adalah s

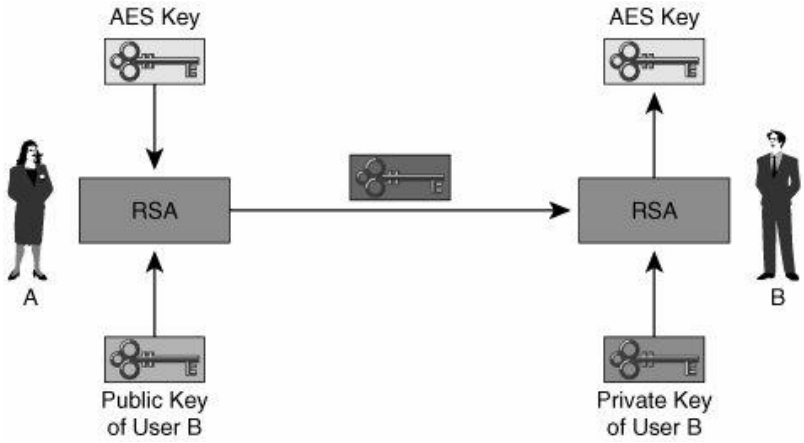
Proses yang dilakukan untuk memverifikasi tanda tangan digital adalah dengan cara sebagai berikut:

1. Ambil kunci publik yaitu *n* dan *e*
2. Hitung *~m = se mod n*
3. Dengan mengetahui *~m*, maka dapat diketahui nilai *m*.

2.2.6.6 *Key Exchange* pada RSA

Pada pertukaran kunci dengan menggunakan algoritma RSA, kunci rahasia ini di *enkripsi* terlebih dahulu bersama dengan kunci publik. Hanya penerima pesan

yang diinginkan saja yang dapat *mendekripsi* kunci rahasia tersebut karena untuk *mendekripsikannya* dibutuhkan kunci privat yang dimiliki penerima pesan. Oleh karena itu, pihak lain yang mendapatkan kunci yang sudah di *enkripsi* pun tidak dapat *mendekripsikannya*. Penggunaan algoritma RSA di dalam pertukaran kunci ini digunakan dalam Encrypting File System (EFS) di sistem operasi *windows*. Proses pertukaran kunci dengan menggunanakan algoritma RSA dapat dilihat pada gambar berikut:



Gambar 2.7 *Key Exchange* AES key dengan RSA

2.2.7 Algoritma AES

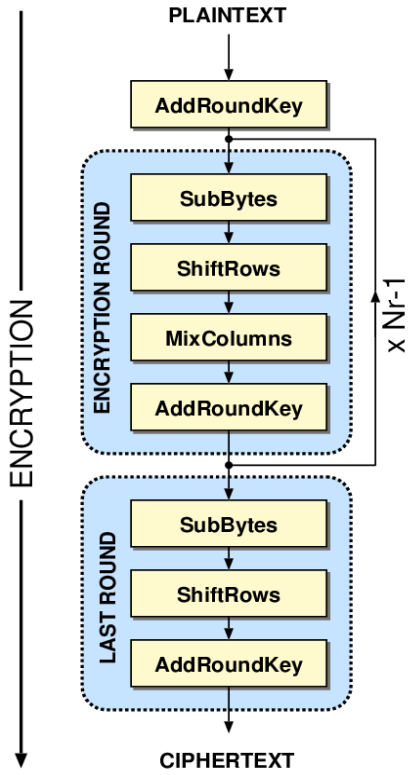
AES dipublikasikan oleh NIST (*National Institute of Standard and Technology*) pada tahun 2001 yang digunakan untuk menggantikan algoritma DES yang semakin lama semakin mudah untuk membobol kuncinya. AES diperoleh dari hasil kompetisi yang diadakan NIST pada tahun 1997, dibuat oleh Dr. Vincent Rijment dan Dr. Joan Daemen, sebagai pemenang.

Teknik enkripsi ini termasuk jenis *block cipher* seperti halnya dengan DES. Perbedaan utama antara teknik enkripsi AES dan teknik enkripsi DES adalah AES

juga menggunakan substitusi (menggunakan S-boxes) secara langsung terhadap naskah, sedangkan substitusi S-box digunakan DES hanya dalam fungsi *cipher f* yang hasilnya kemudian dioperasikan terhadap naskah menggunakan *exclusive or*, jadi DES tidak menggunakan substitusi secara langsung terhadap naskah. AES juga menggunakan kunci enkripsi yang lebih besar yaitu 128 bit, 192 bit, atau 256 bit.

2.2.7.1 Proses Enkripsi

Secara garis besar, enkripsi AES. Input berupa naskah asli sebesar 128 bit, sedangkan o`utput adalah naskah acak sebesar 128 bit. Setiap transformasi dilakukan secara langsung terhadap naskah, mulai dengan transformasi *AddRoundKey*(0). Jadi setiap transformasi harus mempunyai *inverse* agar naskah acak dapat didekripsi, pada putaran 1 sampai dengan *Nr* -1, transformasi *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* dilakukan terhadap naskah. Pada putaran terahir (*Nr*), transformasi *SubBytes*, *ShiftRows* dan *AddRoundKey* dilakukan terhadap naskah (transformasi *Mix-Columns* tidak dilakukan). Total ada *Nr* putaran. lebih jelasnya di paparkan pada gambar 2.8 berikut ini.



Gambar 2.8 Enkripsi AES

Jumlah putaran (*Nr*) tergantung pada besar kunci yang digunakan. Tabel 2.2 menunjukkan jumlah putaran (*Nr*) untuk kunci sebesar 128 bit, 192 bit dan 256 bit. Jadi untuk kunci sebesar 128 bit, besar kunci (*Nk*) adalah 4 *word* (setiap *word* mempunyai 32 bit), besar blok (*Nb*) adalah 4 *word*, dan jumlah putaran (*Nr*) adalah 10.

Tabel 2.2 Tebel jumlah putaran

Kunci	Besaran Kunci dalam *words*	Besar Block (*Nb*) dalam *words*	Jumlah Putaran (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Sumber: en.wikipedia.org/wiki/Advanced_Encryption_Standard

a. *AddRoundKey*

Pada proses enkripsi dan dekripsi AES proses *AddRoundKey* sama, sebuah *round key* ditambahkan pada *state* dengan operasi XOR. Setiap *round key* terdiri dari Nb *word* dimana tiap *word* tersebut akan dijumlahkan dengan *word* atau kolom yang bersesuaian dari *state* sehingga

$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c},] \oplus [w_{round*Nb+c}]$ untuk $0 \leq c \leq Nb$

[wi] adalah *word* dari *key* yang bersesuaian dimana i = round*Nb+c. Transformasi *AddRoundKey* pada proses enkripsi pertama kali pada *round* = 0 untuk *round* selanjutnya *round* = *round* + 1, pada proses dekripsi pertama kali pada *round* = 14 untuk *round* selanjutnya *round* = *round* - 1. Berikut contoh *addroundkey* dengan sebuah *state* bernilai: 0x32/0x43/0xF6/0xA8 dengan sebuah *chipperkey* 0x2B/0x7E/0x15/0x16 maka

$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}]$ = [32,43,F6,A8] $\oplus$ [2B,7E,15,16] hasilnya adalah 0x19/0x3D/0xE3/0xBE.

b. *SubBytes*

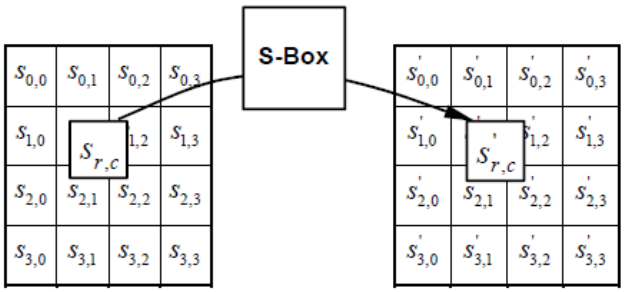
SubBytes merupakan transformasi *byte* dimana setiap elemen pada *state* akan dipetakan dengan menggunakan sebuah tabel substitusi (S-Box). Tabel substitusi S-Box akan dipaparkan dalam Tabel 2.3

Tabel 2.3 Model S-box Rjindael

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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

Sumber: https://en.wikipedia.org/wiki/Rijndael_S-box

Untuk setiap *byte* pada *array state*, misalkan *S*[*r*, *c*] = *xy*, yang dalam hal ini *xy* adalah *digit* heksadesimal dari nilai *S*[*r*, *c*], maka nilai substitusinya, dinyatakan dengan *S*'[*r*, *c*], adalah elemen di dalam tabel subtitusi yang merupakan perpotongan baris *x* dengan kolom *y*. Gambar 3 mengilustrasikan pengaruh pemetaan *byte* pada setiap *byte* dalam *state*.



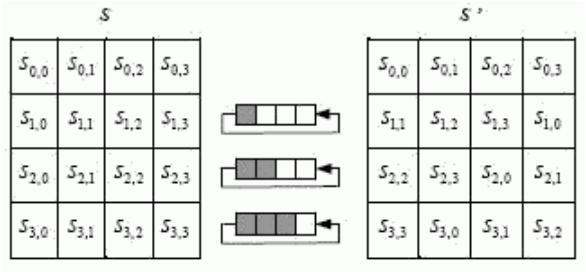
Gambar 2.9 Transformasi SubByte dengan S-Box

Berikut contoh perhitungan SubByte menggunakan S-Box:

Misalkan untuk kolom [$s_{0,0}$, $s_{1,0}$, $s_{2,0}$, $s_{3,0}$,] berisi *state*: 0x19/0x3D/0xE3/0xBE maka nilai *subbyte* adalah $s_{0,0} = 1{,}9$ menjadi D4 $s_{1,0} = 3\text{D}$ menjadi 27 $s_{2,0} = \text{E3}$ menjadi 11 dan $s_{3,0} = \text{BE}$ menjadi AE

c. *Shiftrows*

Transformasi *Shiftrows* pada dasarnya adalah proses pergeseran *bit* dimana *bit* paling kiri akan dipindahkan menjadi *bit* paling kanan (rotasi *bit*). Proses pergeseran *Shiftrow* ditunjukkan dalam Gambar 4 berikut:



Gambar 2.10 *Shitf row* Transformasi

Berikut contoh perhitungan *Shiftrow*:

Misalkan sebuah tabel *S* berisi *state* kemudian di shiftrow menjadi S' seperti berikut ini:

S

D4	E0	B8	1E
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

S'

D4	E0	B8	1E
BF	B4	41	27
5D	52	11	98
30	AE	F1	E5

d. *MixColumns*

MixColumns mengoperasikan setiap elemen yang berada dalam satu kolom pada state. Secara lebih jelas, transformasi *mixcolumns* dapat dilihat pada perkalian matriks berikut ini:

Gambar 2.11 *Mixcolumn* AES proses

Hasil dari perkalian matriks diatas dapat dianggap seperti perkalian yang ada di bawah ini :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{02\}.s_{0,c}) \oplus (\{03\}.s_{1,c}) \oplus s_{2.c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\}.s_{1,c}) \oplus (\{03\}).s_{2.c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\}).s_{2.c} \oplus (\{03\}.s_{3,c})$$

$$s'_{3,c} = (\{03\}.s_{0,c}) \oplus s_{1,c} \oplus s_{2.c} \oplus (\{02\}).s_{3,c}$$

Contoh perkalian matrix $[s_{0,0}, s_{1,0}, s_{2,0}, s_{3,0},]$ dengan s*tate* 0xD4/0xBF/0x5D/0x30 di konversi dari bilangan hexadecimal ke biner dengan operasi *Rijndael Galois Field* dimana perkalian dengan 1 artinya nilai tidak berubah, perkalian dengan 2 artinya menambah 1 byte dengan geser ke kiri, perkalian dengan 3 artinya menambah 1 byte dengan geser ke kiri

kemudian melakukan operasi xor dengan nilai awal sebelum digeser. Sebagai catatan apabila nilai byte lebih besar dari 0xFF setelah digeser, maka harus dilakukan xor dengan 0x11B. Sehingga hasil perkalian ini menjadi:

$$S'0,0 = (\{2\}.1101\ 0100) \oplus (\{3\}.1011\ 1111) \oplus (0101\ 1101) \oplus (0011\ 0000)$$

$$S'1,0 = (1101\ 0100) \oplus (\{2\}.1011\ 1111) \oplus (\{3\}0101\ 1101) \oplus (0011\ 0000)$$

$$S'2,0 = (1101\ 0100) \oplus (1011\ 1111) \oplus (\{2\}.0101\ 1101) \oplus (\{3\}.0011\ 0000)$$

$$S'3,0 = (\{3\}.1101\ 0100) \oplus (1011\ 1111) \oplus (0101\ 1101) \oplus (\{2\}.0011\ 0000)$$

Setelah di lakukan perkalian *Rijndael Galois Field* maka hasilnya:

$$S'0,0 = 1011\ 0011 \oplus 1101\ 1010 \oplus 0101\ 1101 \oplus 0011\ 0000$$

$$S'1,0 = 1101\ 0100 \oplus 0110\ 0101 \oplus 1110\ 0111 \oplus 0011\ 0000$$

$$S'2,0 = 1101\ 0100 \oplus 1011\ 1111 \oplus 1011\ 1010 \oplus 0101\ 0000$$

$$S'3,0 = 0110\ 0111 \oplus 1011\ 1111 \oplus 0101\ 1101 \oplus 0110\ 0000$$

Sehingga hasil perkalian tersebut menghasilkan nilai $[s'_{0,0}, s'_{1,0}, s'_{2,0}, s'_{3,0}]$ adalah 0x04/0x66/0x81/0xE5 setelah di konversi kembali ke hexadesimal

e. *Key Schedule*

Key schedule digunakan pada saat penjadwalan kunci dengan memperluas kunci yang pendek disetiap putaran *Nr,* kunci tersebut di dapat dari hasil transformasi antara kunci awal dengan tabel S-Box dan Rcon. Perhitungan Nilai kolom awal untuk kunci baru didapat dari rotasi 1 byte kolom terakhir kemudian di subbyte dengan S-Box lalu di xor dengan kolom pertama kunci awal dan kolom pertama Rcon, kolom 2-4 hasilnya di merupakan xor antara nilai kunci baru per kolomnya dengan kolom 2-4 kunci lama.

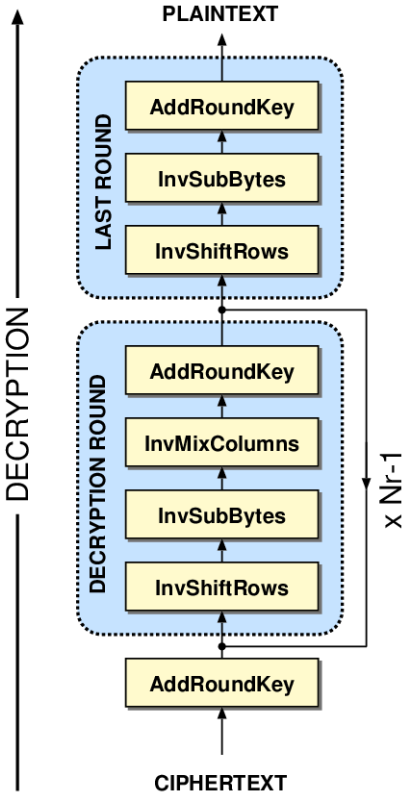
Tabel 2.4 Rcon AES tabel

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Sumber: Developer Club

2.2.7.2 Proses Dekripsi AES

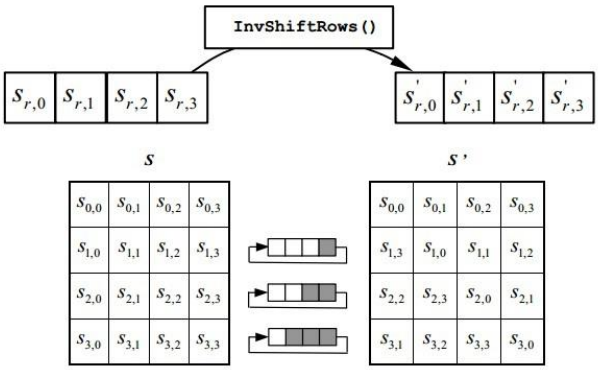
Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi byte yang digunakan pada *invers cipher* adalah *InvShiftRows, InvSubBytes, InvMixColumns,* dan A*ddRoundKey*. Algoritma dekripsi dapat dilihat pada skema berikut ini :



Gambar 2.12 Proses Dekripsi AES

1. *InvShiftRows*

InvShiftRows adalah transformasi byte yang berkebalikan dengan transformasi *ShiftRows*. Pada transformasi *InvShiftRows*, dilakukan pergeseran *bit* ke kanan sedangkan pada *ShiftRows* dilakukan pergeseran *bit* ke kiri. Ilustrasi transformasi *InvShiftRows* terdapat pada gambar :



Gambar 2.13 *Invshiftrows AES*

2. *InvSubBytes*

InvSubBytes juga merupakan transformasi *bytes* yang berkebalikan dengan transformasi *SubBytes*. Pada *InvSubBytes*, tiap elemen pada state dipetakan dengan menggunakan tabel *Inverse S-Box*. Tabel *Inverse S-Box* akan ditunjukkan dalam Tabel berikut:

Tabel 2.5. *Inv S-box AES*

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

sumber: https://en.wikipedia.org/wiki/Rijndael_S-box

3. *InvMixColumns*

Setiap kolom dalam *state* dikalikan dengan matrik perkalian dalam AES. Perkalian dalam matrik dapat dituliskan dengan rumus:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E\ 0B\ 0D\ 09 \\ 09\ 0E\ 0B\ 0D \\ 0D\ 09\ 0E\ 0B \\ 0B\ 0D\ 09\ 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Hasil dari perkalian matriks diatas dapat dianggap seperti perkalian sebagai berikut:

$$s'_{0,c} = (\{0e\}.s_{0,c}) \oplus (\{0b\}.s_{1,c}) \oplus (\{0d\}.s_{2,c}) \oplus (\{09\}.s_{3,c})$$

$$s'_{1,c} = (\{09\}.s_{0,c}) \oplus (\{0e\}.s_{1,c}) \oplus (\{0b\}.s_{2,c}) \oplus (\{0d\}.s_{3,c})$$

$$s'_{2,c} = (\{0d\}.s_{0,c}) \oplus (\{09\}.s_{1,c}) \oplus (\{0e\}.s_{2,c}) \oplus (\{0b\}.s_{3,c})$$

$$s'_{3,c} = (\{0b\}.s_{0,c}) \oplus (\{0d\}.s_{1,c}) \oplus (\{09\}.s_{2,c}) \oplus (\{0e\}.s_{3,c})$$

Sehingga didapat nilai invmixcolumnya.

AES telah menggantikan DES sebagai standard enkripsi untuk keperluan instansi pemerintahan Amerika Serikat. Ada yang meragukan ketangguhan AES karena semua transformasi yang dilakukan dalam enkripsi AES mempunyai rumus aljabar yang elegan. Akan tetapi rumus yang elegan tidak berarti parameter kunci dapat dipecahkan dengan mudah dari persamaan. Kriptografi *public key* menggunakan rumus aljabar yang elegan, namun parameter kunci privat sulit untuk dipecahkan. Kembali ke AES, karena enkripsi adalah manipulasi berbagai bit, pemecahan parameter kunci dapat dirumuskan sebagai masalah SAT dalam aljabar Boolean, namun masalah SAT adalah masalah yang bersifat NP-*complete* yang pada umumnya tidak dapat dipecahkan secara efisien (untuk pemecahan kunci enkripsi AES, ruang pencarian terlalu besar sehingga dalam prakteknya pemecahan tidak dapat dilakukan).

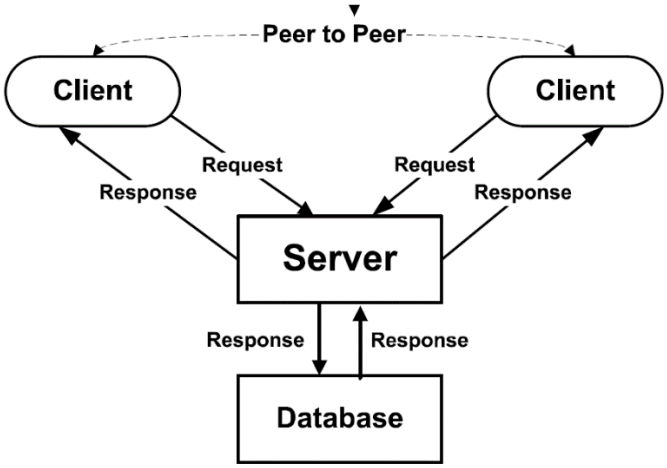
2.2.8 Kelebihan Metode Pengamanan Pada Penelitian.

Penerapan metode pengamanan komunikasi berbasis enkripsi dan tanda tangan digital memiliki kelebihan meliputi:

1. Menerapkan algoritma RSA yang menggunakan pasangan kunci yakni kunci privat dan kunci publik untuk mengamankan pertukaran kunci AES (*Key Exchange)* maka proses enkripsi dan dekripsi sebuah pesan yang berisi kunci AES hanya dapat dilakukan jika memiliki pasangan kunci yang valid, sehingga pengiriman kunci AES aman.
2. Penerapan Algoritma AES dapat menghasilkan pengamanan yang efisien jika diterapkan pada perangkat yang memiiki *resource* yang kecil dan berbeda-beda karena memiliki banyak parameter ukuran kunci yang dapat digunakan sesuai dengan kebutuhan sistem.
3. Dengan menerapkan tanda tangan digital untuk melakukan valisadi bahwa yang mengirim pesan atau lawan chat kita adalah lawan chating kita, sehingga pesan tersebut otentik dari pengirimnya dan tak dapat disangkal bahwa pesan tersebut bukan miliknya (*non-repudation*) sehingga penerima pesan percaya bahwa pesan memang otentik.

2.2.8 Komunikasi pada Sistem Keamanan IM

Sistem keamanan yang akan dibangun akan memiliki bentuk alur sistem komunikasi dengan arsitektur *peer to peer*. Sistem keamanan *mobile* IM yang dibangun terdiri dari *user* yang akan melakukan komunikasi pesan privat melalui jaringan publik (intranet /internet). Alur sistem keamanan pada penelitian ini secara umum dapat diilustrasikan pada Gambar 2.14.

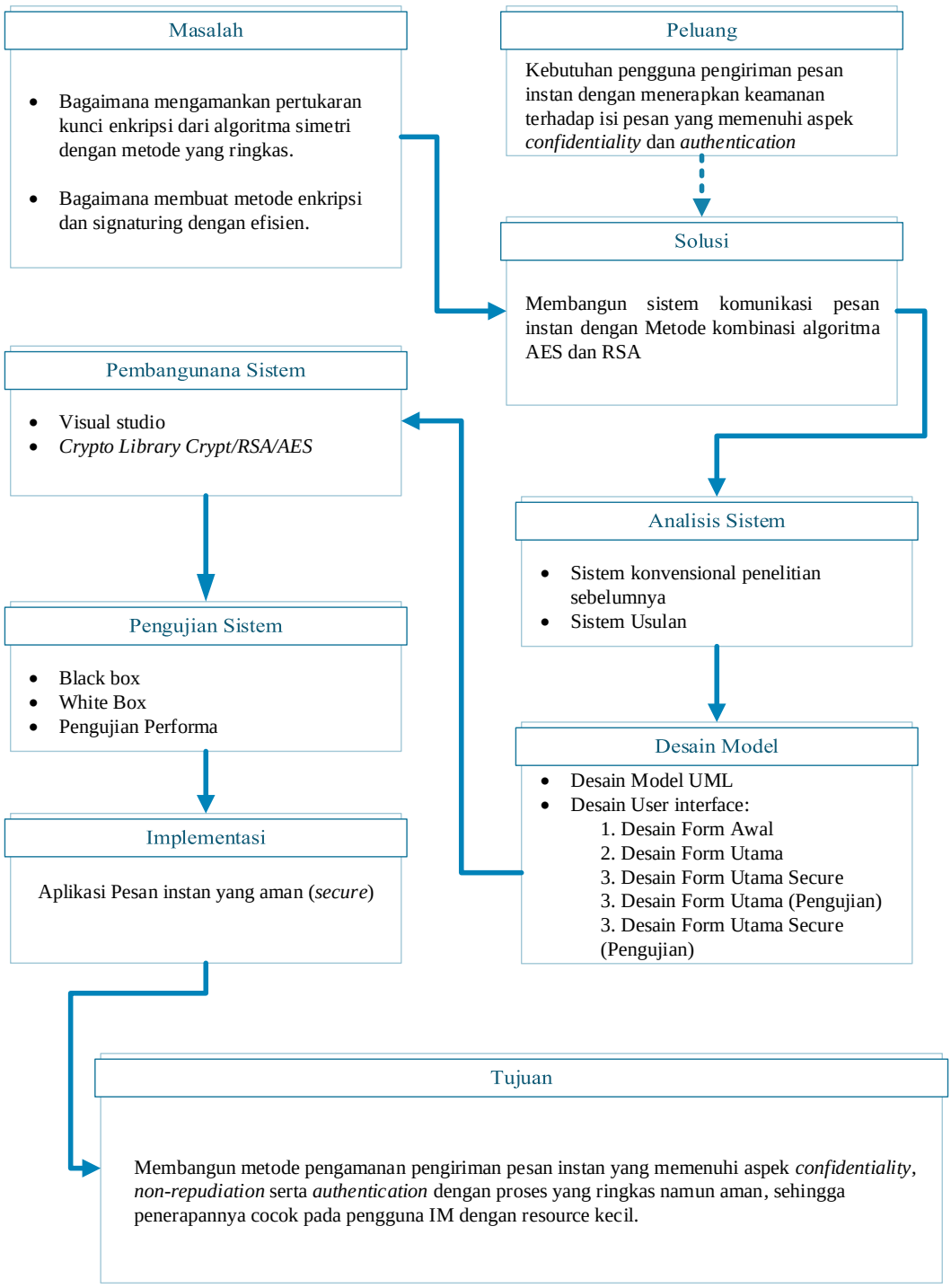


Gambar 2.14 Model *P2P* IM

Berdasarkan Gambar 2.14, dapat dideskripsikan bahwa *user* yang akan melakukan komunikasi harus membangkitkan sepasang kunci yakni: kunci privat dan kunci publik. Setelah itu, setiap *user* akan mengirimkan kunci publiknya masing-masing setiap ingin mengawali komunikasi.

Pada penelitian ini komunikasi IM akan menggunakan konsep *key pairs per session* yaitu pembangkitan pasangan kunci pada setiap sesi komunikasi, setiap pengguna yang ingin melakukan pertukaran pesan teks harus memiliki pasangan kunci yang baru pada setiap sesi komunikasi. Hal ini untuk memberikan tingkat pengamanan yang lebih baik pada proses pertukaran kunci enkripsi AES. Setelah distribusi kunci enkripsi AES selesai, maka komunikasi pesan terenkripsi dimulai.

Kerangka Pemikiran



BAB III
OBJEK DAN METODE PENELITIAN

3.1 Obyek Penelitian

Obyek yang diteliti dalam penelitian adalah tentang metode pengamanan pendistribusian key menggunakan algoritma RSA, pengamanan pesan teks pada sistem IM dengan menggunakan metode enkripsi/dekripsi dengan algortima AES dengan memasukkan tanda tangan digital pada pesan. Metode kombinasi ini selain untuk pengamanan pesan juga untuk meningkatkan kinerja proses enkripsi dengan hanya menggunakan 2 buah Algoritma kriptografi. Metode pengamanan distribusi key berbasis RSA digunakan untuk melakukan pendistribusian kunci simetris dan *signaturing*, serta AES untuk menjaga kerahasiaan pada pesan yang dikirim melalui IM.

3.2 Metode Penelitian

Metode penelitian yang di pakai adalah metode deskriptif dengan menggunakan alat dan bahan, prosedur, tahapan penelitian, pembahasan serta kesimpulan berikut:

3.2.1 Alat

Alat penelitian yang akan digunakan dalam membangun aplikasi ini dapat diuraikan seperti berikut ini:

1. Notebook Core 2 Duo P8600 @2.4 GHz, RAM 4 GB dengan Sistem operasi yang digunakan adalah Windows yang mendukung banyak aplikasi.
2. *Emulator PC* atau virtualbox untuk melakukan pengujian aplikasi

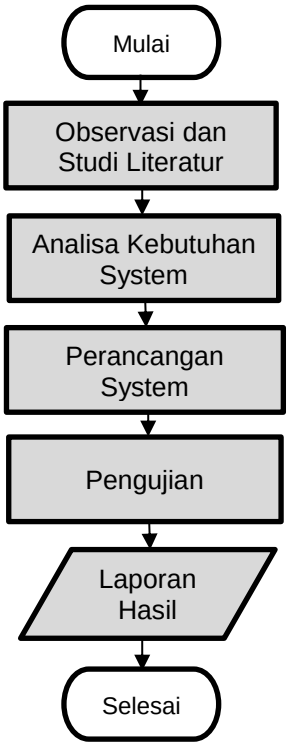
3. Visual Studio untuk membuat aplikasi *Instant Messenger*.
4. *Kali Linux* digunakan untuk testing penyerangan terhadap komunikasi aplikasi

3.2.2 Bahan

Bahan penelitian yang digunakan dalam proses penelitian ini adalah tentang metode kombinasi algorima pengamanan pendistribusian key menggunakan algoritma RSA serta pengamanan pesan teks pada sistem IM dengan menggunakan metode enkripsi/dekripsi dengan konsep algortima AES pada pesan.

3.3 Tahapan Penelitian

Bagian ini membahas tentang tahap-tahap dalam melakukan penelitian yang terdiri dari: tahap observasi, tahap analisis, perancangan sistem hingga pengujian hasil.



Gambar 3.1 Proses jalannya penelitian

Jalannya penelitian untuk membangun sistem pengamanan ini menggunakan konsep sekuensial yang saling terkait. Tahapan pada penelitian ini dapat dijelaskan sebagai berikut:

3.3.1 Observasi dan studi literatur

Tahap ini dilakukan melalui pengumpulan data melalui studi literatur dari buku, jurnal dan artikel yang berhubungan dengan topik penelitian yang dilakukan. Hasil dari studi pustaka berupa teori dan perkembangan metode pengamanan terkini mengenai IM dan informasi pendukung lainnya yang akan digunakan pada penelitian.

3.3.2 Analisis Kebutuhan Sistem

Pada tahap analisis kebutuhan dilakukan pengumpulan informasi yang meliputi proses enkripsi alogoritma RSA dan daftar parameter yang akan digunakan pada penerapan distribusi key algoritma AES dengan RSA. Hal ini dapat diperoleh dengan melakukan penyusunan *scope of work* dari aplikasi untuk simulasi yang akan dibuat beserta perangkat lunak yang dibutuhkan. Pada penelitian ini *scope of work* sistem keamanan IM meliputi:

a. Proses pembangkitan kunci privat (*private key)* dan kunci publik (*public key)* berbasis Algoritma RSA.
b. Proses pemberian tanda tangan pada pesan menggunakan konsep fungsi hash.
c. Proses enkripsi pesan menjadi *ciphertext* sebelum dikirim beserta tanda tangan digitalnya.

d. Uji coba penyerangan terhadap proses distribusi kunci dan pengiriman pesan dengan metode *Man in The Midle Attack* (MITM)
e. Proses verifikasi tanda tangan digital pada pesan yang diterima Proses ini dilakukan setelah melakukan proses dekripsi pesan.

3.3.3 Perancangan Sistem

Perancangan sistem mendeskripsikan desain sistem untuk komunikasi IM yang meliputi:

a. Desain sistem aplikasi form utama

   Pada penelitian ini akan dibangun sebuah aplikasi untuk komunikasi IM Proses awalnya berupa inisialisasi aplikasi dengan memunculkan form yang akan berisi sebuah textbox sebagai tempat ditampilaknnya percakapan dalam *chating*. Terdapat menu yang bisa berupa icon atau menu dropdown yang berisi menu setingan untuk user, setingan untuk pembangkitan kunci, menu untuk verifikasi signature, serta list user yang sedang online. Bagian bawahnya terdapat textbox untuk menuliskan pesan serta tombol untuk mengirim.

b. Desain proses pembangkitan kunci (*generating*)

   Proses ini berfungsi untuk membangkitkan pasangan kunci yaitu kunci publik dan kunci privat dari Algoritma RSA serta pembangkitan kunci algoritma AES, terdapat pemilihan panjang kunci untuk Algoritma AES 125bit, 192bit dan 256 bit. Proses ini dilakukan sebelum user melakukan pertukaran pesan (*chating*) dan kunci kedua algoritma tersebut akan di simpan menjadi sebuah file.

c. Desain proses distribusi kunci simetri

Pengamanan distribusi kunci simetri dengan memanfaatkan pertukaran kunci melalui algoritma RSA. Proses ini dilakukan ketika user akan manggil user yang sedang online pada menu list user di form utama. Ketika user mamanggil user lain yang sedang online (penerima), maka akan muncul sebuah massagebox yang berisi pemberitahuan mengenai informasi user penerima beserta kunci publiknya. Jika pengirim setuju untuk menerima user tersebut maka kunci publik dari user penerima tadi akan disimpan dalam list pertemanan. Hal yang sama juga terjadi pada sisi penerima, penerima juga akan mendapatkan notifikasi permintaan pertemanan disertai kunci public pengirim permintaan tadi. Kunci pulik ini kemudian akan mengenkripsi kunci simetris dari AES milik kita yang akan di kirim ke penerima.

d. Desain proses untuk pengiriman pesan

Tombol kirim pada form utama tadi berisi perintah untuk mengenkripsi pesan dengan Algoritma AES dan memasukkan key signature berupa private key pengirim sebelum dikirim, isi pesan tersebut bersisi plaintext serta kunci private pengirim sebagai signature yang sudah menjadi chipertext, setelah itu pesan tersebut dikirim ke penerima.

e. Desain sistem untuk proses verifikasi (*verifying*)

Pada proses ini validasi pada pesan IM yang sudah diberikan tanda tangan digitial dilakukan. Proses verifikasi pesan akan menggunakan kunci private pengirim dan mencocokan dengan pasangan kunci publik pengirim. Kunci publik pengirim tersebut adalah bagian pasangan kunci yang dibangkitkan

sebelumnya di sisi pengirim. Jika kunci publik yang digunakan bukan merupakan pasangan kunci privatnya, maka otentikasi pesannya tidak valid.

f. Perancangan Sistem Keamanan IM

Tahap ini dibuat sesuai dengan hasil observasi dan analisa yang telah dilakukan. Dokumentasi yang dihasilkan dari tahap desain sistem ini yaitu *Flow Chart* dan *Use Case Diagram*.

3.3.4 Pengujian

Tahap Pengujian dilakukan untuk mengukur kinerja algoritma RSA dan AES yang meliputi: proses pembangkitan pasangan kunci (*Generating)*, proses enkripsi, deksripsi dan distribusi kunci AES, proses pemberian tanda tangan (*Signing)*, proses enkripsi dan deksripsi pesan serta proses verifikasi tanda tangan(*Verifying)* pada pesan *mobile* IM. Tahap pengujian akan menggambarkan bagaimana tingkat kecepatan dan performa algoritma RSA dan AES jika diterapkan pada IM dalam proses pengamanan distibusi kunci, pengamanan data untuk menjaga kerahasiaan dan keaslian pesan dengan melakukan proses autentikasi pesan tersebut.

3.3.5 Evaluasi

Setelah melakukan analisa terhadap hasil pengujian, pada tahap ini dilakukan evaluasi terhadap hasil penelitian. Tahap ini digunakan untuk memberikan kesimpulan untuk menetukan parameter kunci yang paling efisien untuk penerapan pada aplikasi IM serta metode pendistribusian kunci yang ampuh untuk menangkal serangan di jaringan. Selain itu, tahap ini juga akan memberikan masukan untuk penelitian dengan topik yang sama.

BAB IV
ANALISA DAN DESAIN SISTEM

Analisis dan perancangan sistem memerlukan tahapan yang sistematis untuk mendapatkan aplikasi yang baik dan bersesuaian dengan kegunaan dan tujuannya. Tahap awal dari analisis adalah menganalisis kebutuhan-kebutuhan sistem mulai dari kebutuhan pengguna, kebutuhan non-fungsional, dan kebutuhan fungsional. Sedangkan untuk tahap perancangan aplikasi yaitu perancangan proses transmisi data, proses kriptografinya dan perancangan antarmuka.

4.1 Analisa Sistem

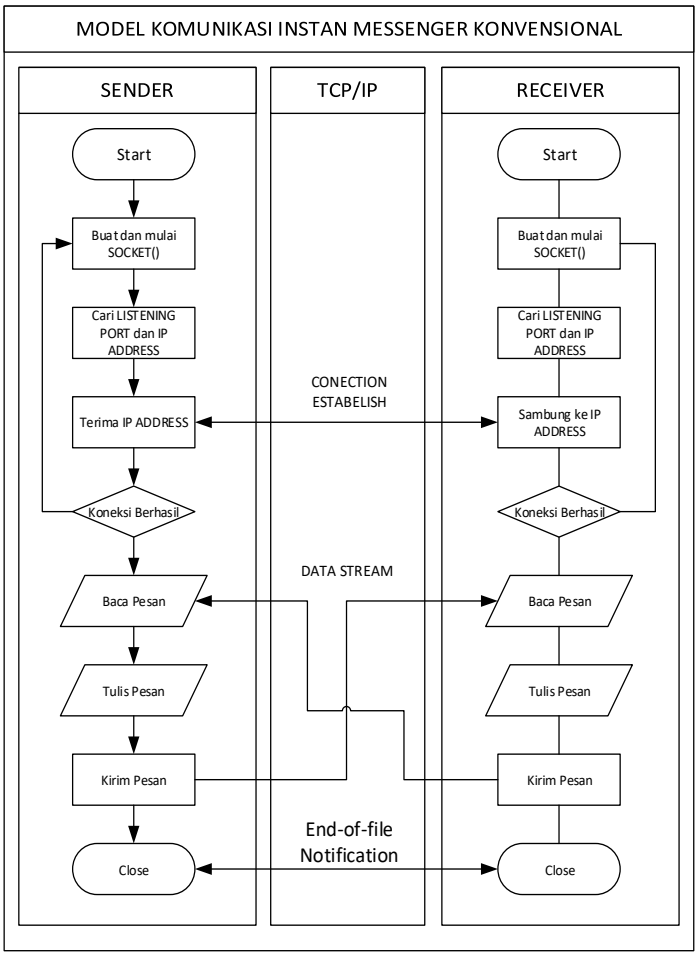
Analisis sistem (*system analysis*) dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasikan permasalah-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikan. Analisis dapat juga diartikan memahami sistem pemikiran yang kompleks dengan memecahnya ke dalam unsur-unsur yang lebih sederhana sehingga hubungan antar unsur-unsur itu menjadi jelas.

4.1.1 Analisis Sistem Berjalan

Pada penelitian sebelumnya telah dibangun sebuah aplikasi pesan instan (IM) yang di lengkapi dengan pengamanan pengiriman pesan dengan berbagai metode kombinasinya. Penelitian terakhir metode yang digunakan yaitu kombinasi tiga algoritma pengamanan pengiriman pesan dimulai dari proses pembangkitan kunci,

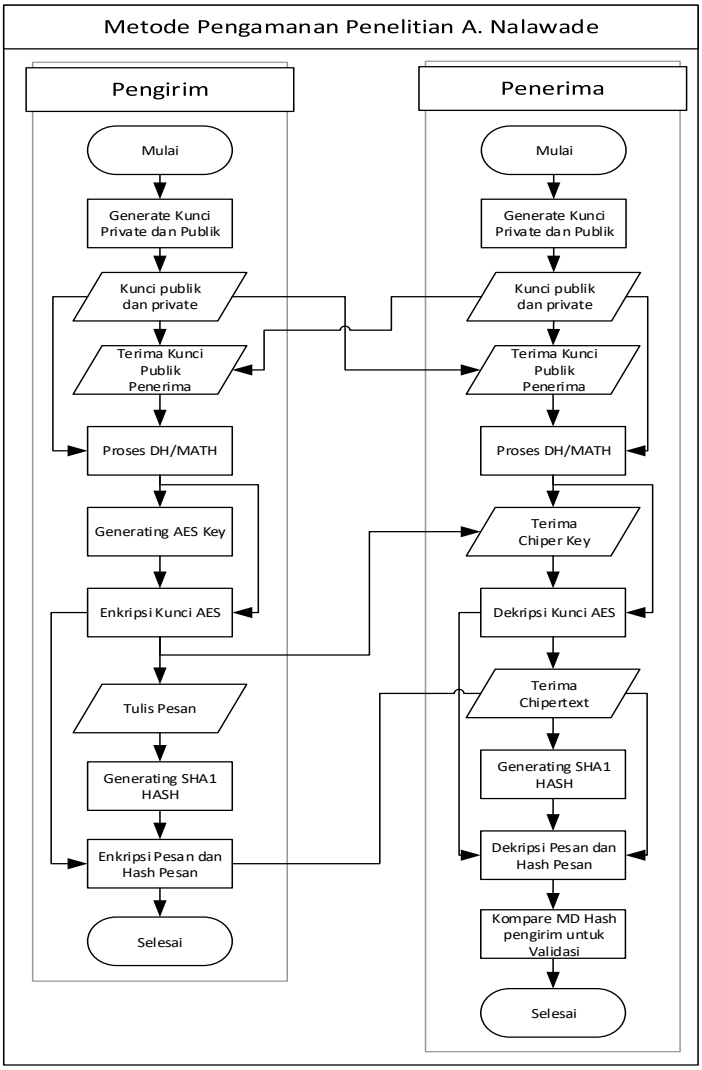
pertukran kunci simetris, melakukan *hashing* pada pesan text sebagai signature pesan, proses enkripsi pesan tadi serta pengiriman pesan ke jaringan IM. Dari proses-proses tersebut tentu memakan waktu untuk setiap prosesnya.

Berdasarkan penelitian dan pengamatan langsung pada model konvensioanl arsitektur IM tanpa penerapan pengamanan pengiriman pesan adalah sebagai berikut:



Gambar 4.1 Alur komunikasi IM Konvensional

Sedangkan untuk metode kombinasi pengamanan pengiriman pesan pada penelitian A.Nalawade digambarkan dalam bentuk flowchart pada Gambar 4.2.



Gambar 4.2 Crypto System IM A. Nalawade

Gambar 4.2 diatas memperlihatkan proses pengamanan pengiriman pesan pada penelitian A. Nalawade dengan menggunakan tiga kombinasi algoritma enkripsi yaitu Diffie Hellman untuk pengamanan pertukaran kunci simetri, proses pengamanan pengiriman pesan menggunakan algoritma AES 256 bit dan pemberian signature pada isi pesan menggunakan algoritma SHA1. Metode pengamanan pada penelitian ini dimulai dari pembangkitan kunci untuk algoritma Diffie Hellman kemudian di lakukan pertukaran kunci publik dari Diffie Hellman

(DH) tadi, setelah kedua user *chat* menerima kunci public dari masing masing algoritma DH pada setiap user dilanjutkan dengan proses DH-Math dan hasil DH-Math tadi digunakan untuk mengenkripsi key Algoritma AES dan mengirimkannya ke penerima pesan. Ketika kedua user telah memiliki kunci simetri AES baru kemudian proses pengiriman pesan yang telah di enkripsi dimulai.

Dari proses-proses tadi, penulis mencoba mempersingkat proses pertukaran kunci, serta cara signaturing pada penelitian tersebut dengan hanya menggunakan satu algoritma saja.

Pada penelitian (Roy 2016) tentang komparasi antara RSA dan Diffie Hellman diperoleh hasil pada table berikut.

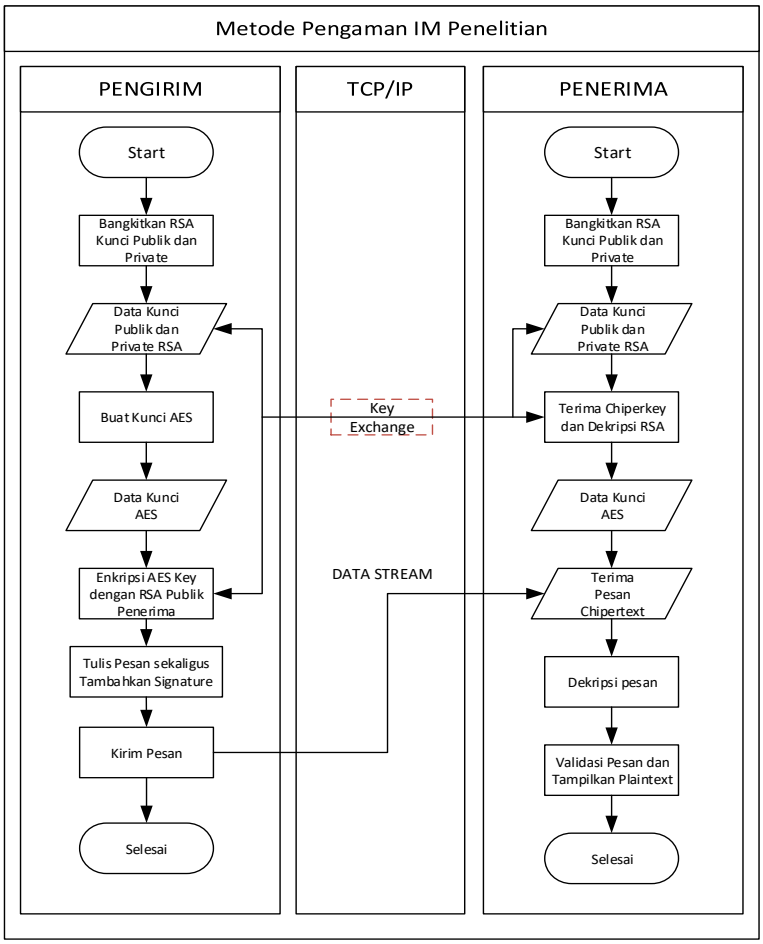
Tabel 4.1 Studi Komparasi RSA dan Diffie Hellman oleh Ayan Roy

Parameter	RSA	Diffie-Hellman
Pembangkitan Kunci	Pembangkitan Kunci RSA melalui proses yang rumit	Pembangkitan Kunci Diffie-Helman tidak rumit
Kemanan Kunci	Bergantung pada kerumitan Pemfaktoran bilangan prima	Bergantung pada kerumitan Diskrit Logaritma
Proses Enkripsi	Proses enkripsi lebih mudah dan tak rakus resource	Proses enkripsi tidak mudah dan lebih memakan resource
Enkoding Kunci Publik	Kunci Publik sangat kecil untuk di encoding	Kunci Publik sangat besar untuk di encoding
Proses Autentikasi	Autentikasi hanya di pihak pengirim	Autentikasi di kedua sisi, baik pengirim maupun penerima
Serangan	Serangan pada metode modulus dan serangan berulang (Saat ini belum ditemukan algoritma yang mangkus untuk mencari modulus RSA)	Kemungkinan besar bisa diserang dengan metode Man in The Middle Attact
Kapabilitas	Enkripsi, Key Exchange, Data Signaturing	Key Exchange

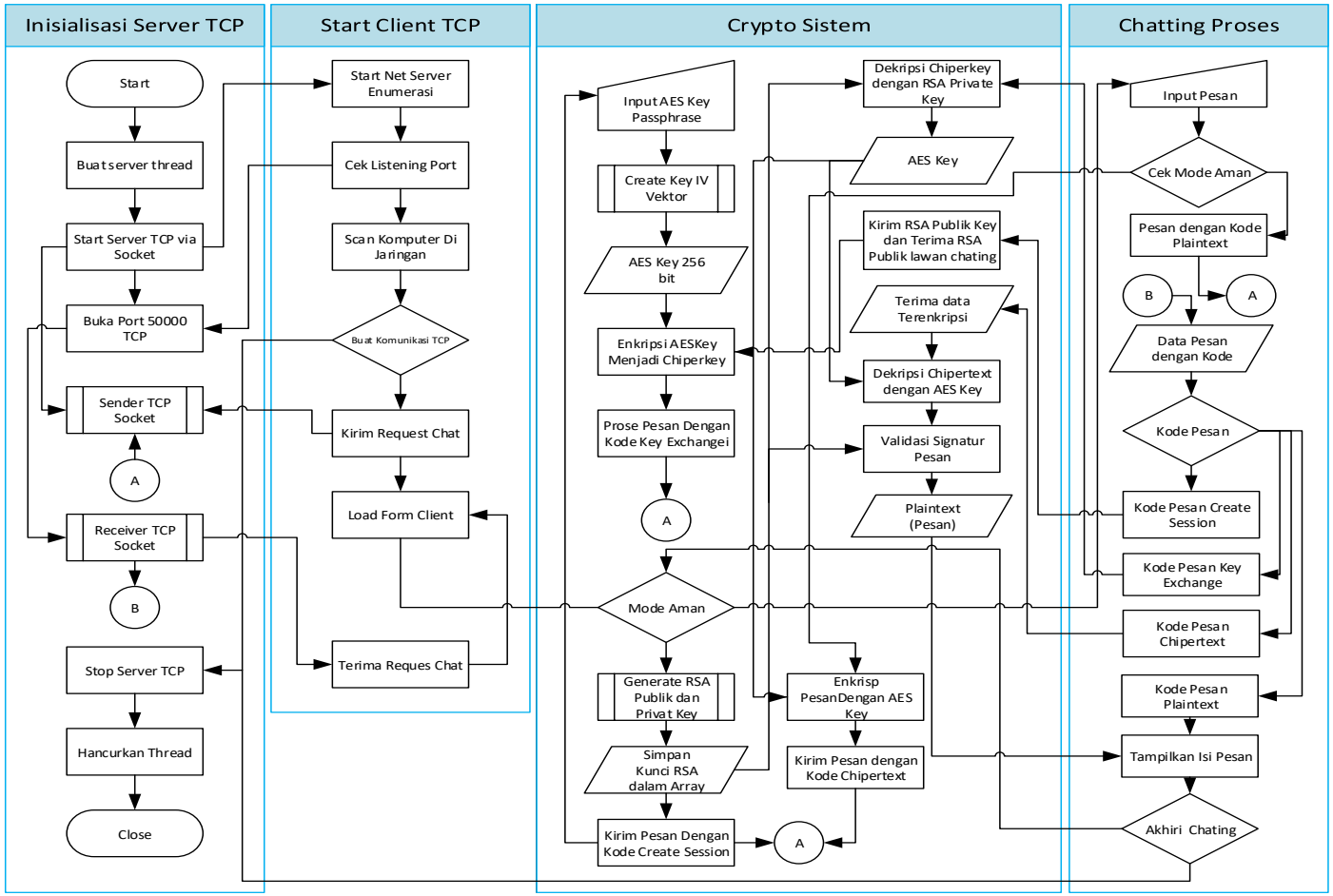
Berdasarkan table 4.1 diatas penulis akan meringkas proses *key exchange* dan *signaturing* yang menggunakan algoritma Diffie Hellman dan SHA1 dengan hanya menggunakan Algoritma RSA saja.

4.1.2 Analisis Sistem Usulan

Sistem yang akan dibangun pada penelitian ini adalah pengamanan pengiriman pesan pada LAN Messenger menggunakan Kriptografi. Metode pengamanan pesan menggunakan kombinasi dua algoritma yakni algoritma kunci public (RSA) dan simetri (AES), lebih jelasnya diapat dilihat pada gambar 4.3 berikut ini.



Gambar 4.3 Metode pengamana RSA dan AES usulan



Gambar 4.4 Diagram alir sistem usula

4.1.2.1 Analisis Metode Pengamanan

Pada bagian ini akan di jelaskan proses pengamanan mulai dari generating key RSA, proses enkripsi Key AES, pertukaran kunci, enkripsi pesan dan pemberian tanda tangan digital pada pesan.

1. **Proses generating key RSA**

   Proses ini merupakan tahap awal pada metode pengamanan komunikasi IM dimana pihak pengirim (*sender*) dan penerima (*receiver*) terlebih dahulu membangkitkan pasangan kunci (*key pair* RSA) yang akan digunakan untuk mengenkripsi kunci algoritma simetris AES. Konsep algoritma pembangkitan kunci dideskripsikan pada Gambar 4.5 berikut:

```
RSA Key Generator
Mulai

Pilih nilai prima p (random)
Pilih nilai prima q (random)
p≠q
Hitung m = (p-1) * (q-1)
Hitung n = p*q
Pilih e yang relative prima terhadap m
Cari d, sehingga e*d = 1 mod (m) atau d=e^-1
Kunci public = e, n
Kunci private = d, n

selesai
```

Gambar 4.5 Algoritma pembangkitan kunci RSA

Contoh perhitungan manual enkripsi RSA:

Misalnya kita mau mengenkripsi kata "**SECRET**" dengan RSA, lalu kita dekripsi kembali ke dalam *plain text*. Karena *p* dan *q* berjumlah minimal 100 digit atau lebih, nilai *d* dan *e* bisa berjumlah sama dengan 100 digit dan nilai akan berjumlah 200 digit. Untuk itu di contoh pemakaian berikut, kita akan memakai angka-angka yang kecil agar mudah dalam penghitungan.

Cara pengerjaannya adalah:

1. Kita pilih $p = 3$ dan $q = 5$
2. Hitung $N = pq = 3*5 = 15$
3. Nilai *e* harus merupakan bilangan prima yang lebih besar dan relatif dekat dengan $(p-1)\ (q-1) = (2)(4) = 8$, sehingga kita pilih $e = 11$. Angka 11 adalah bilangan prima terdekat dan lebih besar daripada 8
4. Nilai *d* harus dipilih sehingga

$$\frac{(ed - 1)}{(p - 1)(q - 1)}$$

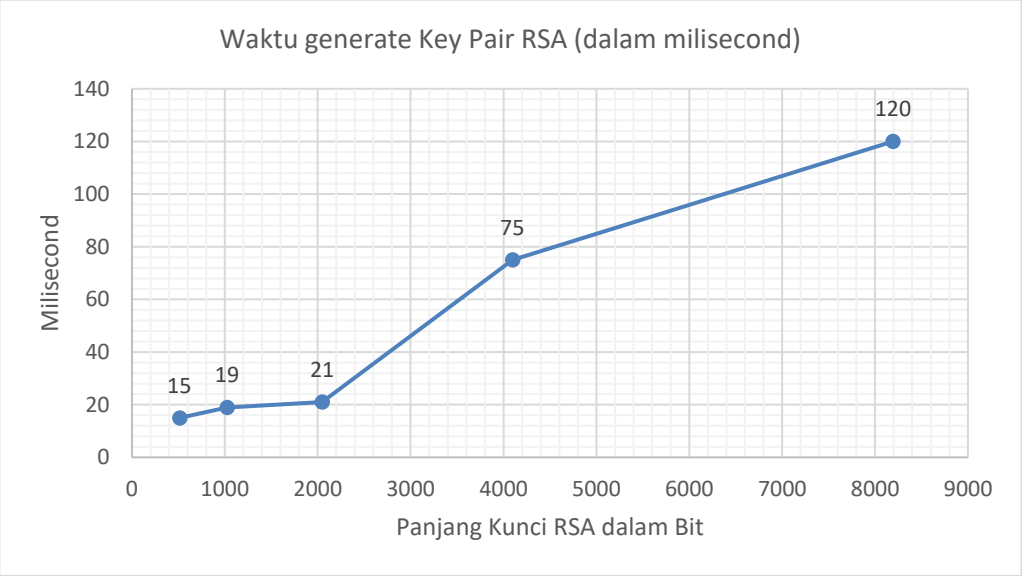
merupakan sebuah integer. Lalu nilai $(11d - 1) / [(2)(4)] = (11d - 1) / 8$ juga merupakan integer. Setelah melalui proses penghitungan, salah satu nilai yang mungkin adalah $d = 3$.

5. Lalu kita masukkan kata yang akan dienkripsi "SECRET". Kita akan mengkonversi string ini ke representasi desimal menggunakan nilai karakter ASCII, yang akan menghasilkan nilai ASCII 83 69 67 82 69 84
6. Pengirim akan mengenkripsi setiap digit angka pada saat yang bersamaan menggunakan nilai kunci publik $(e, n) = (11, 15)$. Lalu setiap karakter *ciphertext* akan masuk ke persamaan $C_i = M_i^{11} \ mod 15$. Yang akan menghasilkan nilai digit masukan adalah 836967826984 yang akan dikirim sebagai 0x2c696d286924

   Penerima akan mendekripsi setiap digit angka menggunakan nilai kunci privat **$(d, n) = (3, 15)$.** Lalu, setiap karakter *plaintext* akan masuk persamaan $M_i = C_i^3 \ mod 15$. String masukan yang bernilai 0x2c696d286924, akan dikonversi

kembail menjadi 836967826984, dan akhirnya angka-angka tersebut akan diubah kembali menjadi bentuk string *plaintext* yang bernilai "**SECRET**".

Hasil analisis performa pembangkitan kunci RSA terhadap panjang kunci yang telah diteliti sebelumnya oleh (Spinoza, 2006) ditunjukkan pada Gambar 4.6 berikut:



Gambar 4.6 Performa *generating key* RSA terhadap panjang kunci

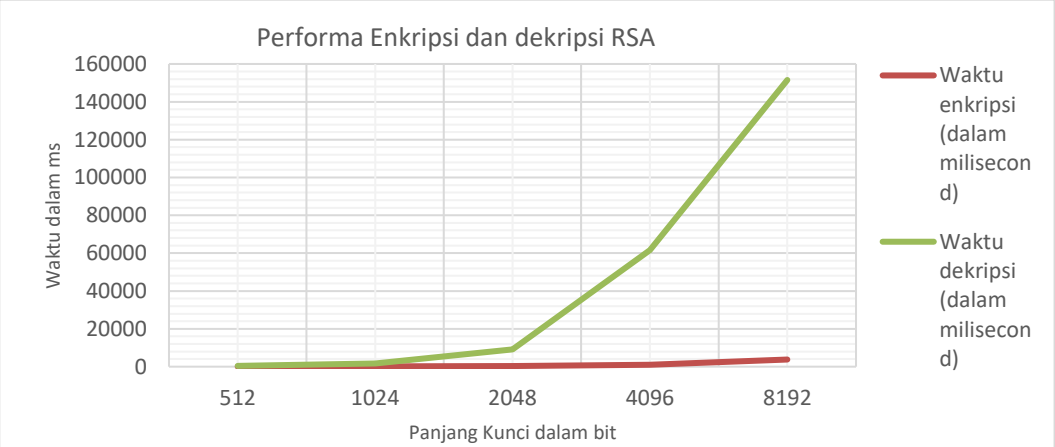
2. Proses enkripsi AES Key dengan RSA dan key exchange

Proses ini merupakan tahap pertukaran kunci enkripsi simetri (AES). Metode pengamanan pertukaran kunci simetri yakni melakukan enkripsi kunci AES kemudian mengirimnya ke penerima kunci. Konsep algoritma enkripsi, dekripsi dan pertukaran kunci RSA dideskripsikan sebagai berikut:

```
RSA Key Exchange
Mulai

Pengirim()
        Generate pairkey RSA
        Kirim kunci public RSA ke lawan chat
        Terima Kunci Publik RSA lawan
        Baca kunci public RSA lawan (n,e)
        Enkripsi C = M^e mod n
        Chiperkey = Kunci AES^e mod n
        Kirim chiperkey AES
Penerima()
        Generate pairkey RSA
        Kirim kunci public RSA ke lawan chat
        Terima Chiperkey AES
        Dekripsi M = C^d mod n
        AESKey = Chiperkey^d mod n
        Simpan AESkey

Selesai
```

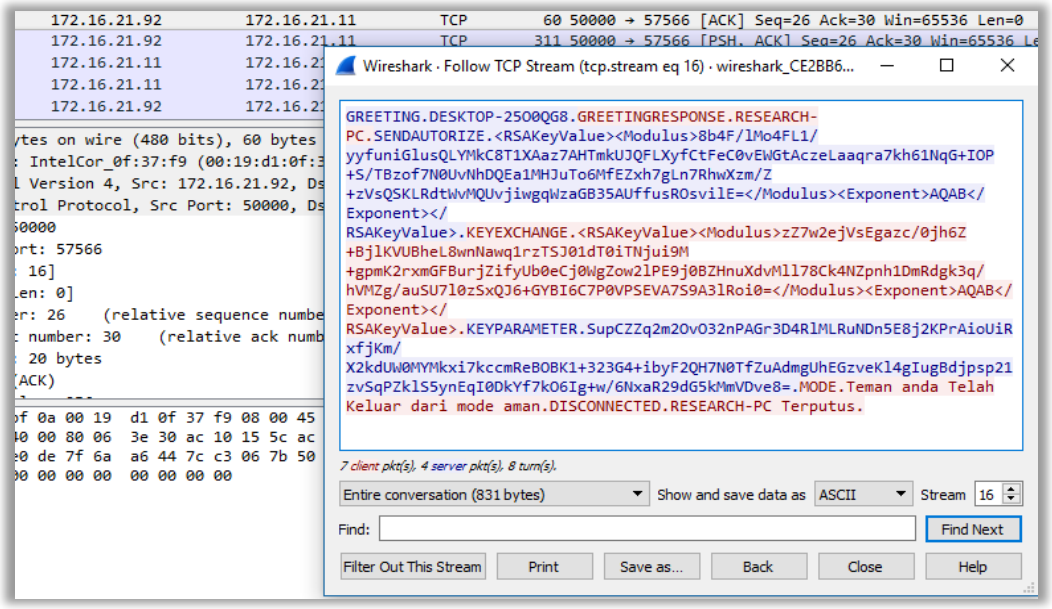
Gambar 4.7 Algoritma Pertukaran Kunci AES dengan RSA

Analisa performa enkripsi dan dekripsi yang telah diteliti sebelumnya oleh (Alimohammadi and Pouyan 2014) pada Gambar 4.7 berikut.



Gambar 4.8 Perbandingan enkripsi dan dekripsi RSA

Dari gambar 4.8, peneliti akan menggunakan algoritma RSA dengan panjang kunci 1024 karena lama waktu antara enkripsi dan dekripsi pesan tidak terpaut jauh. Analisa pertukaran kunci ketika di jaringan menggunakan alat pengendus paket TCP Wireshark dapat di lihat pada gambar 4.9 berikut:



Gambar 4.9 Pertukaran kunci simetri AES dengan RSA.

Pada gambar 4.9 dapat dilihat terjadinya pertukaran kunci publik RSA kemudian di disusul oleh kunci simetri AES yang telah di enkripsi oleh kunci publik RSA.

3. Proses pemberian signature pada pesan text

Proses ini merupakan tahap pemberian tanda tangan digital pada pesan text (*plaintext*) sebelum di enkripsi aleh algoritma AES. Hasil pemberian tanda tangan digital ini akan menjadi pesan khusus (*message signature*). Konsep algoritma pemberian tanda tangan digital menggunakan RSA dideskripsikan pada Gambar 4.10 berikut:

```
RSA Signaturing
Mulai

Generate pasangan kunci RSA
Ambil nilai hash dari data pesan (plaintext)
Enkripsi nilai hash dengan kunci private RSA

Tanda tangan Digital = hash^e mod n
Message signature = Tanda tangan Digital + plaintext

Selesai
```

Gambar 4.10 Algoritma pemberian tanda tangan digital dengan RSA

4. Proses enkripsi pesan dengan algoritma AES

Proses ini merupakan tahap untuk mengubah pesan (*plaintext*) menjadi pesan bersandi (*chipertext*) untuk menjaga kerahasiaan pesan yang d ke penerima. Proses enkrispi pesan pada sisi pengirim akan menggunakan kunci simetri algoritma AES dapat dideskripsikan pada Gambar 4.11 berikut ini.

```
AES (Rijndael Encryption)
Mulai

Masukkan Key (palintext)
Ubah dan ekspansi key menjadi 256bit sebanyak 14
Masukkan key pada byte array vector 4x4
Masukkan plaintext ke byte array vector 4x4
Lakukan proses addround key
Mulai looping sebanyak 13 kali
Lakukan proses Subbytes
Lakukan shifrows
Lakukan Mixcolumn
Lakukan addroundkey
Ulangi hingga 13 kali
Lakukan subbytes
Lakukan shiftrows
Addroundkey

Selesai
```

Gambar 4.11 Algoritma Proses Enkripsi AES (Rijndael 256bit)

5. Proses dekripsi pesan dengan algoritma AES

Pada tahapan ini akan dilakukan deksipsi *chipertext* menjadi *plaintext* sehingga penerima bisa membaca isi pesan yang sebenarnya. Proses ini akan menggunakan kunci AES yang sebelumnya telah dikirim dan dimiliki oleh masing-masing *user*. Algoritma dekripsi dengan konsep algoritma AES bisa dideskripsikan pada Gambar 4.12.

```
AES (Rijndael decryption)
Mulai

Masukkan Key (palintext)
Ubah dan ekspansi key menjadi 256bit sebanyak 14
Masukkan key pada byte array vector 4x4
Masukkan chipertext ke byte array vector 4x4

Lakukan proses addround key

Mulai looping sebanyak 13 kali
Lakukan invshiftrows
Lakukan invsubbytes
Lakukan invMixcolumn
Lakukan addroundkey
Ulangi hingga 13 kali

Lakukan invshiftrows
Lakukan invsubbytes
Addroundkey

Selesai
```

Gambar 4.12 Algoritma dekripsi pesan

6. Proses verifikasi tanda tangan digital pesan

Algoritma ini akan digunakan untuk melakukan validasi terhadap pesan yang diterima oleh penerima. Proses verifikasi tanda tangan menggunakan *public key* pengirim yang harus diketahui sebelumnya oleh penerima pesan yang menggunakan fungsi hash. Algoritma verifikasi tanda tangan digital bisa dideskripsikan pada Gambar 4.13

```
RSA Signaturing Verify
Mulai

Ambil pesan yang telah di dekripsi AES
Pisahkan pesan dan tanda tangan digital
Dekrip tandatangan digital dengan kunci publik pengirim
Periska hash pesan teks
Compare hash dari pesan text dan hash hasil decrypt RSA

Jika sama = valid
Jika tidak = pesan tak valid

Selesai
```

Gambar 4.13 Algoritma verifikas tanda tangan digital

4.2 Desain Sistem

Tahap ini dilakukan untuk membangun rancangan sistem keamanan IM berdasarkan hasil analisa. Himpunan struktur dan tehnik untuk pemodelan desain program berorientasi objek (*Unified Modeling Language*) akan di jelaskan berikut beserta Desain Interfacenya

4.2.1 Desain Model dengan UML

Unified Modeling Language merupakan salah satu alat bantu yang dapat digunakan dalam mendesain program dalam bahasa pemograman yang berorientasi objek.

4.2.1.1 Functional Requirment

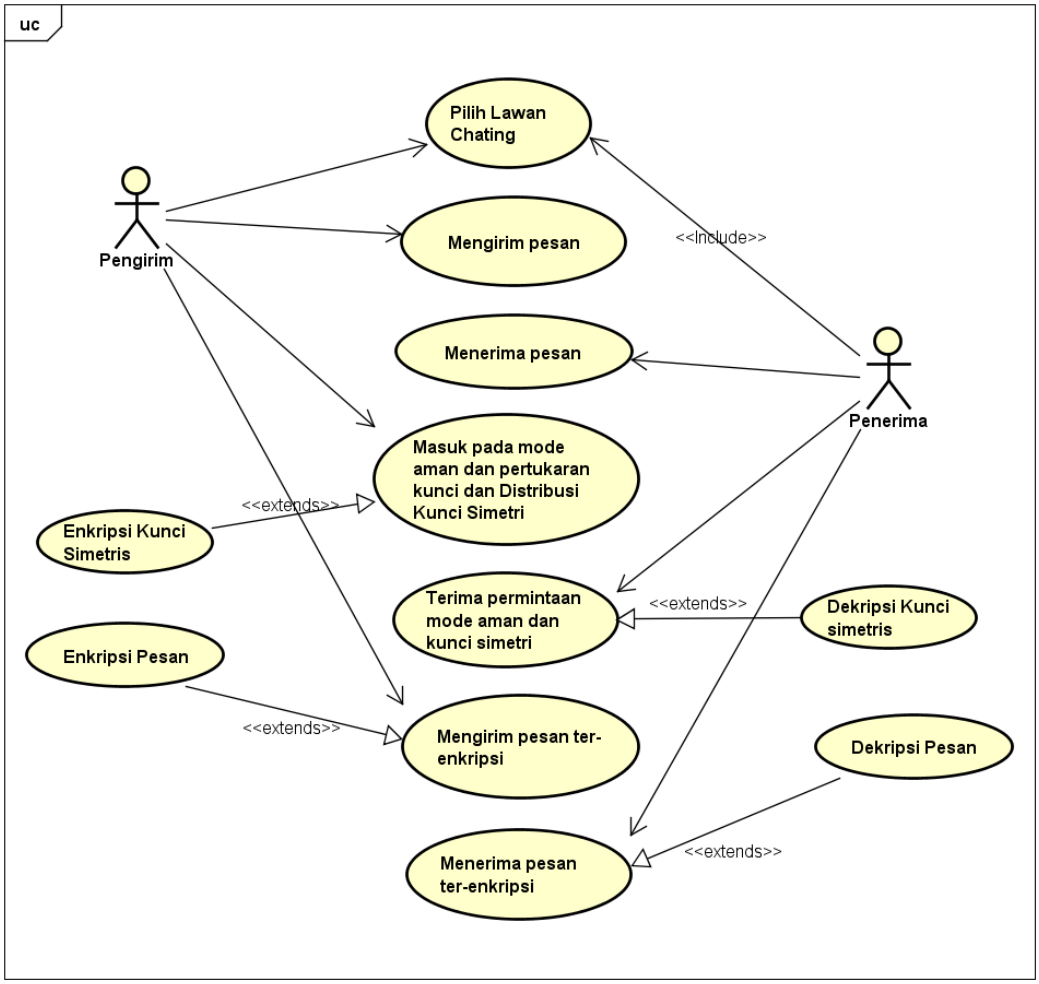
Pada tahap ini akan di tuliskan sistem yang akan dibuat dengan semua fungsi yang di perlukan.

Functional Requirment

Membuat aplikasi Visual Basic (Desktop) pesan instan dalam jaringan Local Area Network dengan pengamanan pada pertukaran kunci dan pengiriman pesan. Aplikasi dapat melakukan hal berikut:

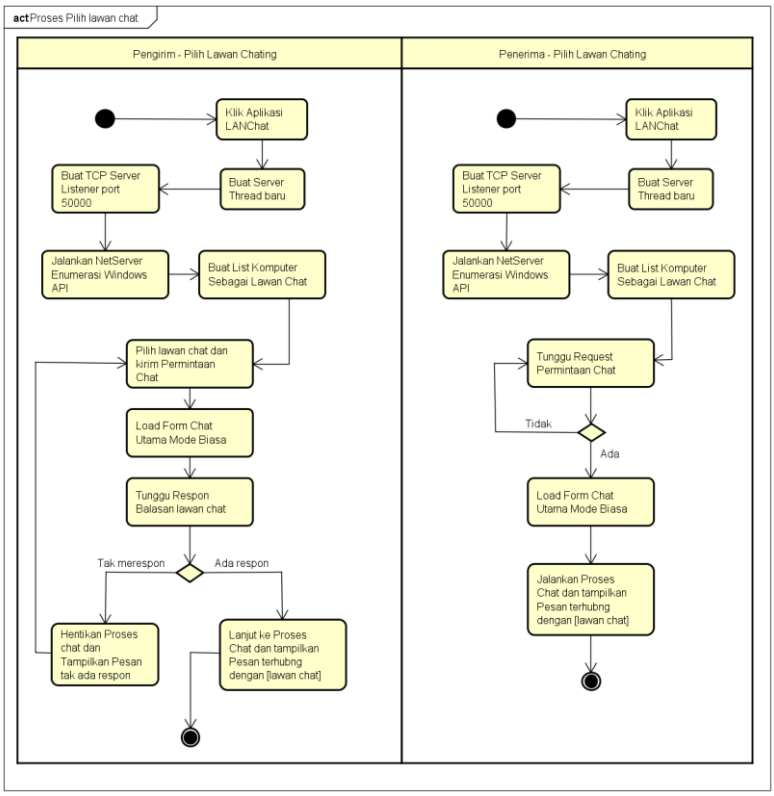
1. Melakukan scanning pada komputer di jaringan LAN sebagai user chating
2. Melakukan pertukaran pesan antara pengirim dan penerima pesan mode biasa
3. Membangkitkan Pasangan Kunci RSA
4. Membangkitkan Kunci AES dari Inputan AES Passphrase
5. Melakukan Pertukaran kunci publik RSA antara pengirim dan penerima
6. Melakukan Enkripsi pada Kunci AES dengan Kunci Publik RSA
7. Melakukan distribusi kunci simetris algoritma AES sebelum masuk ke pertukaran pesan mode aman
8. Melakukan dekripsi pada Chiperkey AES yang diterima
9. Melakukan pertukaran pesan dalam mode aman (terenkripsi)

4.2.1.2 Usecase Diagram

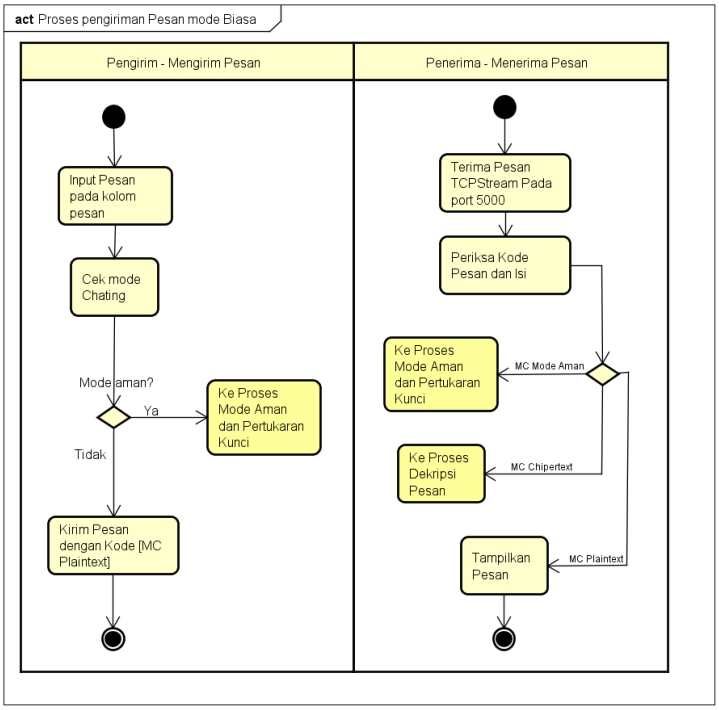


Gambar 4.14 Usecase Diagram

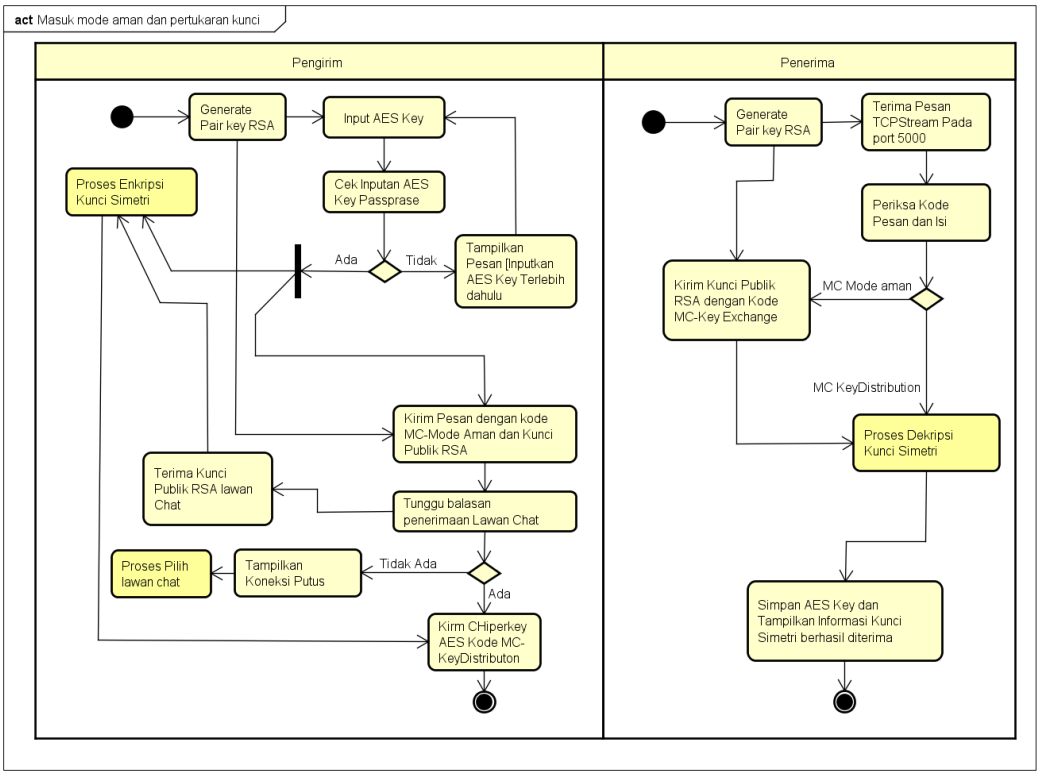
4.2.1.3 Activity Diagram



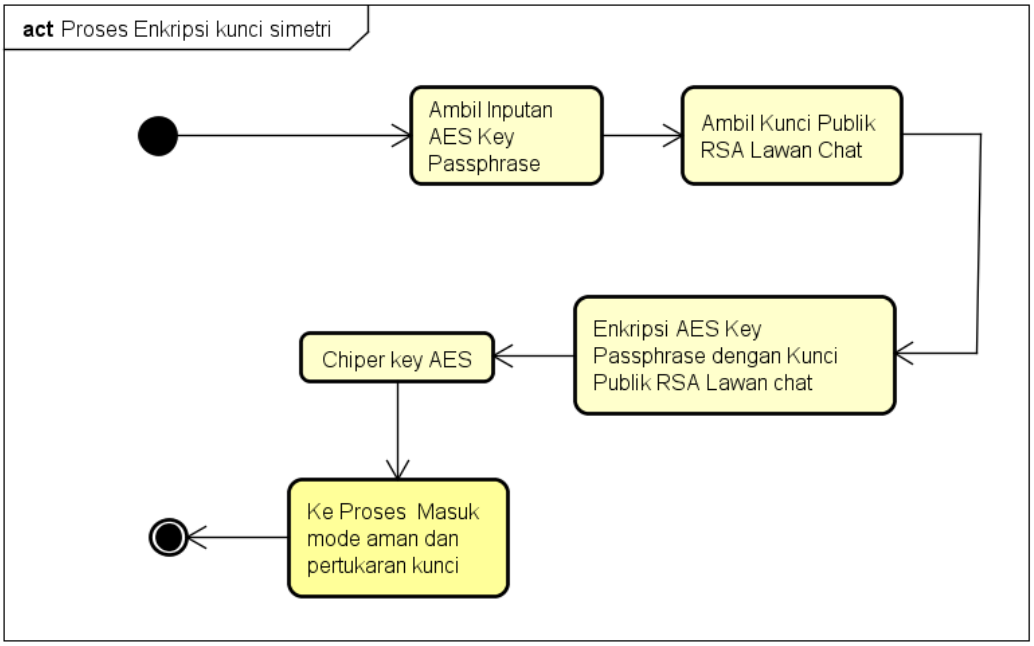
Gambar 4.15 Activity Diagram Proses pilih lawan *chating*



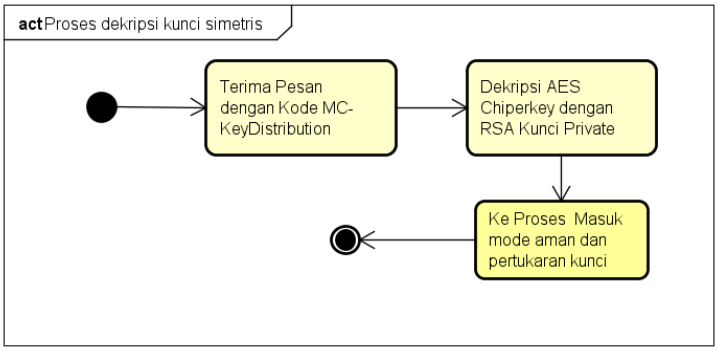
Gambar 4.16 Activity Diagram Proses pengiriman dan penerimaan mode biasa



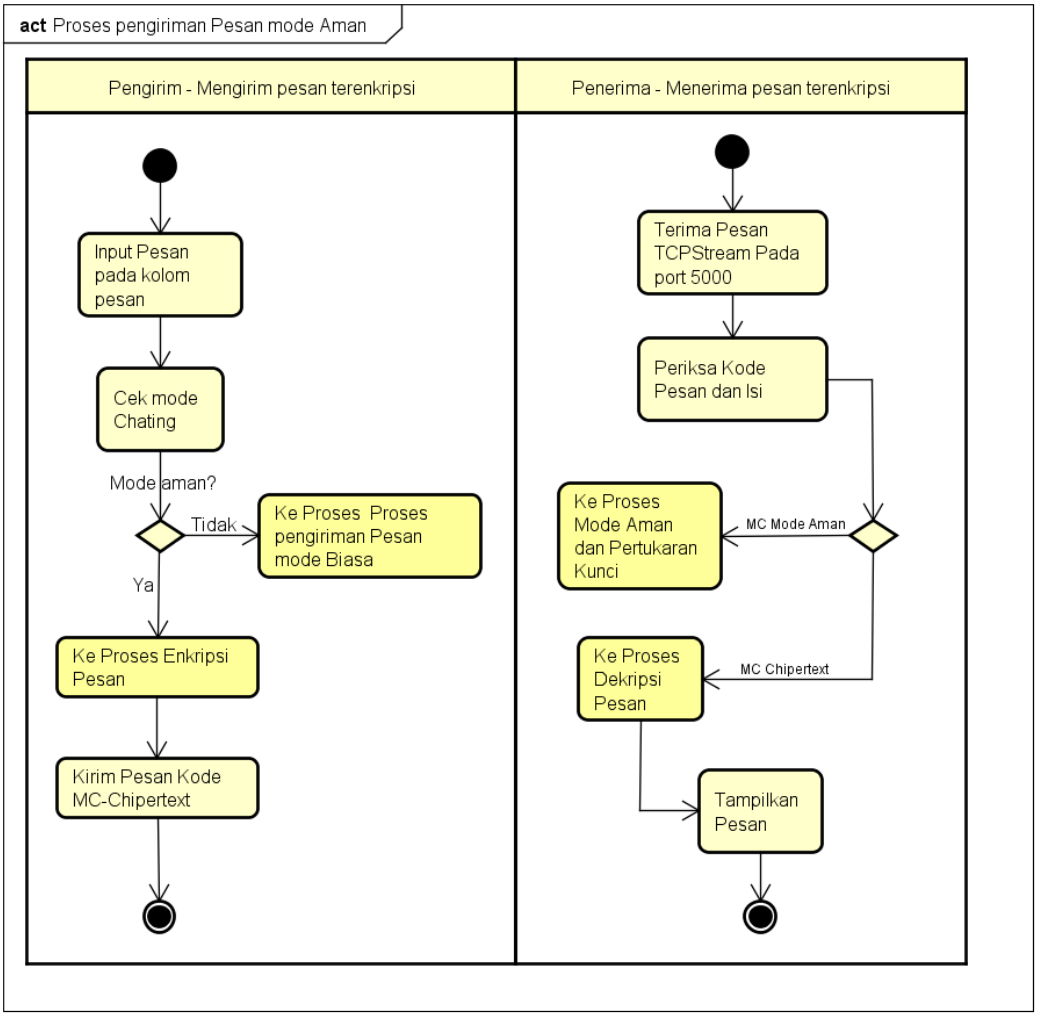
Gambar 4.17 Activity Diagram Proses masuk mode aman dan pertukaran kunci



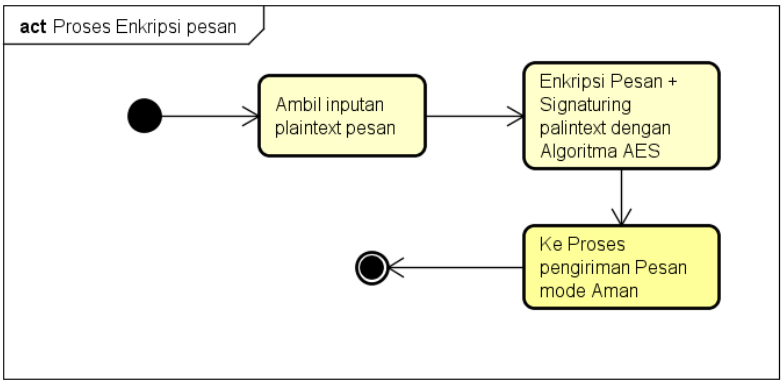
Gambar 4.18 Activity Diagram Proses enkripsi kunci simetri



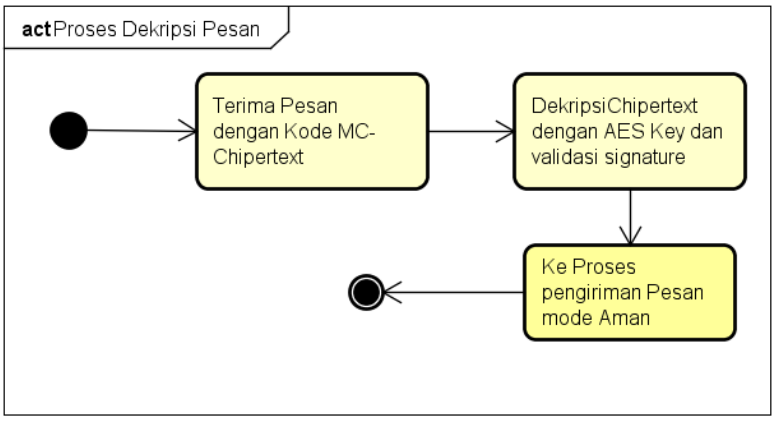
Gambar 4.19 Activity Diagram Proses dekripsi kunci simetri



Gambar 4.20 Activity Diagram Proses pengiriman pesan mode aman

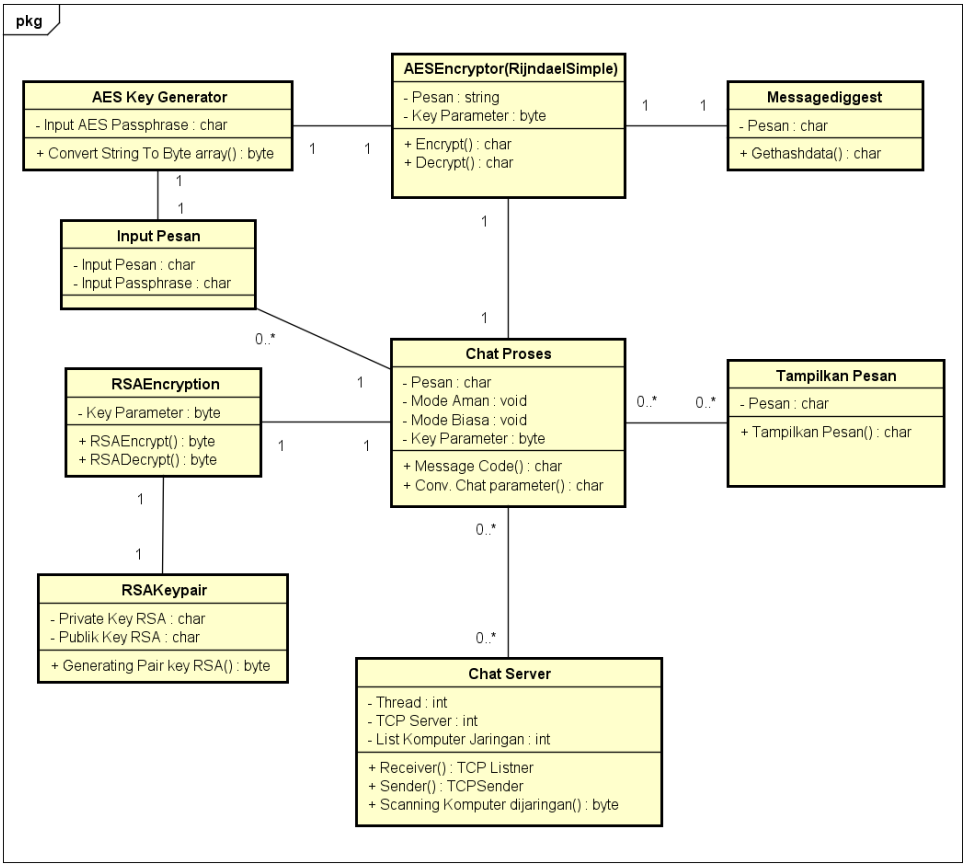


Gambar 4.21 Activity Diagram Proses enkripsi pesan



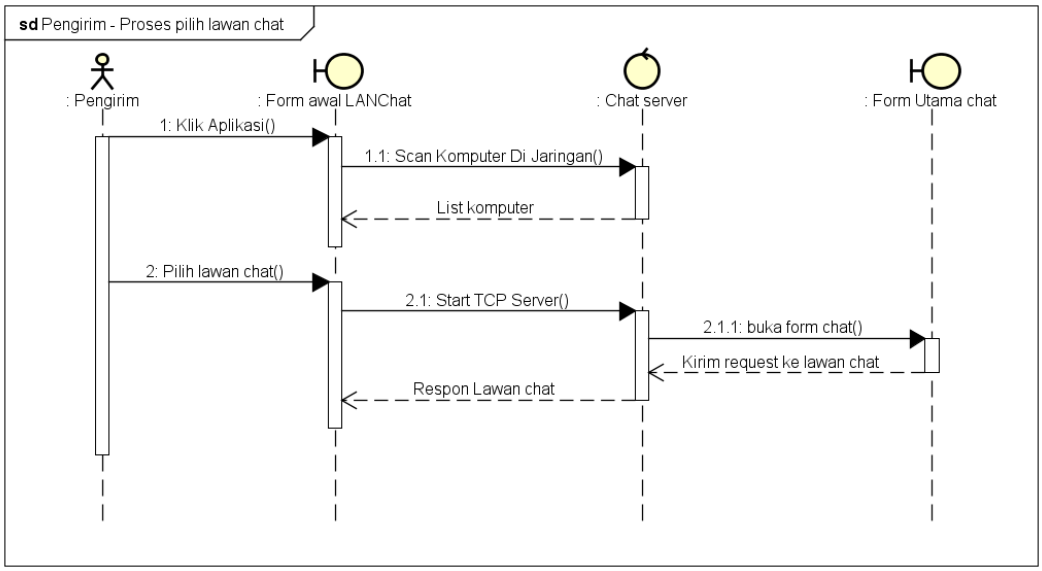
Gambar 4.22 Activity Diagram Proses dekripsi pesan

4.2.1.4 Class diagram

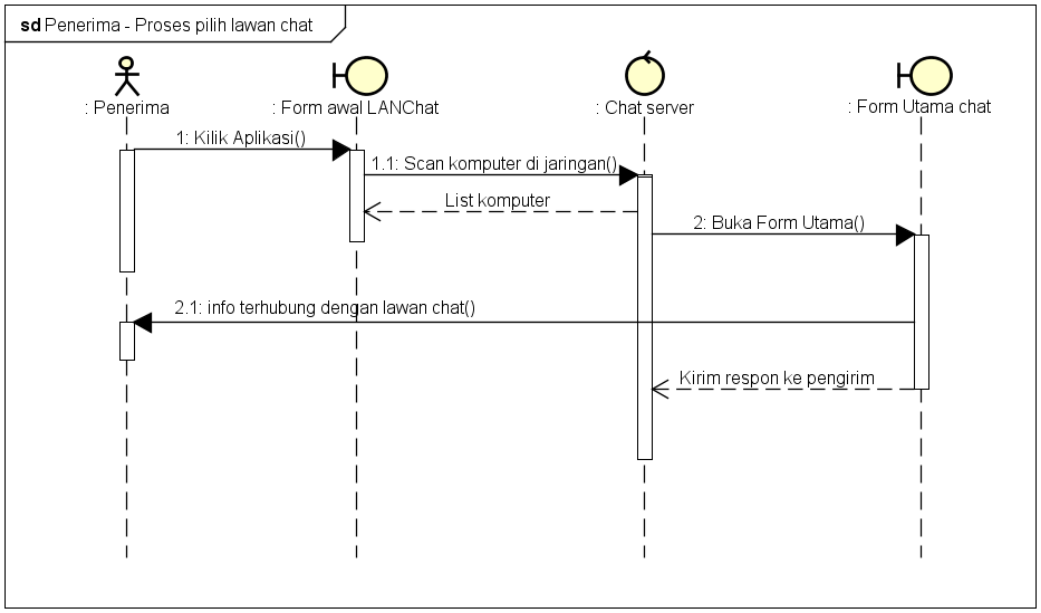


Gambar 4.23 Class Diagram sistem LANChat

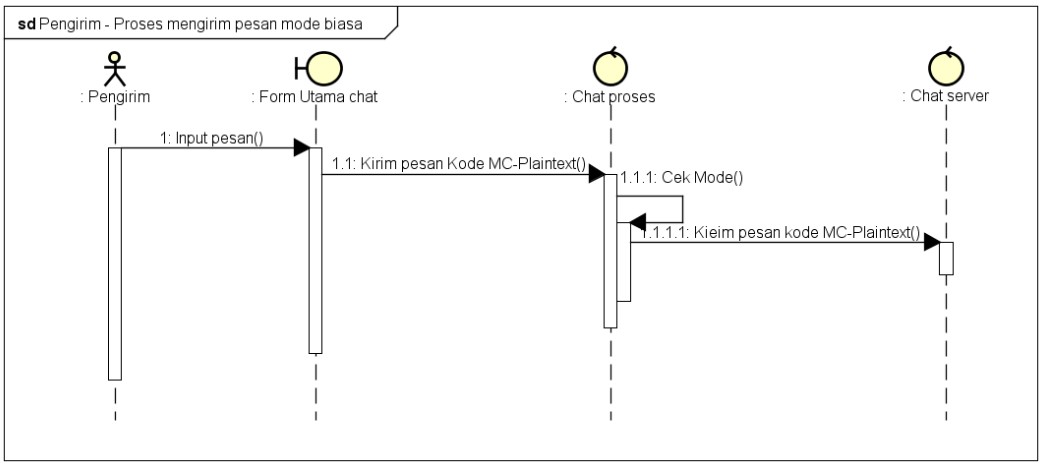
4.2.1.5 Sequence Diagram



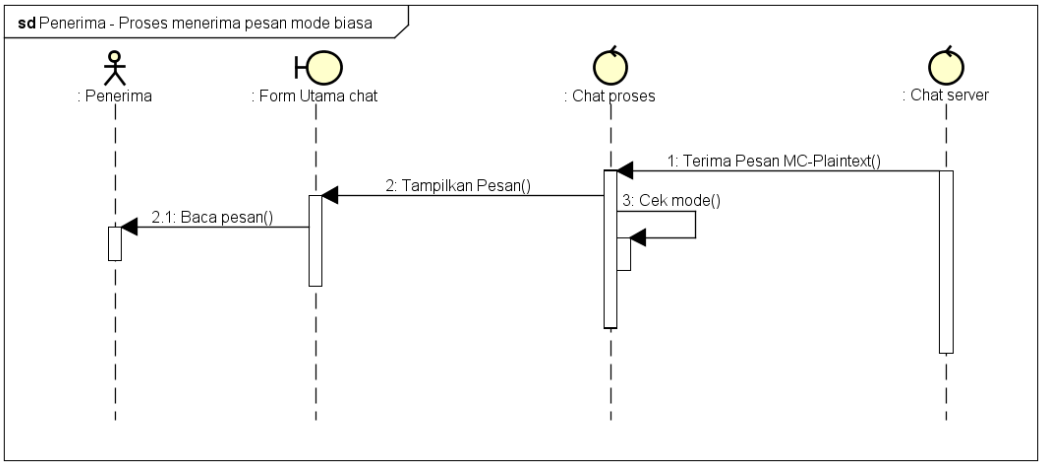
Gambar 4.24 Proses pilih lawan chat – Pengirim



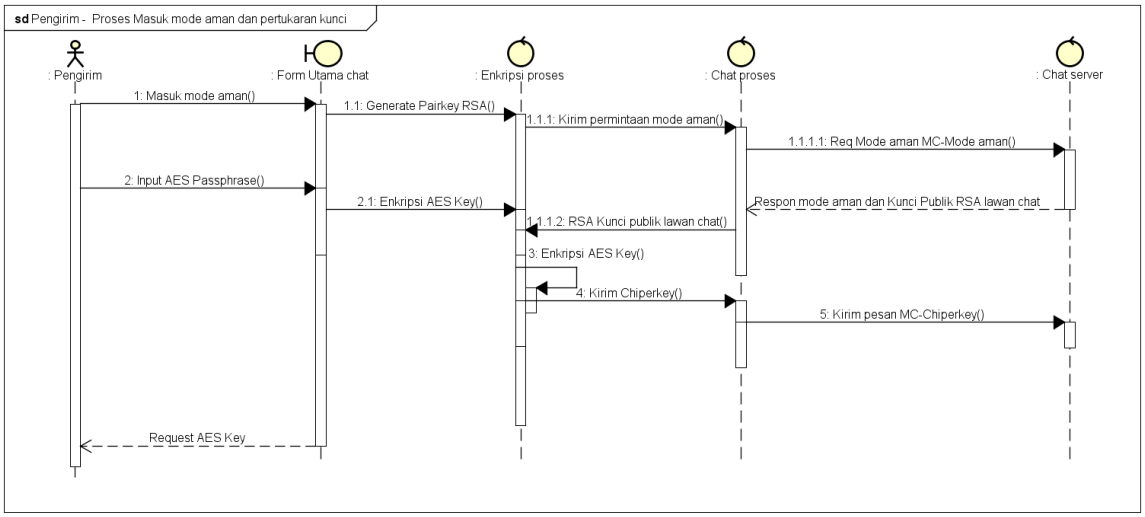
Gambar 4.25 Proses pilih lawan chat – Penerima



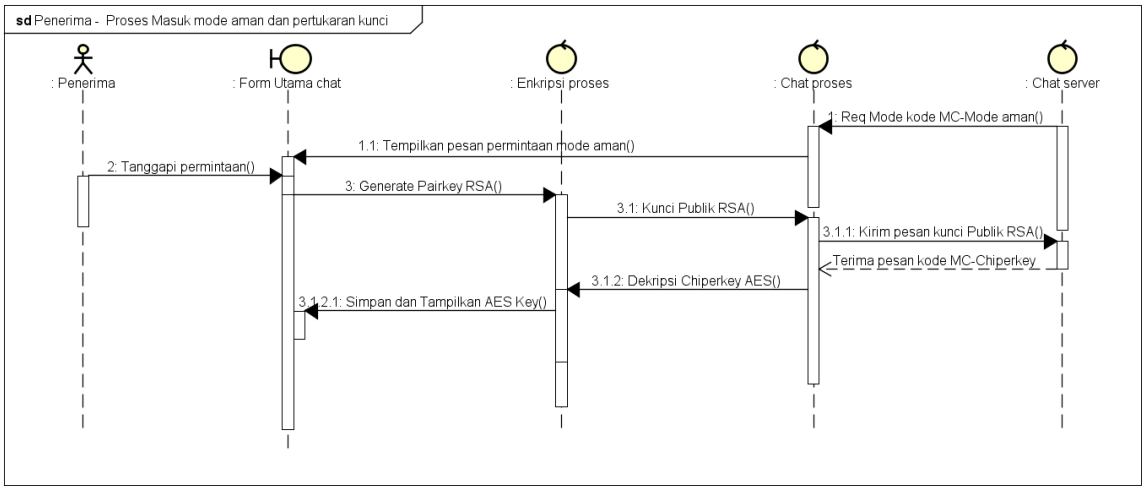
Gambar 4.26 Proses mengirim pesan mode biasa



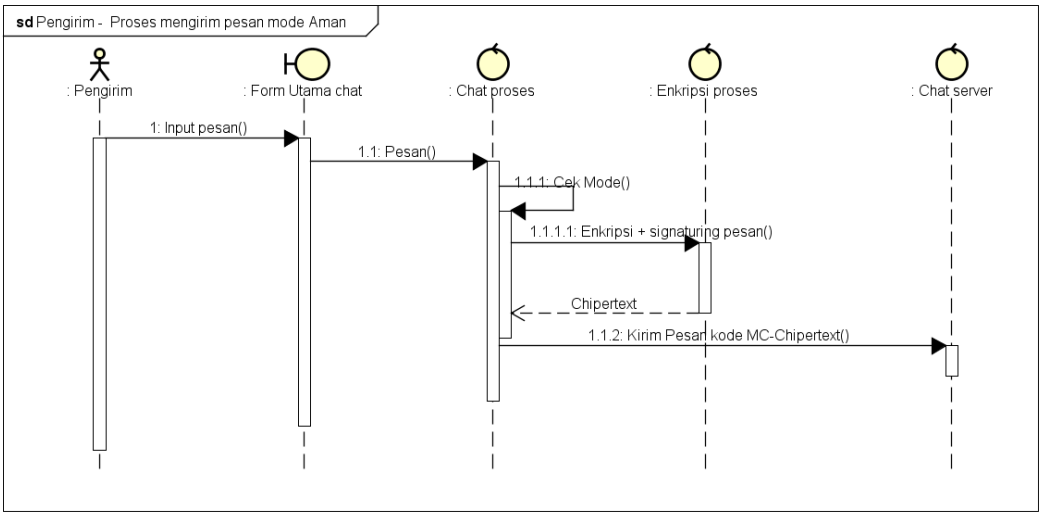
Gambar 4.27 Proses menerima pesan mode biasa



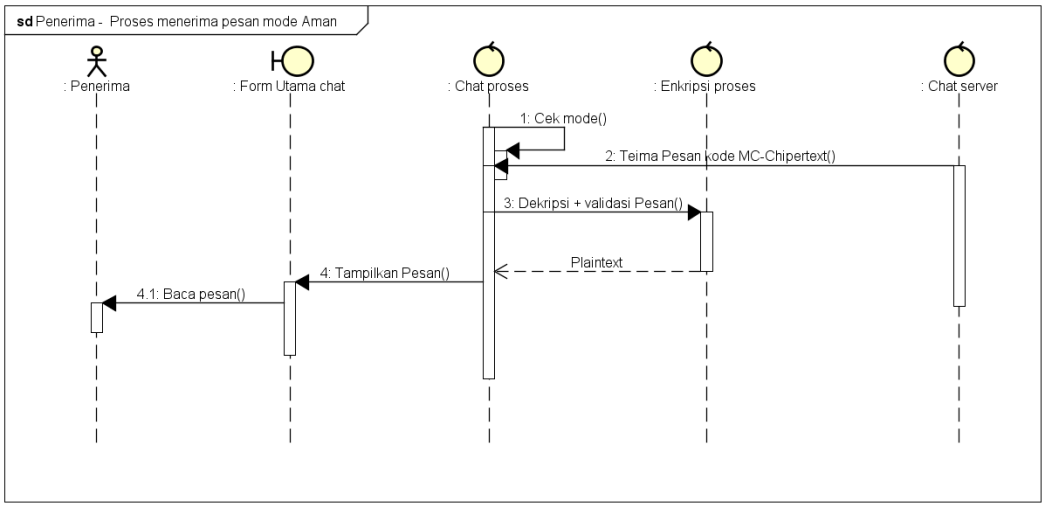
Gambar 4.28 Proses masuk mode aman dan pertukaran kunci - Pengirim



Gambar 4.29 Proses masuk mode aman dan pertukaran kunci - Penerima



Gambar 4.30 Proses mengirim pesan mode aman – Pengirim

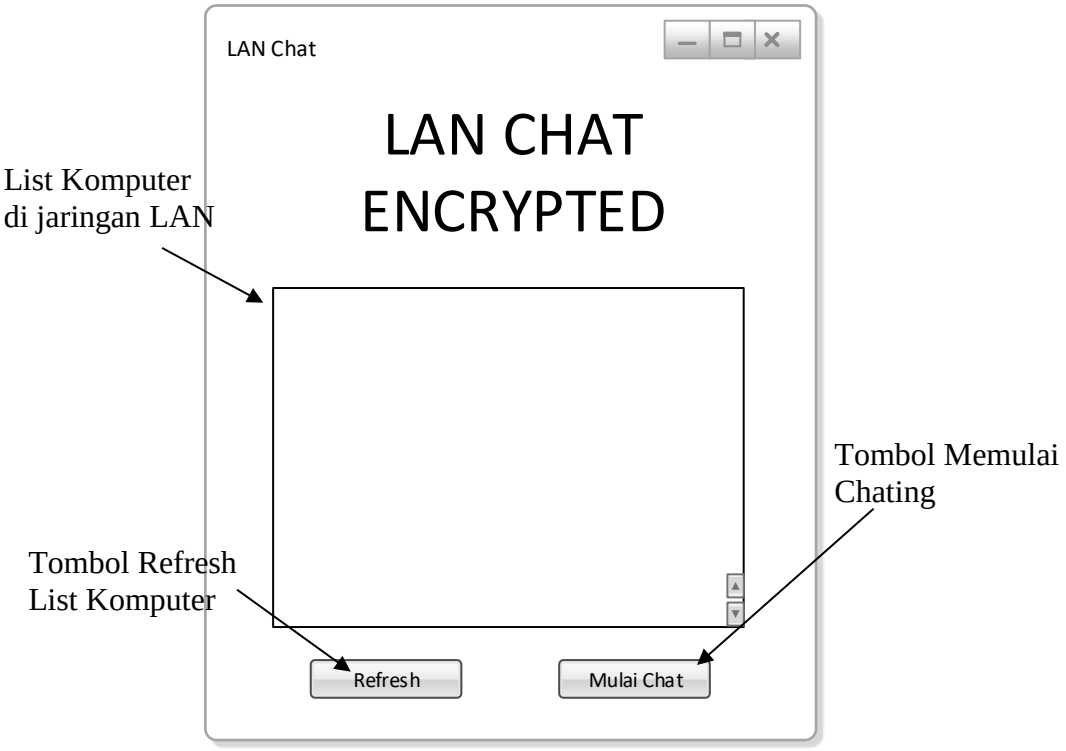


Gambar 4.31 Proses menerima pesan mode aman – Pengirim

4.2.3 Desain User interface

Pada penelitian ini akan dibangun sebuah aplikasi untuk komunikasi IM dengan user interface yang dapat di ilustrasikan dengan gambar modelnya.

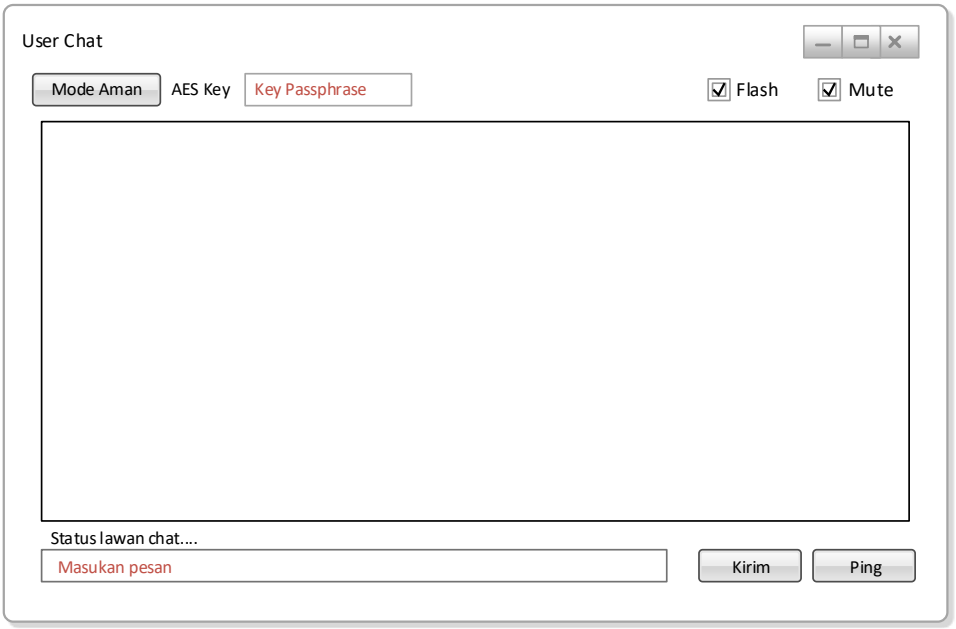
4.2.3.1 Desain interface untuk insialisasi IM



Gambar 4.32 Desain antar muka awal inisialisasi server IM

Desain antar muka awal untuk inisialisasi server IM, di form ini terdapat list box yang nanti akan menampilakn List komputer yang ada di jaringan, selanjutnya terdapat tombol refresh yang nantinya akan merefresh list komputer tadi, selanjutnya ada tombol mulai chat yang berfungsi untuk memanggil form utama.

4.2.3.2 Desain interface untuk Form Utama



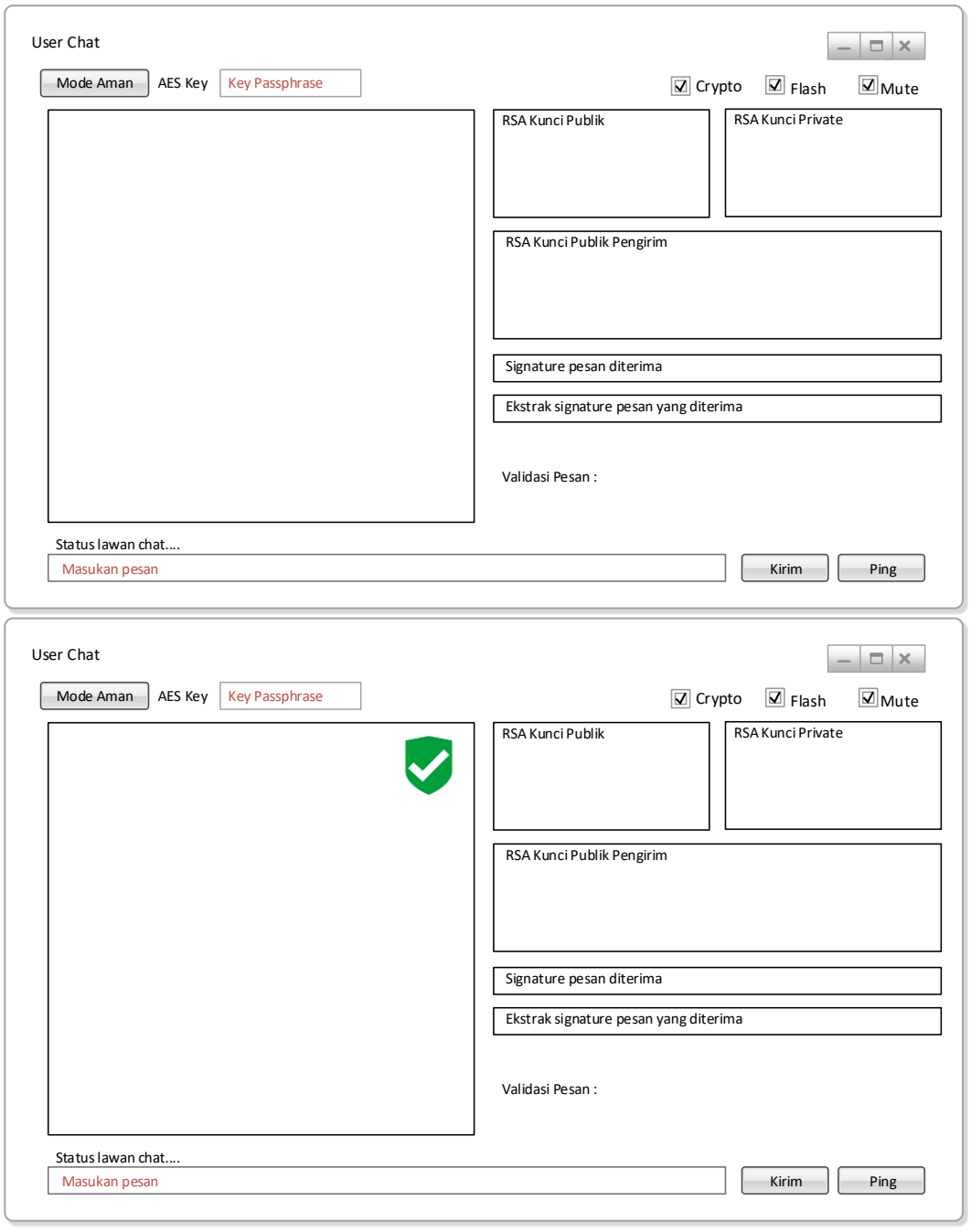
Gambar 4.33 Desian Form Chat mode biasa

Gambar 4.34 Desian Form Chat dalam mode aman

Pada Gambar 4.34 form utama diload ketika akan melakukan chating dengan lawan chating di jaringan lan. pada mode secure di Gambar 4.32 ditandai

dengan icon mode aman, dimana data pesan yang dikirim (plaintext) sudah di enkripsi.

4.2.3.3 Desain interface untuk Form Utama (Pengujian)



Gambar 4.35 Desian Form Chat mode biasa dan mode aman pengujian

BAB V
HASIL PENELITIAN DAN PEMBAHASAN

5.1 Hasil Penelitian

5.1.1 Pengujian White Box

White box testing adalah metode desain *test case* yang menggunakan struktur kontrol desain prosedural untuk mendapatkan *test case*. Dalam pelaksanaannya, teknik pengujian *white box* ini mempunyai empat langkah, yaitu sebagai berikut:

1. Menggambar *flowgraph* (Aliran Kontrol) yang ditransfer dari *flowchart*
2. Menghitung *cyclomatic complexsity* (CC) untuk *flowgraph* yang telah dibuat.
3. Menentukan jalur pengujian dari *flowgraph* berjumlah sesuai dengan *cyclomatic complxity* yang telah ditentukan
4. *Bases path testing*, yaitu teknik yang memungkinkan perancang *test case* mengukur kompleksitas logis dari desain procedural dan menggunakannya sebagai pedoman untuk menetapkannya basis set dari jalur eksekusi.

Hasil rancangan dengan menggunakan *white box testing* pada alur program, struktur logika program atau prosedur programnya dengan cara pemetaan flowchart ke dalam *flowgraph* kemudian menghitung besarnya jumlah *edge* dan *node* dimana jumlah *edge* dan *node* ini akan menentukan besarnya *cyclomatic compexity* (CC). Perhitungan CC untuk melihat kesamaan nilai antar *white box testing*, jika nilai

V(G) = CC pada *white box testing* dengan *bases path testing* maka proses pengujian telah berhasil.

Beberapa istilah saat pembuatan *flowgraph:*

1. *Node,* yaitu lingkaran pada *flowgraph* yang menggambarkan satu atau lebih perintah prosedural
2. *Edge,* yaitu tanda panah yang menggambarkan aliran kontrol dan setiap *node* harus mempunyai tujuan *node*
3. *Region,* yaitu daerah yang dibatasi oleh *node* dan *edge* dan untuk menghitung *region* daerah di luar *flowgraph* juga harus dihitung
4. *Predicate Node,* yaitu kondisi yang terdapat pada *node* dan mempunyai karakteristik dua atau lebih *edge* lainnya.

1. Prosedur untuk proses pengamanan IM

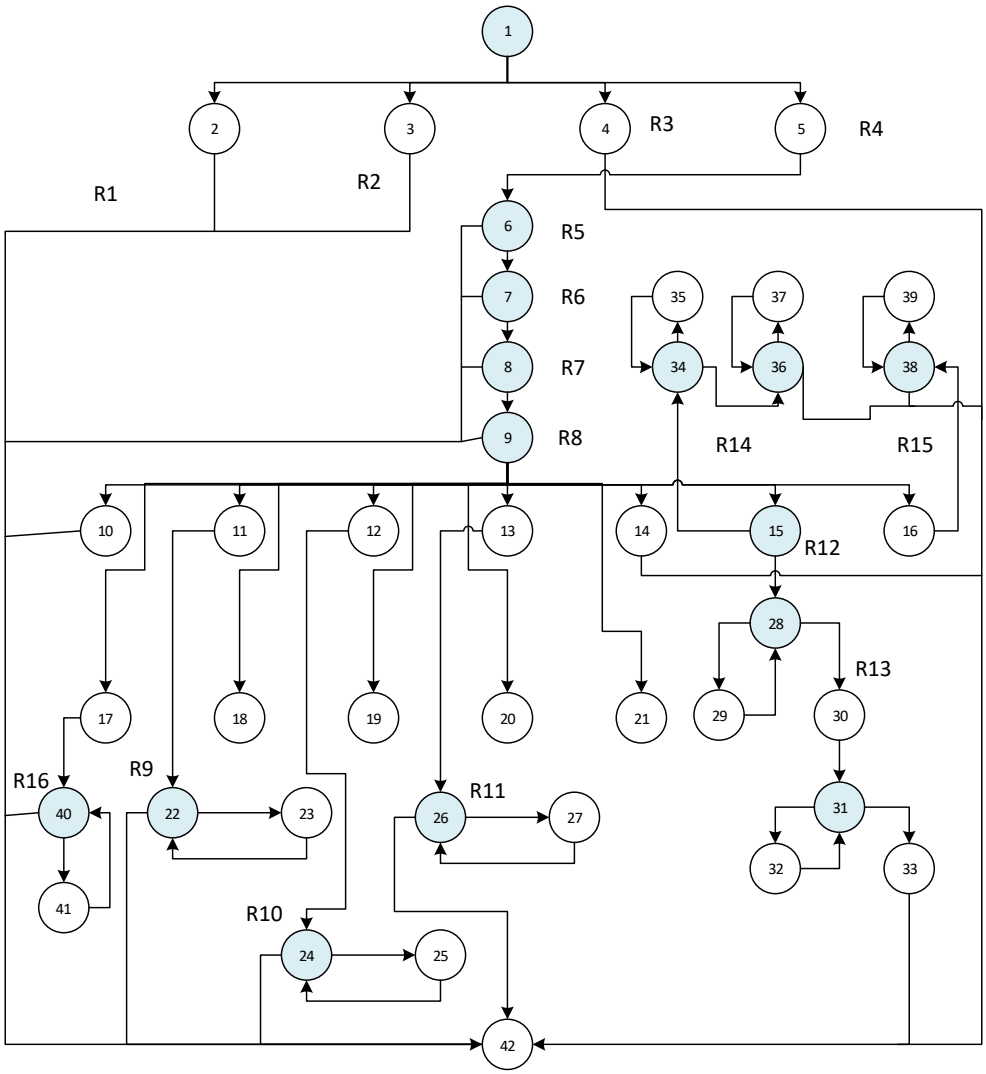
Prosedur progres work pesan beserta *Code Message*

```
Private Sub BackgroundWorker1_ProgressChanged(ByVal sender As System.Object, ByVal e As
System.ComponentModel.ProgressChangedEventArgs) Handles
BackgroundWorker1.ProgressChanged                                                    1
    Select Case e.ProgressPercentage
      Case -1 ' Raised From Exception in Receiving Loop in BackgroundWorker()
        SendFinalDisconnectMessage = False
2       SetChatState(False)
        Me.Text = Me.Text & " [Koneksi Putus]"

3     Case -2 ' Inisialisasi koneksi gagal ==========================================
        SendFinalDisconnectMessage = False
        Me.Text = "Gagal Menghubungi Client!!"
        MessageBox.Show("Tak ada respon " & ConnectTo & " ...", "Gagal menghubungi!",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        Me.Close()

4     Case 1 ' Buat Koneksi ================================================
        SetChatState(True) ' Enable the Chat Interface

      Case 0 ' Pesan Normal diterima                                                 5
==============================================
        If Not IsNothing(e.UserState) AndAlso TypeOf e.UserState Is String Then
6         Dim msg As String = CType(e.UserState, String)
          Dim values() As String = msg.Split(FieldMarker)                            8
          If values.Length >= 2 Then ' Kemampuan untuk mengirim lebih dari 2 field
==============================
7         Dim strCode As String = values(0)
          Dim value As String = values(1) ' Semua pesan harus mempunyai 2 fields (even if the
second isn't used)

            Try
              Dim code As MessageCodes = [Enum].Parse(GetType(MessageCodes),
strCode.ToUpper)
              Select Case code                                                       9
                Case MessageCodes.ACK ' Signal Ack diterima                          10
================================

                Case MessageCodes.GREETING ' terima nama dari client lain            11
==================================
                  Me.Text = value
                  DisplayMessage(Color.Red, value & " Terhubung")
                  If Not cbMute.Checked Then                                         23
22                  ChatForm.Play(Sound.ChatOpen)
                  End If
                  SendMessage(MessageCodes.GREETINGRESPONSE,
Environment.MachineName) ' kirim kembali nama kita ==================

                Case MessageCodes.GREETINGRESPONSE ' telah mengirim nama dan          12
menunggu respon balik ==================
                  Me.Text = value
                  DisplayMessage(Color.Red, value & " Terhubung")
24                If Not cbMute.Checked Then                                         25
```

```
                ChatForm.Play(Sound.ChatOpen)
            End If

        Case MessageCodes.TYPING ' sisi lain sedang mengetik
========================
            Static index As Integer = -1
            index = index + 1
            If index > Typing.GetUpperBound(0) Then
                index = 0
            End If
            lblStatus.Text = value & " " & Typing(index)

        Case MessageCodes.TYPINGCANCEL ' sisi lain sedang menghapus pesannya di
text box ==============
            lblStatus.Text = ""

        Case MessageCodes.TEXT ' Terima pesan dari client lain
========================
            If Not cbEnkript.Checked Then
                DisplayMessage(Color.DarkBlue, value)
            Else
                input = value
                passPhrase = tbKeyAES.Text
                'Dekripsi pesan setelah diterima
                decrypt = RijndaelSimple.Decrypt(input, passPhrase, saltValue,
                                                 hashAlgorithm, passwordIterations,
                                                 initVector, keySize)
                Dim signverify As String = decrypt
                Dim rcvsignature As String = signverify.Split("^"c)(0)
                Dim mixmessage As String() = signverify.Split("^"c)
                Dim clmessage As String = mixmessage(mixmessage.Length - 1)
                Dim extractsign As String =
Encryption.Signature.GetMd5Hash(hashsignature, clmessage)
                If rcvsignature = extractsign Then 'Validasi signature pesan
                    DisplayMessage(Color.DarkGreen, clmessage)
                Else
                    DisplayMessage(Color.DarkOrange, clmessage)
                End If

            End If
            If Not cbMute.Checked Then
                ChatForm.Play(Sound.ChatMessage)
            End If
            If cbFlash.Checked Then
                FlashWindow(Me.Handle)
            End If

        Case MessageCodes.PAGE ' Terima ping dari client lain
========================
            If Not cbMute.Checked Then
                ChatForm.Play(Sound.ChatPage)
            End If

        Case MessageCodes.DISCONNECTED ' Client lain telah menutup koneksinya
====================
            ContinueProcessingMessages = False
            lblStatus.Text = ""
            DisplayMessage(Color.Red, value)
```

```
                    Me.Text = Me.Text & " [Terputus]"
                    SetChatState(False)
                    SendFinalDisconnectMessage = False
                    If Not cbMute.Checked Then
                        ChatForm.Play(Sound.ChatClose)
                    End If

                Case MessageCodes.REQUESTSECURE 'Pertukaran Kunci simetris AES
                    MessageBox.Show("Anda diajak ke MODE AMAN",
                                    "Pertukaran Kunci RSA", MessageBoxButtons.OK,
MessageBoxIcon.Information)
                    tbReceive.Text = value
                    SendSession(MessageCodes.KEYEXCHANGE, txtPublicKey.Text)

                Case MessageCodes.KEYEXCHANGE 'Terima Kunci Publik RSA dan enkripsi
kunci AES dengan RSA Pengirim
                    tbReceive.Text = value
                    Try
                        Dim EncryptedMessage As RSAResult
                        EncryptedMessage = RSAEncryption.Encryptor(tbKeyAES.Text,
tbReceive.Text)
                        SendChiperkey(MessageCodes.KEYSIMETRICPARAMETER,
EncryptedMessage.AsBase64String)
                    Catch ex As Exception
                        MessageBox.Show(ex.Message, "Error Chiperkey", MessageBoxButtons.OK,
MessageBoxIcon.Error)
                    End Try

                Case MessageCodes.KEYSIMETRICPARAMETER
                    Try
                        Dim DecryptedMessage As RSAResult =
RSAEncryption.Decrypt(Convert.FromBase64String(value), txtPrivateKey.Text)
                        tbKeyAES.Text = DecryptedMessage.AsString
                        cbEnkript.Checked = True
                    Catch ex As Exception
                        MessageBox.Show(ex.Message, "Error Dechiperkey",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                    End Try
                    MessageBox.Show("Kunci AES berhasil di terima", "AES Received",
MessageBoxButtons.OK, MessageBoxIcon.Warning)
                    PictureBox1.Visible = True

                Case MessageCodes.MODE
                    MessageBox.Show("Teman Anda sudah keluar dari mode aman", "Mode
Berubah", MessageBoxButtons.OK, MessageBoxIcon.Warning)
                    PictureBox1.Visible = False
                    cbEnkript.Checked = False

            End Select
        Catch ex As Exception
        End Try
    End If
End If
End Select
End Sub
```

2. Flowgraph untuk proses pengamanan IM



Gambar 5.1 Flowgraph proses pengaman IM

Dari *flowgraph* diatas, maka didapatkan:

Region (R) = 16

Node (N) = 41

Edge (E) = 55

Predicate Node (P) = 15

a. Menghitung Nilai *Cyclomatic Complexity* (CC)

Cyclomatic complexity digunakan untuk mencari jumlah path dalam satu *flowgraph*. *Cyclomatixcomplexity*V(G) untuk grafikalir dihitung dengan rumus:

V(G) = E – N + 2

= 55 – 41 + 2

V(G) = 16

atau, V(G) = P + 1

= 15 + 1

V(G) = 16

CC = R1 - R16

b. Menentukan *Basis Path*

Basis set yang dihasilkan dari jalur independent secara linier adalah jalur sebagai berikut:

Tabel 5.1 Pengujian Basis Path

Jalur	Path	Cek
1	1-2-42	Ok
2	1-3-42	Ok
3	1-4-42	Ok
4	1-5-6-42	Ok
5	1-5-6-7-42	Ok
6	1-5-6-7-8-42	Ok
7	1-5-6-7-8-9-42	Ok
8	1-5-6-7-8-9-10-42	Ok

9	1-5-6-7-8-9-11-22-23-22-42	Ok
10	1-5-6-7-8-9-12-24-25-24-42	Ok
11	1-5-6-7-8-9-13-26-27-26-42	Ok
12	1-5-6-7-8-9-14-42	Ok
13	1-5-6-7-8-9-15-28-29-28-30-31-32-31-33-42	Ok
14	1-5-6-7-8-9-15-28-34-35-34-36-37-36-42	Ok
15	1-5-6-7-8-9-16-38-39-38-42	Ok
16	1-5-6-7-8-9-17-40-41-40-42	Ok

Ketika aplikasi dijalankan, maka terlihat bahwa semua basis set yang dihasilkan oleh simpul telah dieksekusi satu kali. Berdasarkan ketentuan tersebut dari segi kelayakan *software*, sistem ini telah memenuhi syarat.

5.1.2 Pengujian Black Box

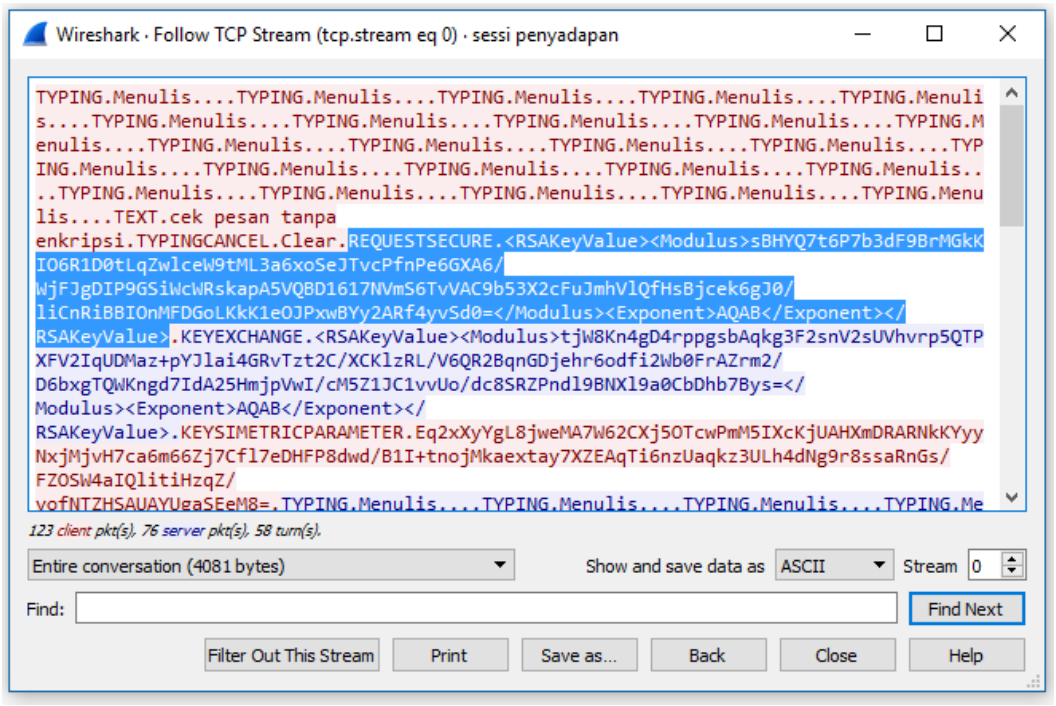
Pengujian *black box* dilakukan untuk memastikan bahwa suatu *event* atau masukan akan menjalankan proses yang tepat dan menghasilkan *output* sesuai dengan rancangan. Untuk contoh pengujian terhadap beberapa proses memberikan hasil sebagai berikut:

Tabel 5.2 Hasil Pengujian *Black Box* Terhadap Beberapa Proses

Input/Event	**Fungsi**	**Hasil yg Diharapkan**	**Hasil Uji**
Klik aplikasi	Inisialisasi Server IM	Icon Aplikasi IM berjalan pada sistem tray	Sesuai

Klik icon aplikasi pada sistem tray	Menampilkan form awal aplikasi LANChat dan list komputer	Form inisialisasi aplikasi Lanchat muncul beserta list komputer	Sesuai
Klik tombol refresh	Melakukan refresh list pada listbox list komputer	List komputer terupdate sesuai dengan komputer yang ada di jaringan	Sesuai
Klik tombol mulai chat	Meload form chat sesuai dengan lawan chating yang di pilih	Form Utama muncul	Sesuai
Klik mode aman tanpa mengisi AES Key	Melakukan pertukaran kunci Publik RSA (masuk ke mode aman)	Muncul messagebox error “Kunci AES belum terisi”	Sesuai
Klik mode aman dengan mengisi AES Key	Melakukan pertukaran kunci Publik RSA (masuk ke mode aman)	Tool Wireshark menampilkan adanya Pertukaran kunci Publik RSA (Lihat Gambar 5.3)	Sesuai
Klik tombol kirim ketika pesan pada text box terisi pesan	Mengirim isi pesan pada textbox pesan	Pesan terkirim pada lawan chat	Sesuai
Klik tombol ping	Mengirim ack data dengan ditandai nada ping	Bunyi ping pada lawan chat terdengar	Sesuai
Checkbox Flash tercentang/tidak	Agar window aplikasi flashing ketika ada pesan masuk dan pengguna tidak sedang membuka form utama	- Tercentang Form Utama flashing ketika ada pesan masuk - Tidak Flash dimatikan pada form utama	Sesuai
Checkbox Mute tercentang/tidak	Mematikan atau menghidupakan bunyi untuk semua event yang terjadi pada form utama	- Tercentang :Semua bunyi ketika aplikasi di jalankan mati - Tidak	Sesuai

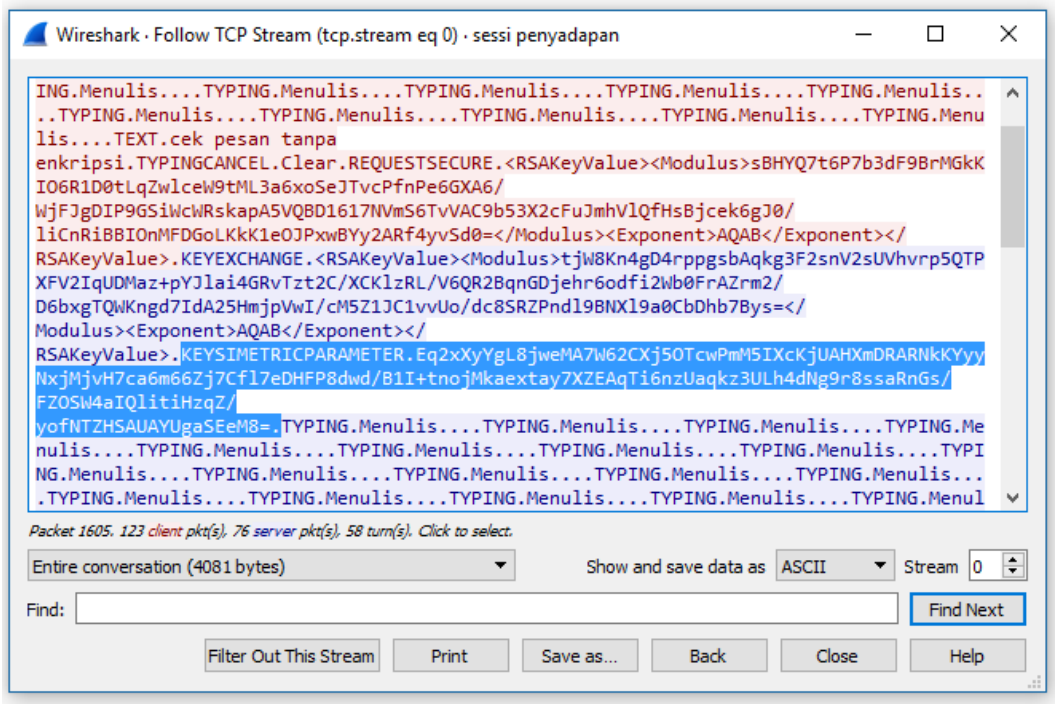
		Semua bunyi terdengan ketika event proses di jalankan	
Penyadapan pada proses pengaman distribusi kunci simetris	Melihat isi pesan pendistribusian kunci simetris pada jaringan (Masuk mode aman)	Tool Wireshark menampilkan adanya Distribusi kunci AES (Lihat Gambar 5.4)	Sesuai
Penyadapan pada proses pengamanan pengiriman pesan	Melihat isi pesan di jaringan ketika pesan tersebut sudah di mode aman (enkripsi oleh Algoritma AES)	Tool Wireshark menampilkan adanya Pesan yang telah terenkripsi dengan Algoritma AES (Lihat Gambar 5.4)	Sesuai



Gambar 5.3 Pengirim mengirimkan permintaan masuk ke Mode Aman

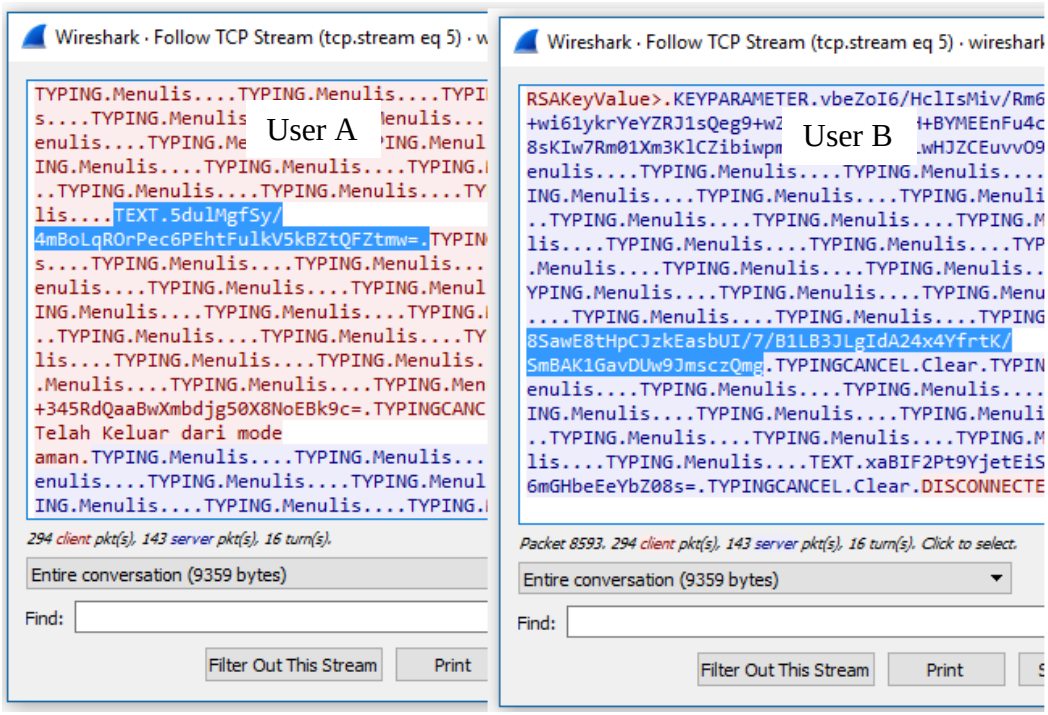
Gambar 5.3 merupakan hasil pantauan data yang lewat di jaringan menggunakan tool pengendus paket Wireshark, bidang yang di blok dengan warna biru merupakan request pengirim untuk meminta masuk ke mode aman sekaligus

mengirimkan kunci public RSA miliknya ke penerima pesan. Selanjutnya penerima request tadi membalas dengan mengirim kunci public miliknya ke pengirim request dengan diawali dengan [KEYEXCHANGE] dilanjutkan dengan kunci public RSA yang tulisannya berwarna biru.



Gambar 5.4 Pengirim mengirimkan Kunci AES yang telah terenkripsi

Pada Gambar 5.4 setelah penerima request mengirimkan kunci public RSA, pengirim mengenkripsi kunci simetri AES dengan kunci public RSA penerima lalu mengirim *chiperkey* AES ke penerima dengan ditandai string [KEYSIMETRICPARAMETER] yang telah di blok dengan warna biru. Disini proses pengamanan distribusi kunci simetris berlangsung, kunci asli AES tak bias dibaca karena sudah di enkripsi dengan RSA.



Gambar 5.5 Komunikasi pesan yang telah terenkripsi antar client

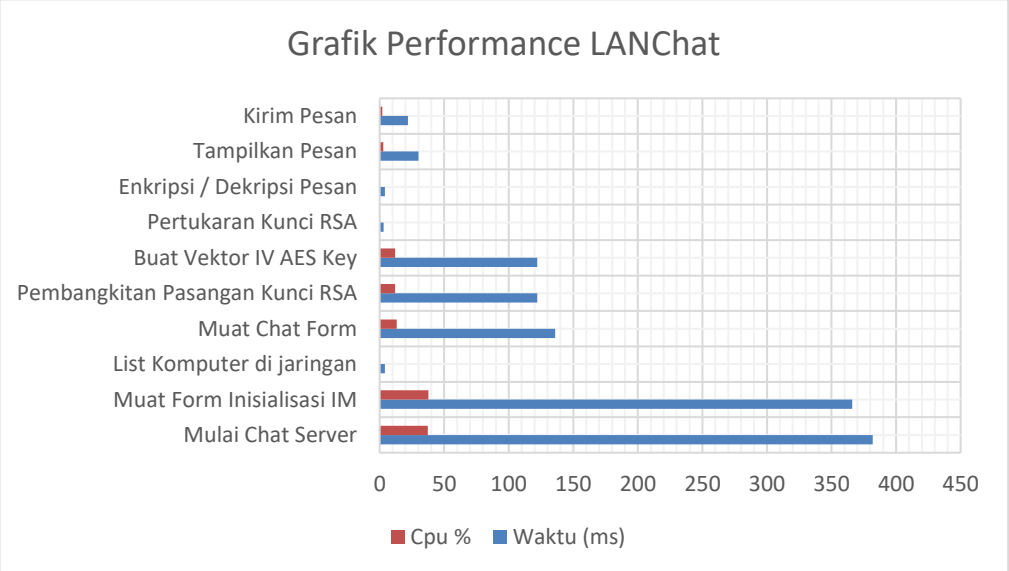
Gambar 5.5 memperlihatkan komunikasi antar pengguna IM telah di enkripsi sebelumnya sebelum pesan tersebut di kirim melalui jaringan. Pesan pengirim (user A) di blok dengan warna biru sebelah kanan, sedangkan penerima (user B) membalas pesan ke pengirim pesan berada disebelah kanan dengan block warna biru.

5.1.3 Pengujian Performance

Pada tahap ini pengujian dilakukan untuk mengetahui penggunaan resource pc yang digunakan untuk menjalankan aplikasi LAN Chat, mulai dari pengukuran penggunaan processor dan waktu eksekusi perintah. Berikut merupakan tabel hasil pengukuran pada palikasi LANChat dengan menggunakan visual studio analyzer.

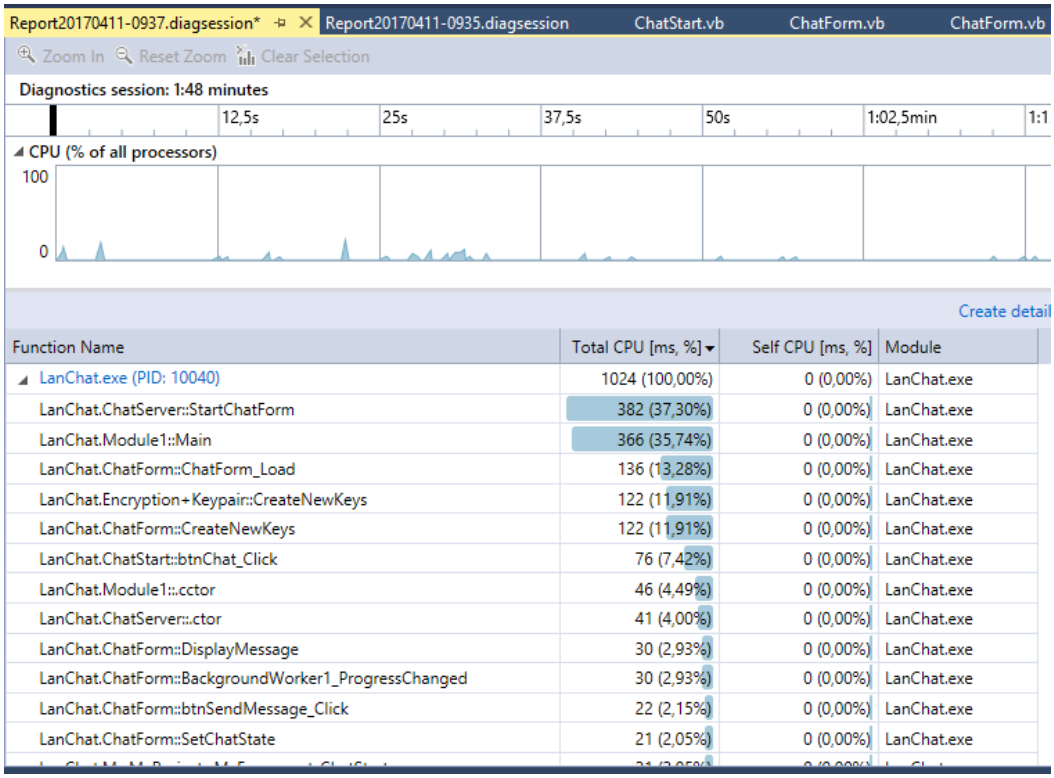
Tabel 5.3 Pengukuran waktu proses utama pada aplikasi LAN Chat

No.	Proses	Waktu (ms)	Cpu %
1	Mulai Chat Server	382	37,3
2	Muat Form Inisialisasi IM	366	37,74
3	List Komputer di jaringan	4	0,39
4	Muat Chat Form	136	13,28
5	Pembangkitan Pasangan Kunci RSA	122	11,91
6	Buat Vektor IV AES Key	122	11,91
7	Pertukaran Kunci RSA	3	0,29
8	Enkripsi / Dekripsi Pesan	4	0,39
9	Tampilkan Pesan	30	2,93
10	Kirim Pesan	22	2,15



Gambar 5.6 Grafik pengukuran waktu proses aplikasi LAN Chat

Pada gambar 5.6 proses pertukaran kunci, enkripsi dan dekripsi, serta pengiriman pesan rata rata mempunyai waktu yang singkat diikuti proses pembangkitan pasangan kunci RSA, pembuatan kunci AES, muat chat form yang rata rata memakan waktu 100-150ms. Yang terakhir adalah proses inisialisasi server chat dan load form Inisialisasi memakan waktu sekitar 350-400ms. Proses-proses tersebut tergantung dari kecepatan CPU sebuah komputer. Jika dihitung total waktu untuk semua proses pada palikasi LAN Chat rata-rata berkisar antara 1 sampai 1.5 detik.



Gambar 5.7 Pengukuran realtime waktu proses aplikasi LAN Chat

Pengujian performance pada Gambar 5.7 dengan total lama waktu dalam sistem IM yang dibuat yakni 1024ms untuk keseluruhan proses yang ada dalam sistem

pengamanan pengiriman pesan dengan menggunakan metode kombinasi RSA dan AES.

Dari hasil pengujian dapat disimpulkan untuk uji *black box* yang meliputi uji *input*, proses dan *output* dengan acuan rancangan perangkat lunak yang sudah dibuat sebelumnya telah terpenuhi dengan hasil sesuai dengan rancangan.

5.2 Pembahasan

5.2.1 Kebutuhan Hardware dan Software

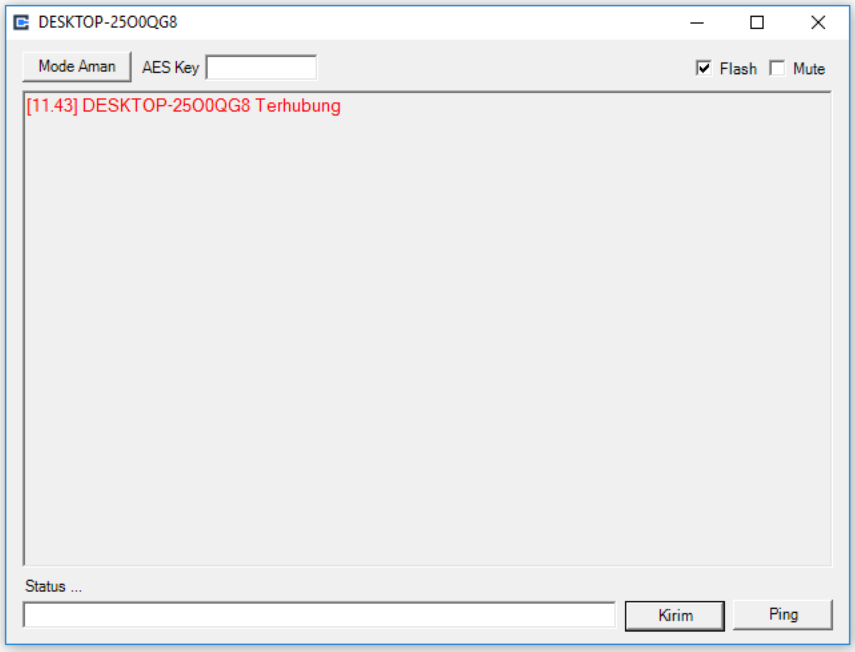
Agar sistem dapat berjalan secara maksimal maka disarankan untuk menggunakan perangkat hardware dan software sebagai berikut:

- Prosessor minimal 600 MHz
- VGA Min 16 Bit
- Resolusi minimal 1024 x 768
- Ram Minimal 1GB
- Harddisk minimal ruang Kosong 100 MB
- Mouse
- OperatingSistem:Windows 2000/XP/7
- Jaringan LAN pada tiap user

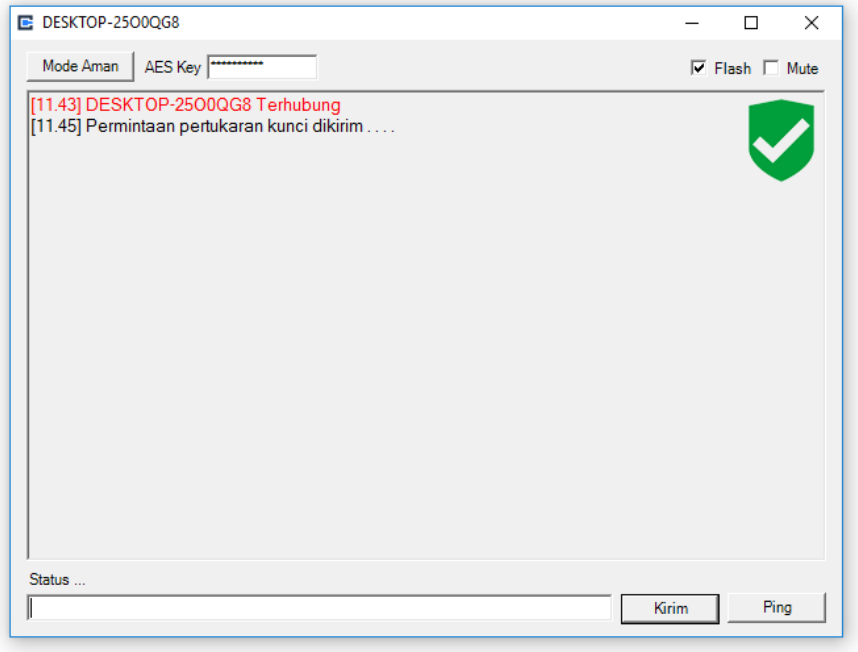
5.2.2 Langkah-Langkah Penggunaan Sistem

Gambar 5.8 Tampilan Form awal LANChat

Setelah aplikasi di eksekusi, program akan menginisialisasi Server IM dengan di tandai oleh icon LANChat pada sistem tray yang sudah dijalankan terlebih dahulu. Selanjut dengan mengklik icon tersebut sessi komunikasi dibuka dengan memunculkan tampilan awal pada Gambar 5.9



Gambar 5.9 Form utama mode biasa

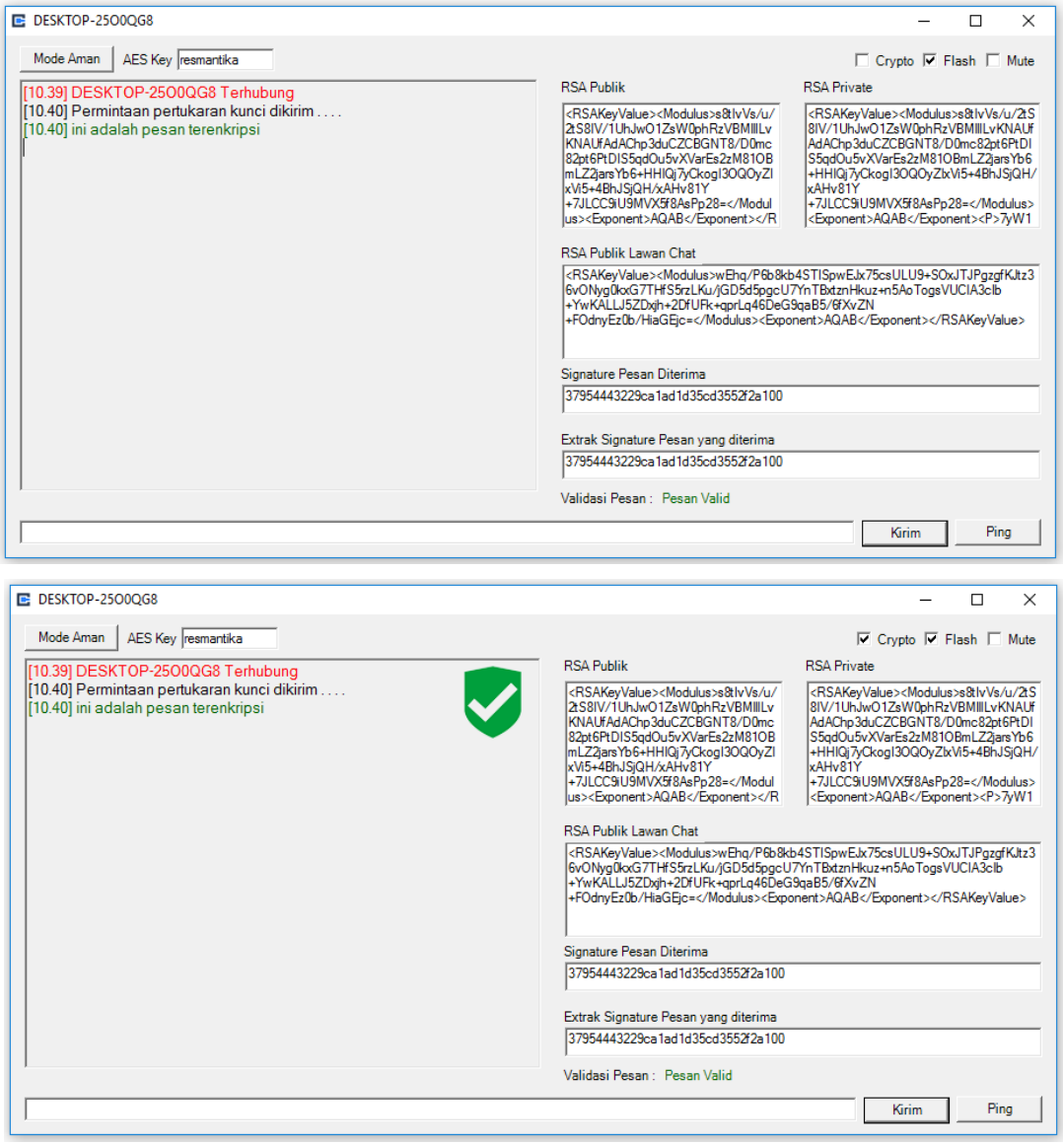


Gambar 5.10 Form utama dalam mode aman

Pada Gambar 5.4 form utama diload ketika akan melakukan chating dengan lawan chating di jaringan lan. Jika anda akan melakukan chating dengan mode aman, maka klik tombol [mode aman], pada mode secure di Gambar 5.5 ditandai

dengan icon mode aman di kanan atas kotak pesan, dimana data pesan yang dikirim (plaintext) sudah di enkripsi.

Form Utama (Pengujian)



Gambar 5.11 Form utama mode biasa dan mode aman (pengujian)

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan pada proses pengamanan distribusi kunci dan enkripsi pesan pada aplikasi LAN Chat (IM) serta pembahasan yang telah diuaraikan sebelumnya, maka dapat ditarik suatu kesimpulan bahwa:

1. Penelitian ini merekomendasikan bahwa pengaman distribusi kunci dan enkripsi dengan metode kombinasi algoritma RSA dan AES sangat effisien untuk komputer dengan spesifikasi yang minim.
2. Penggunaan panjang kunci RSA sebesar 1024bit dan panjang kunci AES sebesar 256bit merupakan parameter-parameter panjang kunci yang ideal untuk aplikasi IM dimana proses komputasinya rata-rata 1-4ms.
3. Penggunaan metode kombinasi RSA dan AES untuk pengamanan pengiriman pesan instan yang diterapkan pada aplikasi LANChat telah memenuhi aspek *confidentiality*, *non-repudiation* serta *authentication* dibuktikan dengan pantauan data pada jaringan, serta proses sistem yang ringkas, sehingga penerapannya cocok pada pengguna IM dengan resource kecil.

6.2 Saran

Setelah melakukan penelitian dilakukan pada proses pengamanan distribusi kunci dan enkripsi pesan pada aplikasi LAN Chat (IM) serta pembahasan yang telah diuaraikan sebelumnya, ada beberapa saran yang perlu di perhatikan untuk melengkapi sistem pengamanan IM, yaitu sebagai berikut:

1. Pemberian *signature* dengan metode RSA diperlukan jika isi pesan berupa dokumen, file video atau audio maupun file lainya, untuk pesan yang isinya text saja metode pemberian signature menggunakan RSA setelah diteliti dirasa tidak perlu, karena proses *signaturing* menjadi dua kali enkripsi dan ini berlebihan.
2. Untuk pengembangan selanjutnya sistem IM dibuatkan model chat group.
3. Untuk pengembangan selanjutnya penambahan pengamanan sistem login dengan menggunakan SSL atau TLS maupun menggunakan metode pengamanan lain.

DAFTAR PUSTAKA

Alimohammadi, M, and A A Pouyan. 2014. "Performance Analysis of Cryptography Methods for Secure Message Exchanging in VANET." *International Journal of Scientific & Engineering Research* 5(2): 1–9.

Allen, Lee. 2015. 1 Climate Change 2013 - The Physical Science Basis *Advanced Penetration Testing for Highly-Secured Environments: The Ultimate Security Guide*. http://ebooks.cambridge.org/ref/id/CBO9781107415324A009.

Al Barghuthi, Nedaa B., and Huwida Said. 2013. "Social Networks IM Forensics: Encryption Analysis." *Journal of Communications* 8(11): 708–15.

Borisov, Nikita, U C Berkeley, Ian Goldberg, and Eric Brewer. 2004. "Off-the-Record Communication, or , Why Not To Use PGP." *Wpes*: 77–84.

Casey, Donal. 2007. "Building a Secure Instant Messaging Environment." *Network Security* 2007(1): 18–20.

Chandra, Wico. 2011. "Kriptografi Dan Algoritma RSA." *Makalah II2092 Probabilitas dan Statistik - Semantik* 1, No.15(13509094): 1–5.

Guo, Wenping, Zhenlong Li, Ying Chen, and Xiaoming Zhao. 2009. "Security Design for Instant Messaging System Based on RSA and Triple DES." In *Proceedings of 2009 International Conference on Image Analysis and Signal Processing, IASP 2009, ,* 415–18.

Kaminsky, Alan. "Rijndael Algorithm (Advanced Encryption Standard) AES Selection Process."

Konheim, Alan G. 2005. Notes *COMPUTER SECURITY AND CRYPTOGRAPHY*.

Kromodimoeljo, Sentot. 2010. *TEORI & APLIKASI KRIPTOGRAFI*. SPK IT

Consulting.

Kusumah, Angga, Maman Abdurrohman, and Dodi Wisaksono Sudiharto. 2012. "Secure Chatting (Instant Messaging) Menggunakan Metode Enkripsi Blowfish, Twofish, Dan Aes." : 8.

MCCLURE, STUART, JOEL SCAMBRAY, and GEORGE KURTZ. *HACKING EXPOSED 6: ™ NETWORK SECURITY SECRETS & SOLUTIONS*.

Plocki, Pawel. 2012. "Hacking On Deman." *The Guide to Backtrack* 1. http://www.backtrack-linux.org/documents/Hakin9_On_Deman_03_2012_Teasers.pdf.

Roy, Ayan. 2016. "Brief Comparison of RSA and Diffie-Hellman (Public Key) Algorithm." 1(1): 1–4.

Schneier, Bruce. 1996. 13 Government Information Quarterly *Applied Cryptography: Protocols, Algorithm, and Source Code in C*.

Selent, Douglas. 2001. "Advanced Encryption Standard." *Federal Information Processing Standard, FIPS-197* 197(Fips 197): 1–10. http://cmpe.emu.edu.tr/Courses/CMPE553/Notes/Ch5.AES.doc.

Tahsin, Tahmina, Lazeeb Faiz Choudhury, and Md Lutfur Rahman. 2008. "Peer-to-Peer Mobile Applications Using JXTA/JXME." In *Proceedings of 11th International Conference on Computer and Information Technology, ICCIT 2008*, , 702–7.

Techcrunch.com, and Jon Russell. 2014. "Secure Messaging App Telegram Adds Usernames And Snapchat-Like Hold-To-View For Media | TechCrunch." : 6–7.

LAMPIRAN

A. Listing Program

1. Module1

```
1.  Module Module1
2.
3.      Private CS As New ChatServer
4.
5.      Public Sub Main()
6.          Application.Run(CS)
7.      End Sub
8.
9.  End Module
```

2. ChatServer

```
1.  Imports System.Net
2.  Imports System.Threading
3.  Imports System.Net.Sockets
4.  Public Class ChatServer
5.      Inherits ApplicationContext
6.
7.      Private Server As TcpListener = Nothing
8.      Private ServerThread As Thread = Nothing
9.      Private WithEvents Tray As New NotifyIcon
10.     Private Threads As New List(Of Thread)
11.
12.     Public Sub New()
13.         Tray.Icon = My.Resources.Chat
14.         Tray.Visible = True
15.         Tray.Text = "LAN Chat"
16.
17.         Server = New TcpListener(IPAddress.Any, 50000) ' <-- Listen on
    Port 50,000
18.         ServerThread = New Thread(AddressOf ConnectionListener)
19.         ServerThread.IsBackground = True
20.         ServerThread.Start()
21.     End Sub
22.
23.     Private Sub ConnectionListener()
24.         Try
25.             Server.Start()
26.             While True
27.                 Dim client As TcpClient = Server.AcceptTcpClient ' Tahan
    sebelum ada permintaan dari client lain
28.                 Dim T As New Thread(AddressOf StartChatForm)
29.                 Threads.Add(T)
30.                 T.Start(client)
31.             End While
32.         Catch ex As Exception
33.             MessageBox.Show("Takbisa menerima koneksi", "Aplikasi Chat
    Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
34.         End Try
35.         Application.ExitThread()
```

```
    End Sub

    Private Sub StartChatForm(ByVal client As Object)
        Application.Run(New ChatForm(CType(client, TcpClient))) ' Start a New ChatForm with the TcpClient Connection
        Threads.Remove(Thread.CurrentThread) ' We don't get here until the ChatForm is closed
    End Sub

    Private Sub Tray_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Tray.Click
        ChatStart.Show()
    End Sub

    Private Sub ChatServer_ThreadExit(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.ThreadExit
        Tray.Visible = False
    End Sub

End Class
```

3. ChatStart

```
Imports System.Runtime.InteropServices
Public Class ChatStart

    Private Sub ChatStart_Shown(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Shown
        btnRefresh.PerformClick()
    End Sub

    Private Sub btnRefresh_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRefresh.Click
        btnRefresh.Enabled = False
        lbComputers.DataSource = Nothing
        BackgroundWorker1.RunWorkerAsync()
    End Sub

    Private Sub BackgroundWorker1_DoWork(ByVal sender As System.Object, ByVal e As System.ComponentModel.DoWorkEventArgs) Handles BackgroundWorker1.DoWork
        e.Result = ChatStart.GetNetworkComputers()
    End Sub

    Private Sub BackgroundWorker1_RunWorkerCompleted(ByVal sender As Object, ByVal e As System.ComponentModel.RunWorkerCompletedEventArgs) Handles BackgroundWorker1.RunWorkerCompleted
        lbComputers.DataSource = CType(e.Result, List(Of String))
        btnRefresh.Enabled = True
    End Sub

    Private Sub lbComputers_DoubleClick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles lbComputers.DoubleClick
        btnChat.PerformClick()
    End Sub

    Private Sub btnChat_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnChat.Click
        If lbComputers.SelectedIndex <> -1 Then
```

```
29.             Dim chat As New ChatForm(lbComputers.SelectedItem.ToString)
30.             chat.Show()
31.             Me.Close()
32.         End If
33.     End Sub
34.
35. #Region "NetServerEnum_API"
36.
37.     Public Const SV_TYPE_ALL As Integer = &HFFFFFFFF
38.
39.     Public Structure SERVER_INFO_101
40.         Public Platform_ID As Integer
41.         <MarshalAsAttribute(UnmanagedType.LPWStr)> Public Name As String
42.         Public Version_Major As Integer
43.         Public Version_Minor As Integer
44.         Public Type As Integer
45.         <MarshalAsAttribute(UnmanagedType.LPWStr)> Public Comment As
    String
46.     End Structure
47.
48.     Public Declare Unicode Function NetServerEnum Lib "Netapi32.dll" ( _
49.         ByVal Servername As Integer, ByVal Level As Integer, ByRef
    Buffer As Integer, ByVal PrefMaxLen As Integer, _
50.         ByRef EntriesRead As Integer, ByRef TotalEntries As Integer,
    ByVal ServerType As Integer, _
51.         ByVal DomainName As String, ByRef ResumeHandle As Integer) As
    Integer
52.
53.     Public Declare Function NetApiBufferFree Lib "Netapi32.dll" (ByVal
    lpBuffer As Integer) As Integer
54.
55.     Public Shared Function GetNetworkComputers(Optional ByVal DomainName
    As String = Nothing) As List(Of String)
56.         Dim ServerInfo As SERVER_INFO_101
57.         Dim MaxLenPref As Integer = -1
58.         Dim level As Integer = 101
59.         Dim ResumeHandle As Integer = 0
60.         Dim ret, EntriesRead, TotalEntries, BufPtr, CurPtr As Integer
61.         Dim ReturnList As New List(Of String)
62.
63.         Try
64.             ret = NetServerEnum(0, level, BufPtr, MaxLenPref,
    EntriesRead, TotalEntries, SV_TYPE_ALL, DomainName, ResumeHandle)
65.             If ret = 0 Then
66.                 CurPtr = BufPtr
67.                 For i As Integer = 0 To EntriesRead - 1
68.                     ServerInfo = CType(Marshal.PtrToStructure(New
    IntPtr(CurPtr), GetType(SERVER_INFO_101)), SERVER_INFO_101)
69.                     CurPtr = CurPtr + Len(ServerInfo)
70.                     ReturnList.Add(ServerInfo.Name)
71.                 Next
72.             End If
73.             NetApiBufferFree(BufPtr)
74.         Catch ex As Exception
75.         End Try
76.
77.         Return ReturnList
78.     End Function
79.
80. #End Region
81. End Class
```

4. ChatForm

```
1.  Imports System.Text
2.  Imports System.Net.Sockets
3.  Imports System.Security.Cryptography
4.
5.  Public Class ChatForm
6.
7.      Public Enum Sound
8.          ChatOpen
9.          ChatMessage
10.         ChatPage
11.         ChatClose
12.     End Enum
13.
14.     Public Enum MessageCodes 'code pesan dibawah dengan --> HURUF BESAR
    <---
15.         ACK = 0
16.         TEXT = 1
17.         DISCONNECTED = 2
18.         GREETING = 3
19.         GREETINGRESPONSE = 4
20.         PAGE = 5
21.         TYPING = 6
22.         TYPINGCANCEL = 7
23.         REQUESTSECURE = 8
24.         KEYEXCHANGE = 9
25.         KEYSIMETRICPARAMETER = 10
26.         MODE = 11
27.     End Enum
28.
29.     'Bangkitkan Kunci Publik dan Private
30.     Private Sub CreateNewKeys()
31.         Dim Keys As Keypair = Keypair.CreateNewKeys
32.         txtPrivateKey.Text = Keys.Privatekey
33.         txtPublicKey.Text = Keys.Publickey
34.     End Sub
35.
36.     Private Sub ChatForm_Load(sender As Object, e As EventArgs) Handles
    MyBase.Load
37.         CreateNewKeys()
38.         PictureBox1.Visible = False
39.     End Sub
40.
41.     'Parameter AES
42.     Dim passPhrase As String
43.     Dim saltValue As String = "s@1tValued1is1d3ng4ns3mb4t4ng"    ' Bisa
    di isi dengan string apa saja
44.     Dim hashAlgorithm As String = "SHA1"                        ' Bisa
    juga pakai MD5
45.     Dim passwordIterations As Integer = 2                       ' Bisa
    dengan nomor sembarang
46.     Dim initVector As String = "@1B2c3D!e5$6g%&#"               ' Harus
    16Byte atau 16 karakter
47.     Dim keySize As Integer = 256                                '
    Panjang kunci bisa 128, 192, 256
48.     Dim input As String
49.     Dim encrypt As String
50.     Dim decrypt As String
51.     Dim hashsignature As MD5 = MD5.Create()
```

```
52.
53.     Private ConnectTo As String = ""
54.     Private ChatClient As TcpClient = Nothing
55.     Private FieldMarker As String = Chr(0)
56.     Private MessageMarker As String = Chr(1)
57.     Private Typing() As String = {"/", "-", "\", "|"}
58.     Private Const AckIntervalInSeconds As Integer = 60
59.     Private ContinueProcessingMessages As Boolean = True
60.     Private SendFinalDisconnectMessage As Boolean = False
61.     Private Shared Waves As New Dictionary(Of String, EmbeddedWave)
62.
63.     Public Sub New()
64.         InitializeComponent()
65.     End Sub
66.
67.     Public Sub New(ByVal ConnectTo As String)
68.         InitializeComponent()
69.         Me.ConnectTo = ConnectTo
70.         Me.Text = "Menunggu respon " & ConnectTo & " ..."
71.     End Sub
72.
73.     Public Sub New(ByVal ChatClient As TcpClient)
74.         InitializeComponent()
75.         Me.ChatClient = ChatClient
76.     End Sub
77.
78.     Private Sub ChatForm_Shown(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles Me.Shown
79.         AckTimer.Interval =
    TimeSpan.FromSeconds(AckIntervalInSeconds).TotalMilliseconds
80.         SetChatState(False)
81.         BackgroundWorker1.RunWorkerAsync()
82.     End Sub
83.
84.     Private Sub SetChatState(ByVal state As Boolean)
85.         cbMute.Enabled = state
86.         cbFlash.Enabled = state
87.         AckTimer.Enabled = state
88.         btnSendPage.Enabled = state
89.         btnSendMessage.Enabled = state
90.         tbMessageToSend.Enabled = state
91.     End Sub
92.
93.     Private Sub BackgroundWorker1_DoWork(ByVal sender As System.Object,
    ByVal e As System.ComponentModel.DoWorkEventArgs) Handles
    BackgroundWorker1.DoWork
94.         If IsNothing(ChatClient) Then ' Inisiasi koneksi
95.             Try
96.                 ChatClient = New TcpClient()
97.                 ChatClient.Connect(ConnectTo, 50000) ' Blokir sebelum
    koneksi dibuat
98.             Catch ex As Exception
99.                 ContinueProcessingMessages = False
100.                    BackgroundWorker1.ReportProgress(-2) ' Koneksi
    gagal
101.                End Try
102.            Else
103.                SendMessage(MessageCodes.GREETING,
    Environment.MachineName) ' Terima koneksi : Kirim info nama mesin
    komputer kita
104.            End If
```

```
105.
106.             If Not IsNothing(ChatClient) AndAlso ChatClient.Connected
    Then
107.                 BackgroundWorker1.ReportProgress(1) ' Aktifkan Form
    Chating
108.                 SendFinalDisconnectMessage = True
109.
110.                 Dim bytesRead As Integer
111.                 Dim buffer(1024) As Byte
112.                 Dim Messages As String = ""
113.                 Dim MessageMarkerIndex As Integer
114.                 While ContinueProcessingMessages
115.                     Try
116.                         bytesRead = ChatClient.GetStream.Read(buffer,
    0, buffer.Length) ' <-- Blok dulu sebelum data di terima
117.                         If bytesRead > 0 Then ' <-- Jika 0 maka
    koneksi tertutup atau tak ada respon balik
118.                             Messages = Messages &
    System.Text.ASCIIEncoding.ASCII.GetString(buffer, 0, bytesRead) ' Append
    the received data to our message queue
119.                             MessageMarkerIndex =
    Messages.IndexOf(MessageMarker) ' See if the End of Message marker is
    present
120.                             While MessageMarkerIndex <> -1 ' If we
    have received at least one complete message
121.                                 BackgroundWorker1.ReportProgress(0,
    Messages.Substring(0, MessageMarkerIndex)) ' Let the GUI handle the
    complete Message
122.                                 Messages = Messages.Remove(0,
    MessageMarkerIndex + 1) ' Remove the processed message
123.                                 MessageMarkerIndex =
    Messages.IndexOf(MessageMarker) ' See if there are more End of Message
    markers present
124.                             End While
125.                         End If
126.                     Catch ex As Exception
127.                         ContinueProcessingMessages = False
128.                         BackgroundWorker1.ReportProgress(-1)
129.                     End Try
130.                 End While
131.             End If
132.         End Sub

133.     Private Sub BackgroundWorker1_ProgressChanged(ByVal sender As
    System.Object, ByVal e As
    System.ComponentModel.ProgressChangedEventArgs) Handles
    BackgroundWorker1.ProgressChanged
134.             Select Case e.ProgressPercentage
135.                 Case -1 ' Raised From Exception in Receiving Loop in
    BackgroundWorker()
136.                     SendFinalDisconnectMessage = False
137.                     SetChatState(False)
138.                     Me.Text = Me.Text & " [Koneksi Putus]"
139.
140.                 Case -2 ' Inisialisasi koneksi gagal
    ========================================
141.                     SendFinalDisconnectMessage = False
142.                     Me.Text = "Gagal Menghubungi Client!!"
143.                     MessageBox.Show("Tak ada respon " & ConnectTo & "
    ...", "Gagal menghubungi!", MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation)
```

```
144.                         Me.Close()
145.
146.                     Case 1 ' Buat Koneksi
    ================================================
147.                         SetChatState(True) ' Enable the Chat Interface
148.
149.                     Case 0 ' Pesan Normal diterima
    ============================================
150.                         If Not IsNothing(e.UserState) AndAlso TypeOf
    e.UserState Is String Then
151.                             Dim msg As String = CType(e.UserState,
    String)
152.                             Dim values() As String =
    msg.Split(FieldMarker)
153.                             If values.Length >= 2 Then ' Kemampuan untuk
    mengirim lebih dari 2 field ============================
154.                                 Dim strCode As String = values(0)
155.                                 Dim value As String = values(1) ' Semua
    pesan harus mempunyai 2 fields (even if the second isn't used)
156.
157.                                 Try
158.                                     Dim code As MessageCodes =
    [Enum].Parse(GetType(MessageCodes), strCode.ToUpper)
159.                                     Select Case code
160.                                         Case MessageCodes.ACK ' Signal
    Ack diterima ================================
161.
162.                                         Case MessageCodes.GREETING '
    terima nama dari client lain ===================================
163.                                             Me.Text = value
164.                                             DisplayMessage(Color.Red,
    value & " Terhubung")
165.                                             If Not cbMute.Checked Then
166.
    ChatForm.Play(Sound.ChatOpen)
167.                                             End If
168.
    SendMessage(MessageCodes.GREETINGRESPONSE, Environment.MachineName) '
    kirim kembali nama kita ===================
169.
170.                                         Case
    MessageCodes.GREETINGRESPONSE ' telah mengirim nama dan menunggu respon
    balik ====================
171.                                             Me.Text = value
172.                                             DisplayMessage(Color.Red,
    value & " Terhubung")
173.                                             If Not cbMute.Checked Then
174.
    ChatForm.Play(Sound.ChatOpen)
175.                                             End If
176.
177.                                         Case MessageCodes.TYPING ' sisi
    lain sedang mengetik ==========================
178.                                             Static index As Integer = -1
179.                                             index = index + 1
180.                                             If index >
    Typing.GetUpperBound(0) Then
181.                                                 index = 0
182.                                             End If
183.                                             lblStatus.Text = value & " "
    & Typing(index)
```

```
184.
185.                                     Case MessageCodes.TYPINGCANCEL '
    sisi lain sedang menghapus pesannya di text box ==============
186.                                         lblStatus.Text = ""
187.
188.                                     Case MessageCodes.TEXT ' Terima
    pesan dari client lain ===========================
189.                                         If Not cbEnkript.Checked Then
190.
    DisplayMessage(Color.DarkBlue, value)
191.                                         Else
192.                                             input = value
193.                                             passPhrase =
    tbKeyAES.Text
194.                                             'Dekripsi pesan setelah
    diterima
195.                                             decrypt =
    RijndaelSimple.Decrypt(input, passPhrase, saltValue,
196.
    hashAlgorithm, passwordIterations,
197.
    initVector, keySize)
198.                                             Dim signverify As String
    = decrypt
199.                                             Dim rcvsignature As
    String = signverify.Split("^"c)(0)
200.                                             Dim mixmessage As
    String() = signverify.Split("^"c)
201.                                             Dim clmessage As String =
    mixmessage(mixmessage.Length - 1)
202.                                             Dim extractsign As String
    = Encryption.Signature.GetMd5Hash(hashsignature, clmessage)
203.                                             If rcvsignature =
    extractsign Then 'Validasi signature pesan
204.
    DisplayMessage(Color.DarkGreen, clmessage)
205.                                             Else
206.
    DisplayMessage(Color.DarkOrange, clmessage)
207.                                             End If
208.
209.                                         End If
210.                                         If Not cbMute.Checked Then
211.
    ChatForm.Play(Sound.ChatMessage)
212.                                         End If
213.                                         If cbFlash.Checked Then
214.                                             FlashWindow(Me.Handle)
215.                                         End If
216.
217.                                     Case MessageCodes.PAGE ' Terima
    ping dari client lain ============================
218.                                         If Not cbMute.Checked Then
219.
    ChatForm.Play(Sound.ChatPage)
220.                                         End If
221.
222.                                     Case MessageCodes.DISCONNECTED '
    Client lain telah menutup koneksinya =====================
223.                                         ContinueProcessingMessages =
    False
```

```
224.                                            lblStatus.Text = ""
225.                                            DisplayMessage(Color.Red,
    value)
226.                                            Me.Text = Me.Text & "
    [Terputus]"
227.                                            SetChatState(False)
228.                                            SendFinalDisconnectMessage =
    False
229.                                            If Not cbMute.Checked Then
230.
    ChatForm.Play(Sound.ChatClose)
231.                                            End If
232.
233.                                        Case MessageCodes.REQUESTSECURE
    'Pertukaran Kunci simetris AES
234.                                            MessageBox.Show("Anda diajak
    ke MODE AMAN",
235.                                                            "Pertukaran
    Kunci RSA", MessageBoxButtons.OK, MessageBoxIcon.Information)
236.                                            tbReceive.Text = value
237.
    SendSession(MessageCodes.KEYEXCHANGE, txtPublicKey.Text)
238.
239.                                        Case MessageCodes.KEYEXCHANGE
    'Terima Kunci Publik RSA dan enkripsi kunci AES dengan RSA Pengirim
240.                                            tbReceive.Text = value
241.                                            Try
242.                                                Dim EncryptedMessage As
    RSAResult
243.                                                EncryptedMessage =
    RSAEncryption.Encryptor(tbKeyAES.Text, tbReceive.Text)
244.
    SendChiperkey(MessageCodes.KEYSIMETRICPARAMETER,
    EncryptedMessage.AsBase64String)
245.                                            Catch ex As Exception
246.
    MessageBox.Show(ex.Message, "Error Chiperkey", MessageBoxButtons.OK,
    MessageBoxIcon.Error)
247.                                            End Try
248.
249.                                        Case
    MessageCodes.KEYSIMETRICPARAMETER
250.                                            Try
251.                                                Dim DecryptedMessage As
    RSAResult = RSAEncryption.Decrypt(Convert.FromBase64String(value),
    txtPrivateKey.Text)
252.                                                tbKeyAES.Text =
    DecryptedMessage.AsString
253.                                                cbEnkript.Checked = True
254.                                            Catch ex As Exception
255.
    MessageBox.Show(ex.Message, "Error Dechiperkey", MessageBoxButtons.OK,
    MessageBoxIcon.Error)
256.                                            End Try
257.                                            MessageBox.Show("Kunci AES
    berhasil di terima", "AES Received", MessageBoxButtons.OK,
    MessageBoxIcon.Warning)
258.                                            PictureBox1.Visible = True
259.
260.                                        Case MessageCodes.MODE
```

```
                                            MessageBox.Show("Teman Anda sudah keluar dari mode aman", "Mode Berubah", MessageBoxButtons.OK, MessageBoxIcon.Warning)
                                            PictureBox1.Visible = False
                                            cbEnkript.Checked = False


                                        End Select
                                    Catch ex As Exception
                                    End Try
                                End If
                            End If
                        End Select
                    End Sub

        'Fungsi Kirim pesan ==========================================================================
                    Private Function SendMessage(ByVal Code As MessageCodes, ByVal Value As String) As Boolean
                        Try ' Async Write so we don't lock up the GUI in the event of dropped connections
                            Dim msg() As Byte = Encoding.ASCII.GetBytes(Code.ToString & FieldMarker & Value & MessageMarker)
                            ChatClient.GetStream.BeginWrite(msg, 0, msg.Length, AddressOf MyWriteCallBack, ChatClient.GetStream)
                            Return True
                        Catch ex As Exception
                            SendFinalDisconnectMessage = False
                        End Try
                        Return False
                    End Function

                    'Fungsi Kirim Session ==========================================================================
                    Private Function SendSession(ByVal Code As MessageCodes, ByVal Value As String) As Boolean
                        Try ' Async Write so we don't lock up the GUI in the event of dropped connections
                            Dim msg() As Byte = Encoding.ASCII.GetBytes(Code.ToString & FieldMarker & Value & MessageMarker)
                            ChatClient.GetStream.BeginWrite(msg, 0, msg.Length, AddressOf MyWriteCallBack, ChatClient.GetStream)
                            Return True
                        Catch ex As Exception
                            SendFinalDisconnectMessage = False
                        End Try
                        Return False
                    End Function

                    'Fungsi Kirim ChiperKey ==========================================================================
                    Private Function SendChiperkey(ByVal Code As MessageCodes, ByVal Value As String) As Boolean
                        Try ' Async Write so we don't lock up the GUI in the event of dropped connections
```

```
                    Dim msg() As Byte =
    Encoding.ASCII.GetBytes(Code.ToString & FieldMarker & Value &
    MessageMarker)
                    ChatClient.GetStream.BeginWrite(msg, 0, msg.Length,
    AddressOf MyWriteCallBack, ChatClient.GetStream)
                    Return True
                Catch ex As Exception
                    SendFinalDisconnectMessage = False
                End Try
                Return False
            End Function

            Public Sub MyWriteCallBack(ByVal ar As IAsyncResult)
                Try
                    CType(ar.AsyncState, NetworkStream).EndWrite(ar)
                Catch ex As Exception
                End Try
            End Sub

            Private Sub AckTimer_Tick(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles AckTimer.Tick
                SendMessage(MessageCodes.ACK, "Ack")
            End Sub

            'Keterangan status lawan chat
            Private Sub tbMessageToSend_TextChanged(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
    tbMessageToSend.TextChanged
                If tbMessageToSend.TextLength > 0 Then
                    SendMessage(MessageCodes.TYPING, "Menulis...")
                Else
                    SendMessage(MessageCodes.TYPINGCANCEL, "Clear")
                End If
            End Sub


            'Session Start
    ================================================================================
            Private Sub btSession_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles btSession.Click
                If tbKeyAES.Text = "" Then
                    MsgBox("Isilah lebih dulu AES Key", vbCritical,
    "Salah")
                Else
                    cbEnkript.Checked = True
                    PictureBox1.Visible = True
                    If tbKeyAES.Text.Trim <> "" Then
                        If SendSession(MessageCodes.REQUESTSECURE,
    txtPublicKey.Text) Then
                            DisplayMessage(Color.Black, "Permintaan
    pertukaran kunci dikirim . . . . ")

                        Else
                            SetChatState(False)
                            Me.Text = Me.Text & " [Koneksi Putus]"
                        End If
                    End If
                End If
            End Sub

```

```
350.
351.           Private Sub btnSendMessage_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles btnSendMessage.Click
352.               passPhrase = tbKeyAES.Text
353.               input = tbMessageToSend.Text 'tambahkan dulu signed data
354.               Dim [signature] As String = input
355.               Dim hash As String =
    Encryption.Signature.GetMd5Hash(hashsignature, signature)
356.               Dim messagewithsignature As String = hash + "^" + input
357.               'Proses Enkripsi dan dekripsi pesan
    ================================================
358.               encrypt = RijndaelSimple.Encrypt(messagewithsignature,
    passPhrase, saltValue, hashAlgorithm,
359.                                                passwordIterations,
    initVector, keySize)
360.               If (cbEnkript.Checked = True) Then
361.                   If tbMessageToSend.Text.Trim <> "" Then
362.                       If SendMessage(MessageCodes.TEXT, encrypt) Then
363.                           DisplayMessage(Color.Black, input)
364.                           tbMessageToSend.Clear()
365.                       Else
366.                           SetChatState(False)
367.                           Me.Text = Me.Text & " [Koneksi Putus]"
368.                       End If
369.                   End If
370.               Else
371.                   If tbMessageToSend.Text.Trim <> "" Then
372.                       If SendMessage(MessageCodes.TEXT,
    tbMessageToSend.Text) Then
373.                           DisplayMessage(Color.Black,
    tbMessageToSend.Text)
374.                           tbMessageToSend.Clear()
375.                       Else
376.                           SetChatState(False)
377.                           Me.Text = Me.Text & " [Koneksi Putus]"
378.                       End If
379.                   End If
380.               End If
381.           End Sub
382.
383.
384.           Private Sub DisplayMessage(ByVal clr As Color, ByVal msg As
    String)
385.               Try
386.                   Dim SelStart As Integer =
    tbMessageToSend.SelectionStart
387.                   Dim SelLength As Integer =
    tbMessageToSend.SelectionLength
388.
389.                   rtbConversation.SelectionStart =
    rtbConversation.TextLength
390.                   rtbConversation.SelectionColor = clr
391.                   rtbConversation.SelectedText = "[" &
    DateTime.Now.ToShortTimeString & "] " & msg & vbCrLf
392.                   rtbConversation.Focus()
393.                   rtbConversation.ScrollToCaret()
394.
395.                   tbMessageToSend.Focus()
396.                   tbMessageToSend.SelectionStart = SelStart
397.                   tbMessageToSend.SelectionLength = SelLength
398.               Catch ex As Exception
```

```
399.                End Try
400.            End Sub
401.
402.            Private Sub rtbConversation_LinkClicked(ByVal sender As
    System.Object, ByVal e As System.Windows.Forms.LinkClickedEventArgs)
    Handles rtbConversation.LinkClicked
403.                Try
404.                    Process.Start(e.LinkText) ' Attemp to Open URL in the
    default browser
405.                Catch ex As Exception
406.                    MessageBox.Show("Takbisa membuka link: " &
    e.LinkText, "Tak bisa membuka link", MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation)
407.                End Try
408.            End Sub
409.
410.            Private Sub btnSendPage_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles btnSendPage.Click
411.                SendMessage(MessageCodes.PAGE, "Paging")
412.            End Sub
413.
414.            Private Sub ChatForm_FormClosing(ByVal sender As Object,
    ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles
    Me.FormClosing
415.                If SendFinalDisconnectMessage Then
416.                    SendMessage(MessageCodes.DISCONNECTED,
    Environment.MachineName & " Terputus")
417.                End If
418.                Try
419.                    If Not IsNothing(ChatClient) AndAlso
    ChatClient.Connected Then
420.                        ChatClient.Close()
421.                    End If
422.                Catch ex As Exception
423.                End Try
424.            End Sub
425.
426.            Private Shared Sub Play(ByVal ChatSound As Sound)
427.                Dim WaveName As String = ChatSound.ToString & ".wav"
428.                If Not Waves.ContainsKey(WaveName) Then
429.                    Waves.Add(WaveName, New EmbeddedWave(WaveName))
430.                End If
431.                If Waves(WaveName).IsValid Then
432.                    Waves(WaveName).Play()
433.                End If
434.            End Sub

435.        #Region "FlashWindowEx_API"
436.
437.            Public Const FLASHW_STOP As UInteger = 0
438.            Public Const FLASHW_CAPTION As Int32 = &H1
439.            Public Const FLASHW_TRAY As Int32 = &H2
440.            Public Const FLASHW_ALL As Int32 = (FLASHW_CAPTION Or
    FLASHW_TRAY)
441.            Public Const FLASHW_TIMERNOFG As Int32 = &HC
442.
443.            Public Structure FLASHWINFO
444.                Public cbsize As Int32
445.                Public hwnd As IntPtr
446.                Public dwFlags As Int32
447.                Public uCount As Int32
```

```
448.                Public dwTimeout As Int32
449.            End Structure
450.
451.            Public Declare Function FlashWindowEx Lib "user32.dll" (ByRef
    pfwi As FLASHWINFO) As Int32
452.
453.            Private Sub FlashWindow(ByVal handle As IntPtr)
454.                Dim flash As New FLASHWINFO
455.                flash.cbsize =
    System.Runtime.InteropServices.Marshal.SizeOf(flash)
456.                flash.hwnd = handle
457.                flash.dwFlags = FLASHW_ALL Or FLASHW_TIMERNOFG
458.                FlashWindowEx(flash)
459.            End Sub
460.
461.        #End Region
462.
463.        #Region "Class EmbeddeWave()"
464.
465.            Public Class EmbeddedWave
466.
467.                Private _WaveBytes() As Byte, _WaveLoaded As Boolean =
    False, _WaveName As String = ""
468.
469.                Public ReadOnly Property IsValid() As Boolean
470.                    Get
471.                        Return _WaveLoaded
472.                    End Get
473.                End Property
474.
475.                Public ReadOnly Property Name() As String
476.                    Get
477.                        Return _WaveName
478.                    End Get
479.                End Property
480.
481.                Private Sub New()
482.                End Sub
483.
484.                Public Sub New(ByVal EmbeddedWaveName As String)
485.                    _WaveName = EmbeddedWaveName
486.                    _WaveLoaded = LoadEmbeddedWave()
487.                End Sub
488.
489.                Private Function LoadEmbeddedWave() As Boolean
490.                    Dim resourceStream As System.IO.Stream =
491.
    System.Reflection.Assembly.GetExecutingAssembly().
492.                        GetManifestResourceStream(Me.GetType.Namespace &
    "." & _WaveName)
493.                    If Not IsNothing(resourceStream) Then
494.                        Try
495.                            ReDim _WaveBytes(CInt(resourceStream.Length))
496.                            resourceStream.Read(_WaveBytes, 0,
    CInt(resourceStream.Length))
497.                            Return True
498.                        Catch ex As Exception
499.                            MessageBox.Show(ex.Message, "Load Embedded
    Wave Resource Failed", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
500.                            Return False
501.                        End Try
```

```
                    Else
                        MessageBox.Show(_WaveName, "Embedded Wave
    Resource Not Found", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
                        Return False
                    End If
                End Function

                Public Sub Play()
                    If IsValid Then
                        My.Computer.Audio.Play(_WaveBytes,
    AudioPlayMode.Background)
                    End If
                End Sub

            End Class

            Private Sub PictureBox1_Click(sender As Object, e As
    EventArgs) Handles PictureBox1.Click
                Dim mode As Boolean
                If Not mode = cbEnkript.CheckState = True Then
                    MessageBox.Show("Anda telah keluar dari mode aman",
    "UnSecure", MessageBoxButtons.OK, MessageBoxIcon.Information)
                    PictureBox1.Visible = False
                    cbEnkript.CheckState = False
                    SendMessage(MessageCodes.MODE, "Teman anda Telah
    Keluar dari mode aman")
                End If
            End Sub


        #End Region

        End Class
```

5. Encryption

```
Imports System
Imports System.IO
Imports System.Text
Imports System.Security.Cryptography

Module Encryption

Public Class RSAEncryption

Public Shared Function Encryptor(ByVal Data As String, ByVal PublicKey
    As String) As RSAResult
Try
Dim ByteConverter As New UnicodeEncoding()
Return Encryptor(ByteConverter.GetBytes(Data), PublicKey)
Catch ex As Exception
Throw New Exception("Encrypt(String) : " & ex.Message, ex)

End Try
End Function
```

```
15. Public Shared Function Encryptor(ByVal Data() As Byte, ByVal PublicKey
    As String) As RSAResult
16. Try
17. Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()
18. RSA.FromXmlString(PublicKey)
19. Return New RSAResult(RSAEncrypt(Data, RSA.ExportParameters(False),
    False))
20. Catch ex As Exception
21. Throw New Exception("Encrypt(Byte) : " & ex.Message, ex)
22. End Try
23. End Function

24. Public Shared Function Decrypt(ByVal Data() As Byte, ByVal Privatekey As
    String) As RSAResult
25. Try
26. Dim RSA As RSACryptoServiceProvider = New RSACryptoServiceProvider()
27. RSA.FromXmlString(Privatekey)
28. Dim Result As New RSAResult(RSADecrypt(Data, RSA.ExportParameters(True),
    False))
29. Return Result
30. Catch ex As Exception
31. Throw New Exception("Decrypt(): " & ex.Message, ex)
32. End Try
33. End Function

34. Private Shared Function RSAEncrypt(ByVal DataToEncrypt() As Byte, ByVal
    RSAKeyInfo As RSAParameters, ByVal DoOAEPPadding As Boolean) As Byte()
35. Try
36. Dim encryptedData() As Byte
37. Using RSA As New RSACryptoServiceProvider
38. RSA.ImportParameters(RSAKeyInfo)
39. encryptedData = RSA.Encrypt(DataToEncrypt, DoOAEPPadding)
40. End Using
41. Return encryptedData
42. Catch e As CryptographicException
43. Throw New Exception("RSAEncrypt(): " & e.Message, e)
44. End Try
45. End Function

46. Private Shared Function RSADecrypt(ByVal DataToDecrypt() As Byte, ByVal
    RSAKeyInfo As RSAParameters, ByVal DoOAEPPadding As Boolean) As Byte()
47. Try
48. Dim decryptedData() As Byte
49. Using RSA As New RSACryptoServiceProvider
50. RSA.ImportParameters(RSAKeyInfo)
51. decryptedData = RSA.Decrypt(DataToDecrypt, DoOAEPPadding)
52. End Using
53. Return decryptedData
54. Catch e As CryptographicException
55. Throw New Exception("RSADecrypt(): " & e.Message, e)
56. End Try
57. End Function
58. End Class

59. Public Class RSAResult
60. Private _Data() As Byte
61. Public Sub New(ByVal Data() As Byte)
62. _Data = Data
63. End Sub
64. Public ReadOnly Property AsBytes() As Byte()
65. Get
```

```
66. Return _Data
67. End Get
68. End Property
69. Public ReadOnly Property AsString() As String
70. Get
71. Dim ByteConverter As New UnicodeEncoding()
72. Return ByteConverter.GetString(_Data)
73. End Get
74. End Property
75. Public ReadOnly Property AsBase64String() As String
76. Get
77. Return Convert.ToBase64String(_Data)
78. End Get
79. End Property
80. End Class

81. Public Class Keypair
82. Private _Publickey As String = String.Empty
83. Private _Privatekey As String = String.Empty
84. Public Property Publickey() As String
85. Get
86. Return _Publickey
87. End Get
88. Set(ByVal value As String)
89. _Publickey = value
90. End Set
91. End Property
92. Public Property Privatekey() As String
93. Get
94. Return _Privatekey
95. End Get
96. Set(ByVal value As String)
97. _Privatekey = value
98. End Set
99. End Property
100.        Public Shared Function CreateNewKeys() As Keypair
101.        Try
102.        Using RSA As New RSACryptoServiceProvider
103.        Dim Keys As New Keypair
104.        Keys.Privatekey = RSA.ToXmlString(True)
105.        Keys.Publickey = RSA.ToXmlString(False)
106.        Return Keys
107.        End Using
108.        Catch ex As Exception
109.        Throw New Exception("Keypair.CreateNewKeys():" & ex.Message, ex)
110.        End Try
111.        End Function
112.        End Class

113.        Public Class Signature

114.        Shared Function GetMd5Hash(ByVal md5Hash As MD5, ByVal input As
    String) As String

115.        ' Convert the input string to a byte array and compute the hash.
116.        Dim data As Byte() =
    md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input))

117.        ' Create a new Stringbuilder to collect the bytes
118.        ' and create a string.
119.        Dim sBuilder As New StringBuilder()
```

```
120.        ' Loop through each byte of the hashed data
121.        ' and format each one as a hexadecimal string.
122.        Dim i As Integer
123.        For i = 0 To data.Length - 1
124.        sBuilder.Append(data(i).ToString("x2"))
125.        Next i

126.        ' Return the hexadecimal string.
127.        Return sBuilder.ToString()

128.        End Function 'GetMd5Hash

129.        Shared Function VerifyMd5Hash(ByVal md5Hash As MD5, ByVal input
    As String, ByVal hash As String) As Boolean
130.        ' Hash the input.
131.        Dim hashOfInput As String = GetMd5Hash(md5Hash, input)

132.        ' Create a StringComparer an compare the hashes.
133.        Dim comparer As StringComparer = StringComparer.OrdinalIgnoreCase

134.        If 0 = comparer.Compare(hashOfInput, hash) Then
135.        Return True
136.        Else
137.        Return False
138.        End If

139.        End Function 'VerifyMd5Hash
140.        End Class

141.        Public Class RijndaelSimple

142.        Public Shared Function Encrypt _
143.        (
144.        ByVal plainText As String,
145.        ByVal passPhrase As String,
146.        ByVal saltValue As String,
147.        ByVal hashAlgorithm As String,
148.        ByVal passwordIterations As Integer,
149.        ByVal initVector As String,
150.        ByVal keySize As Integer
151.        ) _
152.        As String

153.        ' Convert strings into byte arrays.
154.        ' Let us assume that strings only contain ASCII codes.
155.        ' If strings include Unicode characters, use Unicode, UTF7, or
    UTF8
156.        ' encoding.
157.        Dim initVectorBytes As Byte()
158.        initVectorBytes = Encoding.ASCII.GetBytes(initVector)

159.        Dim saltValueBytes As Byte()
160.        saltValueBytes = Encoding.ASCII.GetBytes(saltValue)

161.        ' Convert our plaintext into a byte array.
162.        ' Let us assume that plaintext contains UTF8-encoded characters.
163.        Dim plainTextBytes As Byte()
164.        plainTextBytes = Encoding.UTF8.GetBytes(plainText)
```

```
        ' First, we must create a password, from which the key will be
    derived.
        ' This password will be generated from the specified passphrase
    and
        ' salt value. The password will be created using the specified
    hash
        ' algorithm. Password creation can be done in several iterations.
        Dim password As PasswordDeriveBytes
        password = New PasswordDeriveBytes _
        (
        passPhrase,
        saltValueBytes,
        hashAlgorithm,
        passwordIterations
        )

        ' Use the password to generate pseudo-random bytes for the
    encryption
        ' key. Specify the size of the key in bytes (instead of bits).
        Dim keyBytes() As Byte
        keyBytes = password.GetBytes(keySize / 8)

        ' Create uninitialized Rijndael encryption object.
        Dim symmetricKey As RijndaelManaged
        symmetricKey = New RijndaelManaged()

        ' It is reasonable to set encryption mode to Cipher Block
    Chaining
        ' (CBC). Use default options for other symmetric key parameters.
        symmetricKey.Mode = CipherMode.CBC

        ' Generate encryptor from the existing key bytes and
    initialization
        ' vector. Key size will be defined based on the number of the key
        ' bytes.
        Dim encryptor As ICryptoTransform
        encryptor = symmetricKey.CreateEncryptor(keyBytes,
    initVectorBytes)

        ' Define memory stream which will be used to hold encrypted data.
        Dim memoryStream As MemoryStream
        memoryStream = New MemoryStream()

        ' Define cryptographic stream (always use Write mode for
    encryption).
        Dim cryptoStream As CryptoStream
        cryptoStream = New CryptoStream _
        (
        memoryStream,
        encryptor,
        CryptoStreamMode.Write
        )
        ' Start encrypting.
        cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length)

        ' Finish encrypting.
        cryptoStream.FlushFinalBlock()

        ' Convert our encrypted data from a memory stream into a byte
    array.
        Dim cipherTextBytes As Byte()
```

```
209.        cipherTextBytes = memoryStream.ToArray()

210.        ' Close both streams.
211.        memoryStream.Close()
212.        cryptoStream.Close()

213.        ' Convert encrypted data into a base64-encoded string.
214.        Dim cipherText As String
215.        cipherText = Convert.ToBase64String(cipherTextBytes)

216.        ' Return encrypted string.
217.        Encrypt = cipherText
218.        End Function
219.        Public Shared Function Decrypt _
220.                (
221.                    ByVal cipherText As String,
222.                    ByVal passPhrase As String,
223.                    ByVal saltValue As String,
224.                    ByVal hashAlgorithm As String,
225.                    ByVal passwordIterations As Integer,
226.                    ByVal initVector As String,
227.                    ByVal keySize As Integer
228.                ) _
229.                As String
230.
231.                    ' Convert strings defining encryption key
    characteristics into byte
232.                    ' arrays. Let us assume that strings only contain
    ASCII codes.
233.                    ' If strings include Unicode characters, use Unicode,
    UTF7, or UTF8
234.                    ' encoding.
235.                    Dim initVectorBytes As Byte()
236.                    initVectorBytes = Encoding.ASCII.GetBytes(initVector)
237.
238.                    Dim saltValueBytes As Byte()
239.                    saltValueBytes = Encoding.ASCII.GetBytes(saltValue)
240.
241.                    ' Convert our ciphertext into a byte array.
242.                    Dim cipherTextBytes As Byte()
243.                    cipherTextBytes =
    Convert.FromBase64String(cipherText)
244.
245.                    ' First, we must create a password, from which the
    key will be
246.                    ' derived. This password will be generated from the
    specified
247.                    ' passphrase and salt value. The password will be
    created using
248.                    ' the specified hash algorithm. Password creation can
    be done in
249.                    ' several iterations.
250.                    Dim password As PasswordDeriveBytes
251.                    password = New PasswordDeriveBytes _
252.                    (
253.                        passPhrase,
254.                        saltValueBytes,
255.                        hashAlgorithm,
256.                        passwordIterations
257.                    )
258.
```

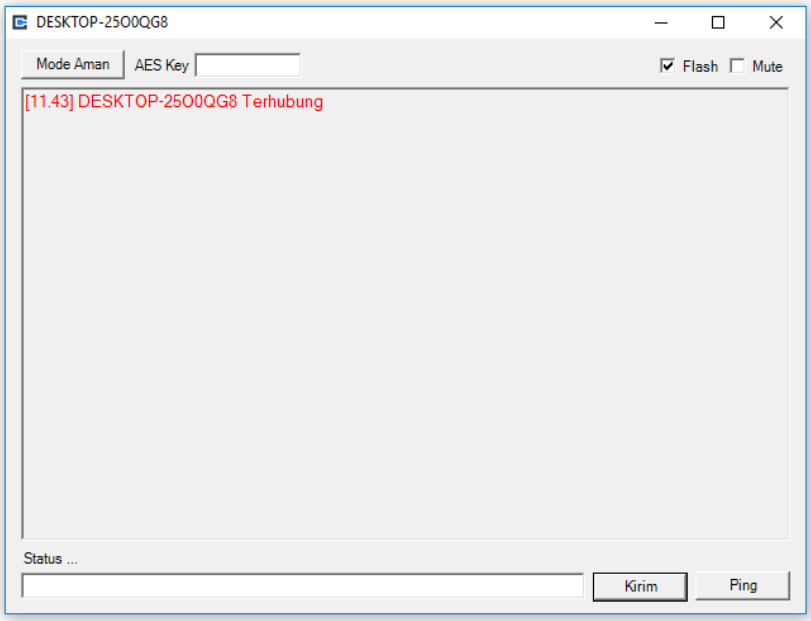
```
                    ' Use the password to generate pseudo-random bytes
    for the encryption
                    ' key. Specify the size of the key in bytes (instead
    of bits).
                    Dim keyBytes As Byte()
                    keyBytes = password.GetBytes(keySize / 8)

                    ' Create uninitialized Rijndael encryption object.
                    Dim symmetricKey As RijndaelManaged
                    symmetricKey = New RijndaelManaged()

                    ' It is reasonable to set encryption mode to Cipher
    Block Chaining
                    ' (CBC). Use default options for other symmetric key
    parameters.
                    symmetricKey.Mode = CipherMode.CBC

                    ' Generate decryptor from the existing key bytes and
    initialization
                    ' vector. Key size will be defined based on the
    number of the key
                    ' bytes.
                    Dim decryptor As ICryptoTransform
                    decryptor = symmetricKey.CreateDecryptor(keyBytes,
    initVectorBytes)

                    ' Define memory stream which will be used to hold
    encrypted data.
                    Dim memoryStream As MemoryStream
                    memoryStream = New MemoryStream(cipherTextBytes)

                    ' Define memory stream which will be used to hold
    encrypted data.
                    Dim cryptoStream As CryptoStream
                    cryptoStream = New CryptoStream _
                    (
                        memoryStream,
                        decryptor,
                        CryptoStreamMode.Read
                    )

                    ' Since at this point we don't know what the size of
    decrypted data
                    ' will be, allocate the buffer long enough to hold
    ciphertext;
                    ' plaintext is never longer than ciphertext.
                    Dim plainTextBytes As Byte()
                    ReDim plainTextBytes(cipherTextBytes.Length)

                    ' Start decrypting.
                    Dim decryptedByteCount As Integer
                    decryptedByteCount = cryptoStream.Read _
                    (
                        plainTextBytes,
                        0,
                        plainTextBytes.Length
                    )

                    ' Close both streams.
                    memoryStream.Close()
                    cryptoStream.Close()
```

```
309.
310.                     ' Convert decrypted data into a string.
311.                     ' Let us assume that the original plaintext string
    was UTF8-encoded.
312.                     Dim plainText As String
313.                     plainText = Encoding.UTF8.GetString _
314.                     (
315.                         plainTextBytes,
316.                         0,
317.                         decryptedByteCount
318.                     )
319.
320.                     ' Return decrypted string.
321.                     Decrypt = plainText
322.                 End Function
323.             End Class
324.
325.         End Module
```

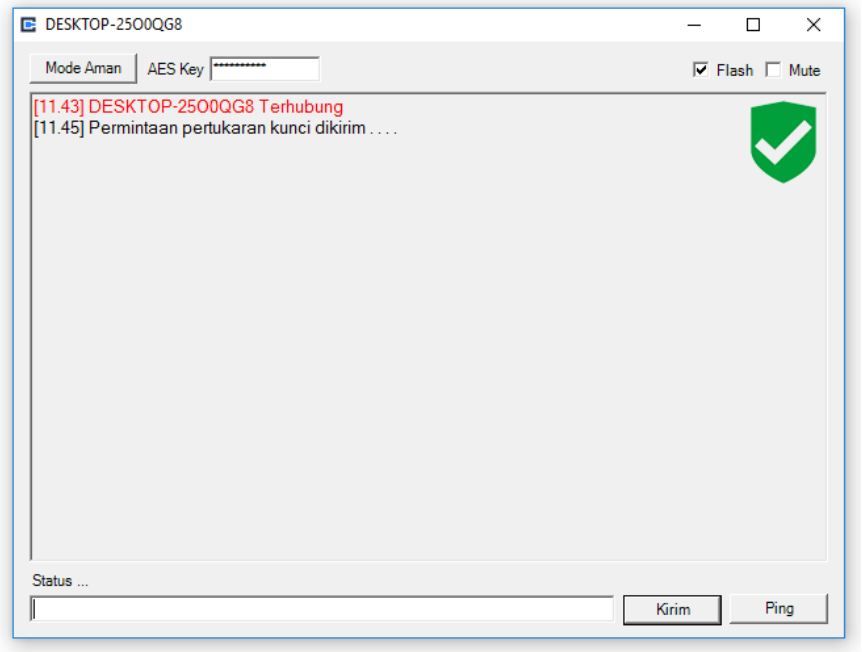
B. Bentuk Output

1. Form Inisialisasi IM LANChat

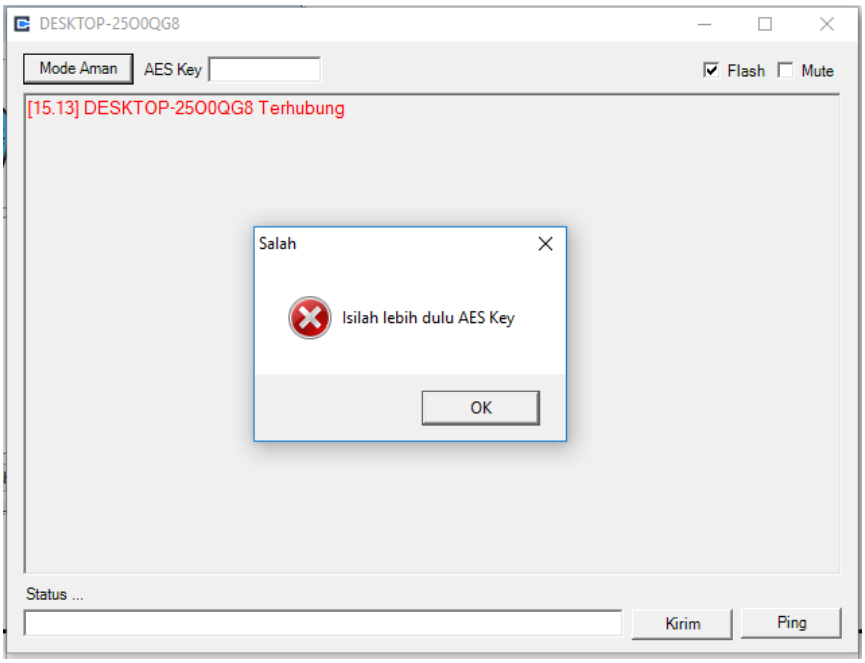
2. Form Chat Utama



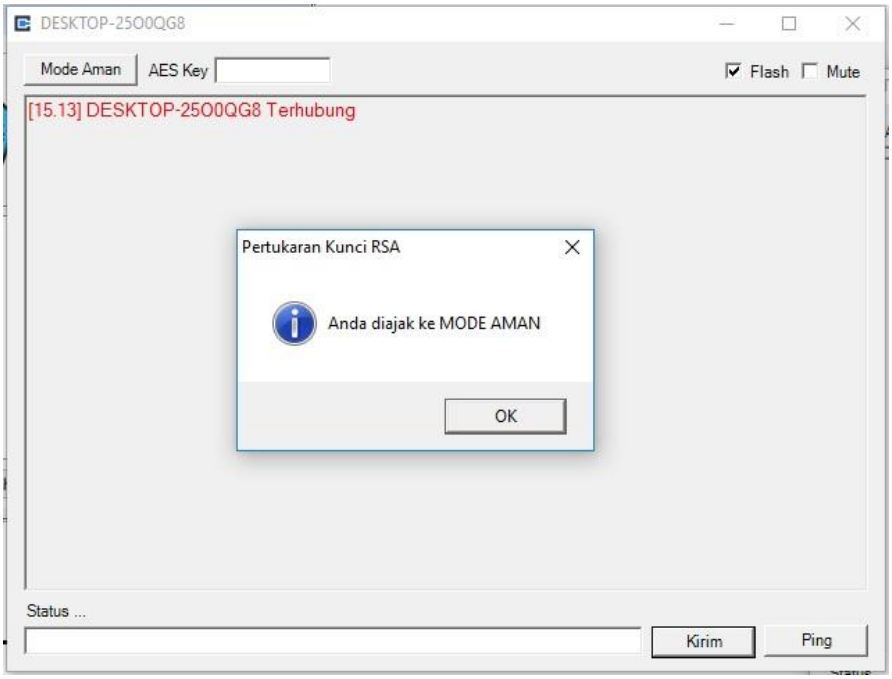
3. Form Chat Utama Mode Aman



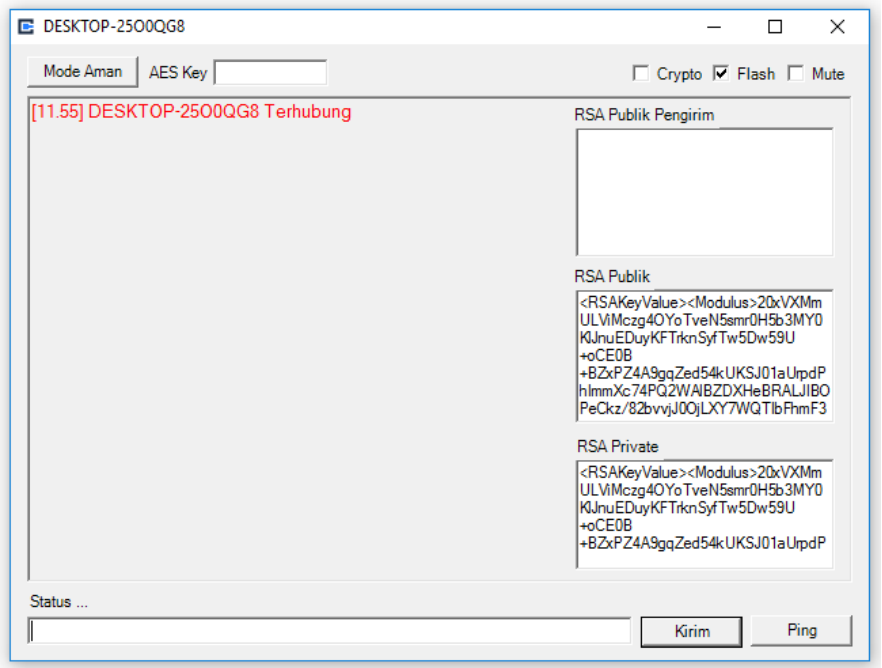
4. Pemberitahuan kesalahan ketika inputan AES belum terisi



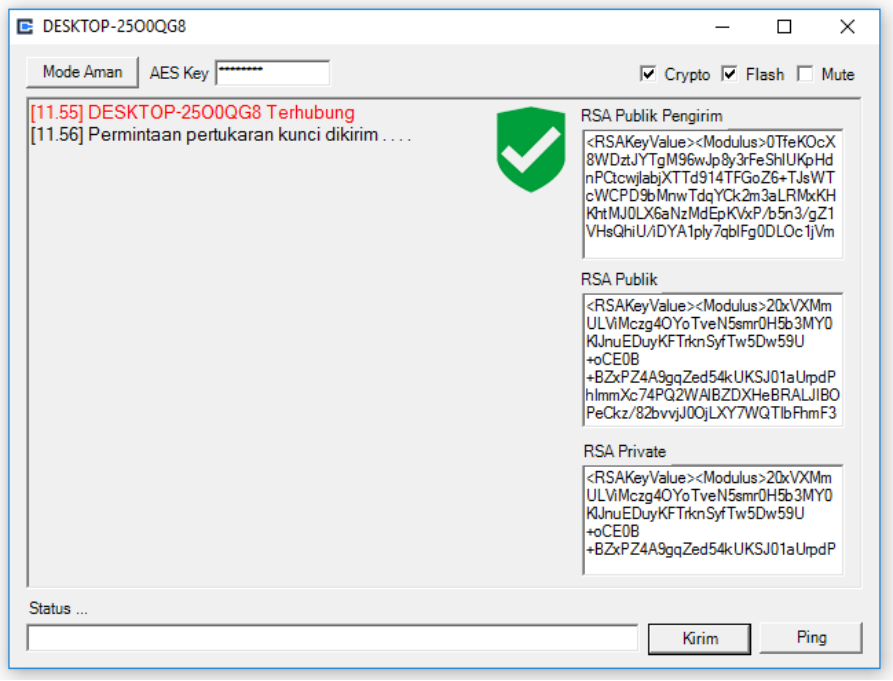
5. Pemberitahuan permintaan ke mode aman



6. Form Chat Utama Pengujian



7. Form chat utama mode aman pengujian



Riwayat Hidup

John Matrio Yahya dengan panggilan akrab Jojo, lahir di Gorontalo pada tanggal 24 Maret 1987. Merupakan anak ke dua dari dua bersaudara pasangan suami istri Bapak Marsel Yahya dan Ibu Zubaeda Sunge. Peneliti sekarang tinggal di Jl. Raja Wadi Pala, Perum Verlina Indah No.11 Kec. Telaga Kab. Gorontalo Prov. Gorontalo.

Pendidikan yang telah di tempuh peneliti yaitu SD Negri 1 Bongoime lulus tahun 1999, SMP Negeri 3 Kabila lulus tahun 2002, SMK Negeri 3 Gorontalo Jurusan Tehnik Elektronika Komunikasi tahun 2005, dan mulai tahun 2009 mengikuti Program S1 Tehnik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo sampai dengan sekarang.

Sampai dengan penulisan skripsi ini, peneliti masih terdaftar sebagai mahasiswa Program S1 Tehnik Informatika Fakultas Ilmu Komputer Universitas Ichsan Gorontalo.