SKRIPSI

APLIKASI KRIPTOGRAFI UNTUK ENKRIPSI PESAN MENGGUNAKAN METODE INTERLOCK PROTOCOL SEBAGAI PENCEGAHAN MAN-IN-THE-MIDDLE-ATTACK

Disusun Oleh :

FARID SUJIWO
07.12.629

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

2011

LEMBAR PERSETUJUAN

APLIKASI KRIPTOGRAFI UNTUK ENKRIPSI PESAN MENGGUNAKAN METODE INTERLOCK PROTOCOL SEBAGAI PENCEGAHAN MAN-IN-THE-MIDDLE-ATTACK

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Komputer Dan Informatika Strata Satu (S-1)

Disusun Oleh :

FARID SUJIWO
NIM : 07.12.629

Mengetahui

Ketua Jurusan Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800089

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
(Joseph Dedy Irawan, ST, MT)	(Sotyohadi, ST)
NIP. 19740416 200501 1 002	NIP.Y.1039700309

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2011

APLIKASI KRIPTOGRAFI UNTUK ENKRIPSI PESAN MENGGUNAKAN METODE INTERLOCK PROTOCOL SEBAGAI PENCEGAHAN MAN-IN-THE-MIDDLE-ATTACK

Farid Sujiwo

Jurusan Teknik Elektro S-1, Konsentrasi T.Komputer dan Informatika
Fakultas Teknologi Industri, Institut Teknologi Nasional Malang
Jln. Raya Karanglo Km 2 Malang
Xanderhan88@gmail.com

Dosen Pembimbing : **1. Joseph Dedy Irawan, ST, MT**
2. Sotyohadi, ST

Abstraksi

Perkembangan teknologi komunikasi pada jaringan kini semakin pesat dan berbagai macam cara yang dapat digunakan, salah satunya adalah dengan cara bertukar pesan atau yang sering disebut chatting. pada umumnya pada aplikasi chatting tidak disediakan pilihan untuk mengamankan pesan yang akan dikirimkan, oleh karena itu informasinya mudah sekali disadap dan diketahui oleh pengguna jaringan yang lain. Kriptografi merupakan ilmu yang mempelajari metode-metode yang berhubungan dengan aspek-aspek pada keamanan informasi misalnya kerahasiaan, integritas data, otentikasi pengirim penerima data, dan otentikasi data. sedangkan aplikasi kriptografi untuk enkripsi pesan dengan metode interock protocol dibuat untuk memberikan keamanan pada pesan yang dikirim pada percakapan yang terjadi antara kedua belah pihak (point to point) pada jaringan local (LAN), untuk menghindari pengintaian antara komunikasi yang sedang berlangsung (Man In The Middle Attack).

Aplikasi ini telah diuji dan berhasil menanggulangi problema Man-In-The-Middle-Attack, pesan yang dikirim tidak dapat dibaca dan dimanipulasi oleh orang lain. Panjang pesan yang dapat dienkripsi maksimal sebanyak 4095 karakter, dan nilai yang akan dibentuk sebagai kunci untuk proses enkripsi dan dekripsi harus bilangan prima.

Kata Kunci : kriptografi, chatting, keamanan, aplikasi

KATA PENGANTAR

Alhamdulillah, puji syukur kehadiratMu Ya Allah yang telah memberikan Rahmat dan HidayahNya, sehingga penulis dapat menyelesaikan skripsi dengan judul "**APLIKASI KRIPTOGRAFI UNTUK ENKRIPSI PESAN MENGGUNAKAN METODE INTERLOCK PROTOCOL SEBAGAI PENCEGAHAN MAN-IN-THE-MIDDLE-ATTACK**" dengan lancar. Skripsi merupakan persyaratan kelulusan Studi di Jurusan Teknik Elektro S-1 Konsentrasi Teknik Komputer dan Informatika ITN Malang dan untuk mencapai gelar Sarjana Teknik.

Keberhasilan penyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terima kasih kepada :

1. Bapak Ir. Soeparno Djiwo, MT selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. Sidik Noetjahjono, MT selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang
3. Bapak Ir. Yusuf Ismail Nakhoda, MT selaku Ketua Jurusan Teknik Elektro S-1,
4. Bapak Joseph Dedy Irawan, ST, MT selaku Dosen pembimbing I.
5. Bapak Sotyohadi, ST selaku Dosen Pembimbing II.
6. Ayah dan Ibu serta saudara-saudara yang selalu memberikan do'a, motivasi dan semangat.
7. Teman - teman Lab. PK&M yang selalu memberikan motivasi dan semangat.

Penulis telah berusaha semaksimal mungkin dan menyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelesaikan laporan ini. Untuk itu penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempurnaan laporan ini.

Harapan penulis semoga laporan skripsi ini memberikan manfaat bagi perkembangan ilmu pengetahuan dan pembaca.

Malang, Agustus 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam proses komunikasi data, walaupun data telah dienkripsi, terdapat kemungkinan data tersebut dapat diketahui oleh orang lain. Salah satu kemungkinan tersebut adalah orang tersebut menyadap media komunikasi yang digunakan oleh kedua orang yang sedang berkomunikasi tersebut. Hal inilah yang disebut dengan *man-in-the-middle-attack*. Dalam keadaan ini, orang yang menyadap berada di antara kedua orang yang sedang berkomunikasi. Data-data yang dikirimkan oleh orang yang sedang berkomunikasi satu sama lain selalu melalui orang yang menyadap tersebut, sehingga orang yang menyadap tersebut dapat mengetahui semua informasi yang dikirimkan satu sama lain. Keadaan ini muncul karena kedua orang yang sedang berkomunikasi tersebut tidak dapat mem-verifikasi status dari orang yang berkomunikasi dengannya tersebut, dengan mengambil asumsi bahwa proses penyadapan tersebut tidak menyebabkan gangguan dalam jaringan.

Problema *man-in-the-middle-attack* ini dapat diilustrasikan sebagai berikut, misalkan Alice dan Bob sedang berkomunikasi dan Mallory ingin menyadapnya. Ketika Alice mengirimkan kunci publiknya kepada Bob, Mallory dapat menangkap kunci ini dan mengirimkan kunci publiknya sendiri kepada Bob. Kemudian, ketika Bob mengirimkan kunci publiknya kepada Alice, Mallory juga dapat menangkap kunci tersebut dan mengirimkan kunci publiknya sendiri kepada Alice. Ketika Alice mengirimkan pesan kepada Bob yang dienkripsi dengan menggunakan kunci publik Bob, Mallory dapat menangkapnya. Karena pesan tersebut dienkripsi dengan menggunakan kunci publik Mallory, maka Mallory dapat mendekripsikan pesan tersebut dengan menggunakan kunci privatnya dan kemudian dienkripsi kembali dengan menggunakan kunci publik dari Bob

dan mengirimkannya kepada Bob. Hal yang sama juga terjadi ketika Bob mengirimkan pesan kepada Alice. Mallory dapat mengetahui semua pesan yang dikirimkan oleh Bob dan Alice tersebut. Problema *man-in-the-middle-attack* ini dapat dicegah dengan menggunakan *interlock protocol*. *Interlock protocol* ini diciptakan oleh Ron Rivest dan Adi Shamir. Algoritma inti dari protokol ini yaitu protokol ini mengirimkan 2 bagian pesan terenkripsi. Bagian pertama dapat berupa hasil dari fungsi hash satu arah (*one way hash function*) dari pesan tersebut dan bagian kedua berupa pesan terenkripsi itu sendiri. Hal ini menyebabkan orang yang menyadap tersebut tidak dapat mendekripsi pesan pertama dengan menggunakan kunci privatnya. Ia hanya dapat membuat sebuah pesan baru dan mengirimkannya kepada orang yang akan menerima pesan tersebut.

1.2. Rumusan Masalah

Berdasarkan latar belakang pemilihan judul, maka yang menjadi permasalahan adalah Bagaimana membangun aplikasi kriptografi dalam menanggulangi *Man-In-The-Middle-Attack* dengan menggunakan metode *interlock protocol*.

1.3. Tujuan

Tujuan penyusunan skripsi ini adalah :

Merancang suatu aplikasi kriptografi pertukaran pesan yang mampu untuk mencegah terjadinya *man-in-the-middle-attack* dengan menggunakan *interlock protocol*.

1.4. Batasan Masalah

Karena keterbatasan waktu dan pengetahuan penulis, maka ruang lingkup permasalahan dalam merancang aplikasi ini adalah sebagai berikut :

1. Proses kerja dari aplikasi ditampilkan dalam bentuk biner.
2. Proses enkripsi dan dekripsi pesan menggunakan algoritma RSA.
3. Fungsi hash satu arah yang digunakan adalah fungsi SHA-1.
4. Nilai-nilai yang dibutuhkan dalam algoritma RSA dan fungsi SHA-1 akan dihasilkan secara acak oleh komputer.
5. Algoritma-algoritma pendukung lainnya yang digunakan mencakup :
   a. Metode *Rabin-Miller* untuk menghasilkan bilangan prima.
   b. Algoritma GCD untuk mengecek prima relatif antar bilangan.
   c. Algoritma *Fast Exponentiation* untuk menghitung perpangkatan modulo bilangan besar.
   d. *Generator Geffe* untuk membangkitkan bilangan acak semu.
6. Aplikasi akan menampilkan proses enkripsi dan dekripsi dari pesan tersebut secara singkat.
7. *Interlock protocol* yang dibahas memiliki 2 alternatif berikut :
   a. Alternatif pertama, yaitu *message* terenkripsi dibagi menjadi 2 bagian yang sama besar.
   b. Alternatif kedua, yaitu pecahan pertama berupa nilai *hash* dari *one way hash function* dari *message* dan pecahan kedua berupa *message* terenkripsi.
8. Proses komunikasi yang terjadi hanya antara *Client* dan *Server*.
9. Tidak membahas tentang jaringan yang digunakan.
10. Pesan yang dikirimkan berupa teks.

1.5 Metodologi Penelitian

Adapun metode penelitian yang digunakan adalah sebagai berikut:

1. Studi literatur

   Pengumpulan data yang dilakukan dengan mencari bahan-bahan kepustakaan dan referensi dari berbagai sumber sebagai landasan teori yang ada hubunganya dengan permasalahan yang dijadikan objek penelitian.

2. Analisa Kebutuhan Sistim

   Data dan informasi yang telah diperoleh akan dianalisa agar didapatkan kerangka global yang bertujuan untuk mendefinisikan kebutuhan sistim di mana nantinya akan digunakan sebagai acuan perancangan sistim.

3. Perancangan dan Implementasi

   Berdasarkan data dan informasi yang telah diperoleh serta analisa kebutuhan untuk membangun sistim ini, akan dibuat rancangan kerangka global yang menggambarkan mekanisme dari sistim yang akan dibuat dan diimplementasikan kedalam sistim.

4. Eksperimen dan Evaluasi

   Pada tahap ini, sistim yang telah selesai dibuat akan diuji coba, yaitu pengujian berdasarkan fungsionalitas program, dan akan dilakukan koreksi dan penyempurnaan program jika diperlukan.

1.6 Sistimatika Penulisan

Untuk mempermudah dan memahami pembahasan penulisan skripsi ini, maka sistimatika penulisan disusun sebagai berikut :

Bab I : Pendahuluan

Berisi Latar Belakang, Rumusan Masalah, Tujuan Penelitian, Pembatasan Permasalahan, Metode Penelitian dan Sistimatika Penulisan.

Bab II : Tinjauan Pustaka

Berisi tentang landasan teori mengenai permasalahan yang berhubungan dengan penelitian yang dilakukan.

Bab III : Perancangan dan Analisa Sistim

Dalam bab ini berisi mengenai analisa kebutuhan sistim baik software maupun hardware yang diperlukan untuk membuat kerangka global yang menggambarkan mekanisme dari sistim yang akan dibuat. .

Bab IV : Pembuatan dan Pengujian Sistim

Berisi tentang implementasi dari perancangan sistim yang telah dibuat serta pengujian terhadap sistim tersebut.

Bab V : Penutup

Merupakan bab terakhir yang memuat intisari dari hasil pembahasan yang berisikan kesimpulan dan saran yang dapat digunakan sebagai pertimbangan untuk pengembangan penulisan selanjutnya.

BAB II
LANDASAN TEORI

2.1 Pengenalan Kriptografi

Cryptography adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek-aspek pada keamanan informasi misalnya kerahasiaan, integritas data, otentikasi pengirim / penerima data, dan otentikasi data.

Cryptography (kriptografi) berasal dari bahasa Yunani yaitu dari kata '*crypto*' dan '*graphia*' yang berarti penulisan rahasia. Kriptografi adalah suatu ilmu yang mempelajari penulisan secara rahasia. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut *cryptology*. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah.

Menurut Stalling, ada beberapa tuntutan yang terkait dengan isu keamanan data, yaitu:

1. *Confidentiality*. Menjamin bahwa data-data tersebut hanya bisa diakses oleh pihak-pihak tertentu saja.
2. *Authentication*. Baik pada saat mengirim atau menerima informasi, kedua belah pihak perlu mengetahui bahwa pengirim dari pesan tersebut adalah orang yang sebenarnya seperti yang diklaim.
3. *Integrity*. Tuntutan ini berhubungan dengan jaminan setiap pesan yang dikirim pasti sampai pada penerimanya tanpa ada bagian dari pesan tersebut yang diganti, diduplikasi, dirusak, diubah urutannya, dan ditambahkan.
4. *Nonrepudiation*. Mencegah pengirim maupun penerima mengingkari bahwa mereka telah mengirimkan atau menerima suatu pesan/informasi. Jika sebuah pesan dikirim, penerima dapat membuktikan bahwa pesan tersebut memang dikirim oleh pengirim yang tertera. Sebaliknya, jika sebuah pesan diterima, pengirim dapat membuktikan bahwa pesannya telah diterima oleh pihak yang ditujunya.

5. *Access Control*. Membatasi sumber-sumber data hanya kepada orang-orang tertentu.
6. *Availability*. Jika diperlukan setiap saat semua informasi pada sistem komputer harus tersedia bagi semua pihak yang berhak atas informasi tersebut.

Dari keenam aspek keamanan data tersebut, empat diantaranya dapat diatasi dengan menggunakan *cryptography* yaitu *confidentiality, integrity, authentication,* dan *nonrepudiation.*

2.1.1 *Confidentiality*

Confidentiality & privacy terkait dengan kerahasiaan data atau informasi. Pada sistem *e-government* kerahasiaan data-data pribadi (privacy) sangat penting. Hal ini kurang mendapat perhatian di sistem *e-government* yang sudah ada. Bayangkan jika data pribadi anda, misalnya data KTP atau kartu keluarga, dapat diakses secara *online*. Maka setiap orang dapat melihat tempat dan tanggal lahir anda, alamat anda, dan data lainnya. Data ini dapat digunakan untuk melakukan penipuan dan pembobolan dengan mengaku-aku sebagai anda (atau keluarga anda).

Ancaman atau serangan terhadap kerahasiaan data ini dapat dilakukan dengan menggunakan penerobosan akses, penyadapan data (*sniffer, key logger*), *social engineering* (yaitu dengan menipu), dan melalui kebijakan yang tidak jelas (tidak ada).

Untuk itu kerahasiaan data ini perlu mendapat perhatian yang besar dalam implementasi sistem *e-government* di Indonesia. Proteksi terhadap data ini dapat dilakukan dengan menggunakan *firewall* (untuk membatasi akses), segmentasi jaringan (juga untuk membatasi akses), enkripsi (untuk menyandikan data sehingga tidak mudah disadap), serta kebijakan yang jelas mengenai kerahasiaan data tersebut.

Pengujian terhadap kerahasian data ini biasanya dilakukan secara berkala dengan berbagai metode.

Ada beberapa jenis informasi yang tersedia didalam sebuah jaringan komputer. Setiap data yang berbeda pasti mempunyai grup pengguna yang

berbeda pula dan data dapat dikelompokkan sehingga beberapa pembatasan kepada pengunaan data harus ditentukan. Pada umumnya data yang terdapat didalam suatu perusahaan bersifat rahasia dan tidak boleh diketahui oleh pihak ketiga yang bertujuan untuk menjaga rahasia perusahaan dan strategi perusahaan. *Backdoor*, sebagai contoh, melanggar kebijakan perusahaan dikarenakan menyediakan akses yang tidak diinginkan kedalam jaringan komputer perusahaan.

Kerahasiaan dapat ditingkatkan dan didalam beberapa kasus pengengkripsian data atau menggunakan VPN. Topik ini tidak akan, tetapi bagaimanapun juga, akan disertakan dalam tulisan ini. Kontrol akses adalah cara yang lazim digunakan untuk membatasi akses kedalam sebuah jaringan komputer. Sebuah cara yang mudah tetapi mampu untuk membatasi akses adalah dengan menggunakan kombinasi dari *username-dan-password* untuk proses otentifikasi pengguna dan memberikan akses kepada pengguna (*user*) yang telah dikenali. Didalam beberapa lingkungan kerja keamanan jaringan komputer, ini dibahas dan dipisahkan dalam konteks otentifikasi.

Dalam menjaga kerahasiaan data, kriptografi mentransformasikan data jelas (*plaintext*) ke dalam bentuk data sandi (*ciphertext*) yang tidak dapat dikenali. *Ciphertext* inilah yang kemudian dikirimkan oleh pengirim (*sender*) kepada penerima (*receiver*). Setelah sampai di penerima, *ciphertext* tersebut ditranformasikan kembali ke dalam bentuk *plaintext* agar dapat dikenali.

2.1.2 *Authentication*

Otentikasi merupakan identifikasi yang dilakukan oleh masing – masing pihak yang saling berkomunikasi, maksudnya beberapa pihak yang berkomunikasi harus mengidentifikasi satu sama lainnya. Informasi yang didapat oleh suatu pihak dari pihak lain harus diidentifikasi untuk memastikan keaslian dari informasi yang diterima. Identifikasi terhadap suatu informasi dapat berupa tanggal pembuatan informasi, isi informasi,

waktu kirim dan hal-hal lainnya yang berhubungan dengan informasi tersebut.

Aspek ini berhubungan dengan metode untuk menyatakan bahwa informasi betul-betul asli, orang yang mengakses atau memberikan informasi adalah betul-betul orang yang dimaksud, atau *server* yang kita hubungi adalah betul-betul *server* yang asli.

Masalah pertama, membuktikan keaslian dokumen, dapat dilakukan dengan teknologi *watermarking* dan *digital signature*. *Watermarking* juga dapat digunakan untuk menjaga "*intelectual property*", yaitu dengan menandai dokumen atau hasil karya dengan "tanda tangan" pembuat. Masalah kedua biasanya berhubungan dengan akses kontrol, yaitu berkaitan dengan pembatasan orang yang dapat mengakses informasi.

Dalam hal ini pengguna harus menunjukkan bukti bahwa memang dia adalah pengguna yang sah, misalnya dengan menggunakan *password* Aspek / servis dari *security biometric* (ciri-ciri khas orang), dan sejenisnya. Ada tiga hal yang dapat ditanyakan kepada orang untuk menguji siapa dia:

- *What you have* (misalnya kartu ATM)
- *What you know* (misalnya PIN atau *password*)
- *What you are* (misalnya sidik jari, *biometric*)

Penggunaan teknologi *smart card*, saat ini kelihatannya dapat meningkatkan keamanan aspek ini. Secara umum, proteksi *authentication* dapat menggunakan *digital certificates*. *Authentication* biasanya diarahkan kepada orang (pengguna), namun tidak pernah ditujukan kepada *server* atau mesin. Pernahkan kita bertanya bahwa mesin ATM yang sedang kita gunakan memang benar-benar milik bank yang bersangkutan? Bagaimana jika ada orang nakal yang membuat mesin seperti ATM sebuah bank dan meletakkannya di tempat umum? Dia dapat menyadap data-data (informasi yang ada di *magnetic strip*) dan PIN dari orang yang tertipu. Memang membuat mesin ATM palsu tidak mudah.

Tapi, bisa anda bayangkan betapa mudahnya membuat *web site* palsu yang menyamar sebagai *web site* sebuah bank yang memberikan layanan *Internet Banking*. (Ini yang terjadi dengan kasus klikBCA.com.)

2.1 3 *Integrity*

Aspek *integrity* (integritas) terkait dengan keutuhan data. Aspek ini menjamin bahwa data tidak bolch diubah (*tampered, altered, modifed*) tanpa ijin dari yang berhak. Acaman terhadap aspek integritas dilakukan dengan melalui penerobosan akses, pemalsuan (*spoofing*), virus yang mengubah atau menghapus data, dan *man in the middle attack* (yaitu penyerangan dengan memasukkan diri di tengah-tengah pengiriman data). Proteksi terhadap serangan ini dapat dilakukan dengan menggunakan *digital signature, digital certificate, message authentication code, hash function*, dan *checksum*. Pada prinsipnya mekanisme proteksi tersebut membuat kode sehingga perubahan satu bit pun akan mengubah kode.

Contoh permasalahan integritas dapat dilihat pada sistem perhitungan pemilihan umum tahun 2004 kemarin, dimana pada awal proses perhitungan masih terdapat data uji coba. Hal ini menimbulkan keraguan atas integritas dari data yang berada di dalamnya.

Jaringan komputer yang dapat diandalkan juga berdasar pada fakta bahwa data yang tersedia apa yang sudah seharusnya. Jaringan komputer mau tidak mau harus terlindungi dari serangan (*attacks*) yang dapat merubah data selama dalam proses persinggahan (transmit). *Man-in-the-Middle* merupakan jenis serangan yang dapat merubah integritas dari sebuah data yang mana penyerang (*attacker*) dapat membajak "*session*" atau memanipulasi data yang terkirim.

Didalam jaringan komputer yang aman, partisipan dari sebuah "transaksi" data harus yakin bahwa orang yang terlibat dalam komunikasi data dapat diandalkan dan dapat dipercaya. Keamanan dari sebuah komunikasi data sangat diperlukan pada sebuah tingkatan yang dipastikan data tidak berubah selama proses pengiriman dan penerimaan pada saat

komunikasi data. Ini tidak harus selalu berarti bahwa "*traffic*" perlu di enkripsi, tapi juga tidak tertutup kemungkinan serangan "*Man-in-the-Middle*" dapat terjadi. "*Man in the middle attack*" dimana seseorang menempatkan diri di tengah pembicaraan dan menyamar sebagai orang lain.

2.1.4 *Nonrepudiation*

Aspek ini menjaga agar seseorang tidak dapat menyangkal telah melakukan sebuah transaksi. Sebagai contoh, seseorang yang mengirimkan email untuk memesan barang tidak dapat menyangkal bahwa dia telah mengirimkan email tersebut. Aspek ini sangat penting dalam hal *electronic commerce*. Penggunaan *digital signature, certifiates*, dan teknologi kriptografi secara umum dapat menjaga aspek ini. Akan tetapi hal ini masih harus didukung oleh hukum sehingga status dari *digital signature* itu jelas legal.

Setiap tindakan yang dilakukan dalam sebuah sistem yang aman telah diawasi (*logged*), ini dapat berarti penggunaan alat (*tool*) untuk melakukan pengecekan sistem berfungsi sebagaimana seharusnya. "*Log*" juga tidak dapat dipisahkan dari bagian keamanan "*system*" yang dimana bila terjadi sebuah penyusupan atau serangan lain akan sangat membantu proses investigasi. "*Log*" dan catatan waktu, sebagai contoh, bagian penting dari bukti di pengadilan jika *cracker* tertangkap dan diadili. Untuk alasan ini maka "*nonrepudiation*" dianggap sebagai sebuah faktor penting didalam keamanan jaringan komputer yang berkompeten.

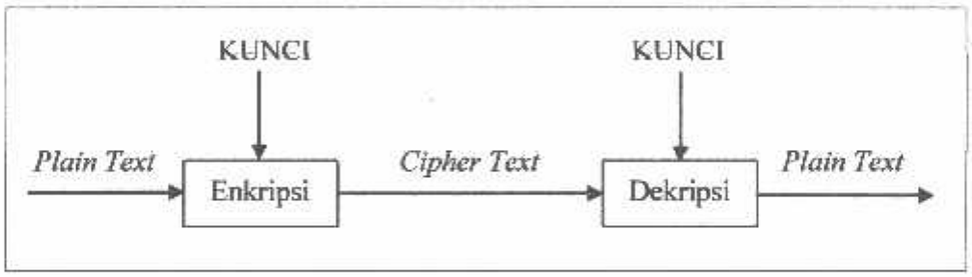
ITU-T telah mendefinisikan "*nonrepudition*" sebagai berikut :

1. Kemampuan untuk mencegah seorang pengirim untuk menyangkal kemudian bahwa dia telah mengirim pesan atau melakukan sebuah tindakan.
2. Proteksi dari penyangkalan oleh satu satu dari entitas yang terlibat didalam sebuah komunikasi yang turut serta secara keseluruhan atau sebagian dari komunikasi yang terjadi.

Proses tranformasi dari *plaintext* menjadi *ciphertext* disebut proses *encipherment* atau enkripsi (*encryption*), sedangkan proses mentransformasikan kembali *ciphertext* menjadi *plaintext* disebut proses dekripsi (*decryption*).

Untuk mengenkripsi dan mendekripsi data, kriptografi menggunakan suatu algoritma (*cipher*) dan kunci (key). *Cipher* adalah fungsi matematika yang digunakan untuk mengenkripsi dan mendekripsi. Sedangkan kunci merupakan sederetan bit yang diperlukan untuk mengenkripsi dan mendekripsi data.

Suatu pesan yang tidak disandikan disebut sebagai *plaintext* ataupun dapat disebut juga sebagai *cleartext*. Proses yang dilakukan untuk mengubah *plaintext* ke dalam *ciphertext* disebut *encryption* atau *encipherment*. Sedangkan proses untuk mengubah *ciphertext* kembali ke *plaintext* disebut *decryption* atau *decipherment*. Secara sederhana, istilah-istilah di atas dapat digambarkan sebagai berikut :



Gambar 2.1 Gambaran umum proses kriptografi

Cryptography adalah suatu ilmu ataupun seni mengamankan pesan, dan dilakukan oleh *cryptographer*. Sedang, *cryptanalysis* adalah suatu ilmu dan seni membuka (*breaking*) *ciphertext* dan orang yang melakukannya disebut *cryptanalyst*.

Cryptographic system atau *cryptosystem* adalah suatu fasilitas untuk mengkonversikan *plaintext* ke *ciphertext* dan sebaliknya. Dalam sistem ini, seperangkat parameter yang menentukan transformasi pencipheran tertentu disebut suatu set kunci. Proses enkripsi dan dekripsi diatur oleh satu atau beberapa kunci kriptografi. Secara umum, kunci-

kunci yang digunakan untuk proses pengenkripsian dan pendekripsian tidak perlu identik, tergantung pada sistem yang digunakan. Secara umum operasi enkripsi dan dekripsi dapat diterangkan secara matematis sebagai berikut :

$$EK(M) = C \quad \text{(Proses Enkripsi)}$$
$$DK(C) = M \quad \text{(Proses Dekripsi)}$$

Pada saat proses enkripsi kita menyandikan pesan M dengan suatu kunci K lalu dihasilkan pesan C. Sedangkan pada proses dekripsi, pesan C tersebut diuraikan dengan menggunakan kunci K sehingga dihasilkan pesan M yang sama seperti pesan sebelumnya.

Dengan demikian keamanan suatu pesan tergantung pada kunci ataupun kunci-kunci yang digunakan, dan tidak tergantung pada algoritma yang digunakan. Sehingga algoritma-algoritma yang digunakan tersebut dapat dipublikasikan dan dianalisis, serta produk-produk yang menggunakan algoritma tersebut dapat diproduksi massal. Tidaklah menjadi masalah apabila seseorang mengetahui algoritma yang kita gunakan. Selama ia tidak mengetahui kunci yang dipakai, ia tetap tidak dapat membaca pesan.

2.2 Algoritma Kriptografi

2.2.1 *Symmetric Algorithms*

Algoritma kriptografi simetris atau disebut juga algoritma kriptografi konvensional. Algoritma ini menggunakan kunci yang sama untuk proses enkripsi dan proses dekripsi.

Algoritma kriptografi simetris dibagi menjadi 2 kategori yaitu algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi pada satu bit atau satu *byte* data. Sedang pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan *bit* atau *byte* data (per blok). Contoh algoritma kunci simetris yang terkenal adalah DES (*Data Encryption Standard*).

Gambar 2.2 Prosedur kerja algoritma simetris

2.2.2 *Asymmetric Algorithms*

Algoritma kriptografi asimetris adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma kunci umum (*publik key algorithm*) karena kunci untuk enkripsi dibuat umum (*publik key*) atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang mengetahui data yang disandikan atau sering disebut kunci pribadi (*private key*). Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA dan ECC.

Syarat – syarat yang harus dipenuhi oleh suatu algoritma *publik-key* yaitu (Stalling, 1995):

1. Mudah secara komputasi bagi suatu pihak B untuk mengkonstruksi sepasang kunci asimetris (kunci publik KU, kunci pribadi KR).
2. Mudah secara komputasi bagi pengirim A, dengan memiliki kunci publik B dan pesan yang ingin dienkripsi, M, untuk menghasilkan *ciphertext* (C).
3. Mudah secara komputasi bagi penerima B untuk mendekripsi *ciphertext* yang dihasilkan dengan menggunakan kunci pribadinya untuk mengembalikan pesan aslinya.
4. Tidak bisa secara komputasi bagi pihak ketiga untuk memperoleh kunci pribadi KRb hanya dengan mengetahui kunci publik KUb.

5. Tidak bisa secara komputasi bagi pihak ketiga untuk mengembalikan data asli M hanya dengan mengetahui kunci publik KUb dan *ciphertext* C.

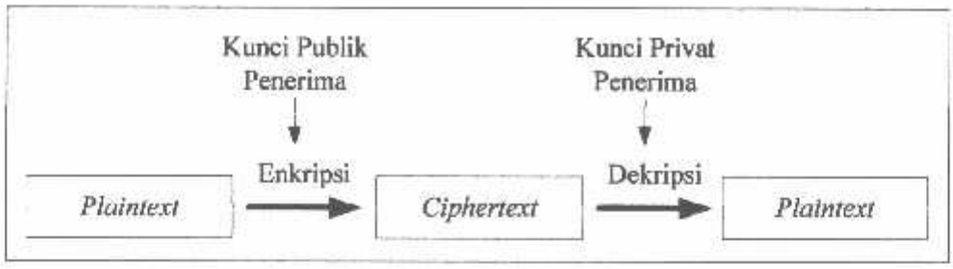
Walaupun bukanlah suatu keharusan bagi semua aplikasi publik-key, namun persyaratan keenam bisa ditambahkan :

6. Fungsi enkripsi dan dekripsi bisa diterapkan dengan urutan yang dibalik.

Kegunaan dari persyaratan keenam adalah untuk penerapan tanda tangan digital (*digital signature*) yang digunakan memecahkan isu otentikasi (*authentication*) dalam masalah keamanan data.

Menurut Stalling (Stalling, 1995), proses enkripsi publik-key sederhana melibatkan empat tahap berikut :

1. Setiap *user* di dalam jaringan membuat sepasang kunci untuk digunakan sebagai kunci enkripsi dan dekripsi dari pesan yang akan diterima.
2. *User* kemudian mempublikasikan kunci enkripsinya dengan menempatkan kunci publiknya ke tempat umum. Pasangan kunci yang lain tetap dijaga kerahasiaannya.
3. Jika *user* A ingin mengirimkan sebuah pesan ke *user* B, ia akan mengenkripsi pesan tersebut dengan menggunakan kunci publik *user* B.
4. Pada saat *user* B ingin mendeskripsikan pesan dari *user* A, ia akan menggunakan kunci pribadinya sendiri. Tidak ada pihak lain yang bisa mendekripsi pesan itu karena hanya B sendiri yang mengetahui kunci pribadi B.



Gambar 2.3 Prosedur kerja algoritma asimetris

2.3 Algoritma Rivest-Shamir-Adleman (RSA)

RSA merupakan salah satu teknik enkripsi dan dekripsi dengan menggunakan dua buah kunci. Kunci-kunci tersebut diperoleh dari hasil perhitungan eksponensial, perkalian, pembagian, penjumlahan dan pengurangan. Perhitungan dilakukan terhadap dua buah bilangan prima.

Walaupun RSA cenderung aman bukan berarti tidak bisa dilakukan "*attack*" terhadap enkripsinya. Didukung perkembangan *hardware* komputer yang semakin cepat maka semakin terbuka kemungkinan memecahkan enkripsi RSA. Pada tahun 1977 Rivest, Shamir dan Adleman mempublikasikan tantangan memecahkan enkripsi RSA yang memakai 129 digit bilangan bulat. Tantangan ini diharapkan bisa bertahan dari "*attack*" untuk waktu yang lama. Tetapi pada tahun 1994 tantangan ini dipecahkan dengan menggunakan komputer yang kekuatan komputasinya berimbang dengan komputer untuk membuat film animasi "Toy Story" (kumpulan, 87 unit komputer dual prosesor, 30 unit komputer empat procesor, 100Mhz SPARC*station*).

Dibawah ini diterangkan beberapa kemungkinan melakukan "*attack*" terhadap RSA:

1. Cara untuk memecahkan enkripsi RSA oleh penyerang adalah mencari kunci pribadi berdasarkan kunci publik. Jalan menuju itu adalah melakukan pemfaktoran bilangan **n** dari kunci publik. Kemudian dari **n** bisa didapatkan **p** dan **q**, bersama dengan **e** maka bisa didapatkan **d**. Masalahnya adalah memfaktorkan bilangan **n**. Apabila bilangan yang dipakai cukup besar maka akan memperkecil penyerangan dengan cara ini.

2. Cara lain adalah mencari cara untuk menghitung **akar pangkat e mod n**, dari persamaan **c=m^e**, **c** akar pangkat **e**, maka akan didapatkan **m**. Tetapi faktor dari **n** tidak bisa diketahui. Sampai sekarang belum ada metode yang diketahui untuk penyerangan yang dilakukan dengan cara ini.
3. Cara pemecahan enkripsi RSA lainnya adalah dengan menebak sebagian dari kata atau *Single Message Attack*, kemudian kata tersebut dienkripsi dengan menggunakan kunci publik dan membandingkan hasilnya dengan data asli yang terenkripsi. Cara ini memang bisa dilakukan, tetapi masih bisa ditangkal dengan menyusupi bit-bit *random* dalam pesan.
4. Cara "*attack*" yang paling baik yang dikenal adalah GNFS (*General Number Field Sieve*), caranya adalah memfaktorkan **n** ke bilangan prima **p** dan **q**, sama seperti ide no.1. Tetapi waktu yang dibutuhkan untuk RSA dengan besar kunci 1024 bit, sekitar 3x10^11 MIPS-year(MIPS-*year*, komputasi 1 juta instruksi perdetik selama setahun), atau dengan komputer 300 MIPS membutuhkan 2^30 tahun komputer. Kecepatan masih bisa ditingkatkan dengan *hardware* yang lebih baik.

Jadi banyak cara yang bisa dilakukan untuk memecahkan enkripsi RSA, walaupun ada cara yang belum dapat dilakukan saat ini. Semakin meningkatnya daya komputasi komputer dari waktu-kewaktu harus diakui semakin memperbesar kemungkinan memecahkan enkripsi RSA. Tetapi RSA masih bisa tetap digunakan karena digit bilangan bulat yang dipakai masih dapat diperbesar dan disesuaikan dengan teknologi yang ada sekarang. Semakin besar nilai kunci yang digunakan RSA, maka semakin aman juga teks yang terenkripsi tersebut.

Dengan demikian kekuatan dari algoritma RSA tergantung pada kesulitan pemfaktoran p dan q dari n (Menezes, 1996). Saat ini, tidak ada algoritma yang diketahui dapat memfaktorkan perkalian dari dua bilangan prima untuk nilai yang sangat besar (ratusan digit desimal) dengan cepat. Rekor tercepat dalam memfaktorkan perkalian dua bilangan prima dicatat pada tanggal 22 Agustus 1999 oleh sebuah tim yang dipimpin oleh Herman te Riele di Amsterdam. Bilangan yang berhasil difaktorkan adalah bilangan 512-bit (155-digit decimal) yang

merupakan perkalian dua bilangan prima 78-digit desimal. Total waktu yang diperlukan adalah 7,4 bulan dengan menggunakan algoritma *General Number Field Sieve*.

Menanggapi hal ini RSA *Laboratories* pada FAQ versi 4.1 menyarankan kunci RSA yang digunakan adalah minimal 1024-bit yang dengan menggunakan teknologi sekarang masih diperlukan waktu bertahun – tahun untuk memfaktorkannya.

Secara garis besar, algoritma kunci publik RSA dapat dijabarkan sebagai berikut:

Algoritma RSA

Key generation:

1. Hasilkan dua buah integer prima besar, p dan q. Untuk memperoleh tingkat keamanan yang tinggi pilih p dan q yang berukuran besar, misalnya 1024 bit.
2. Hitung m = (p-1)*(q-1).
3. Hitung n = p*q
4. Pilih e yg relatif prima terhadap m. e relatif prima thd m artinya faktor pembagi terbesar keduanya adalah 1, secara matematis disebut gcd(e,m) = 1. Untuk mencarinya dapat digunakan algoritma Euclid.
5. Cari d, sehingga e*d = 1 mod (m), atau $d = e^{-1} * \text{mod } (m)$. Untuk mencarinya, dapat digunakan algoritma extended Euclid.
6. Kunci publik : e, n
   Kunci private : d, n

Public key encryption & decryption

B mengenkripsi message M untuk A

Yg harus dilakukan B :
1. Ambil kunci publik A yg otentik (n, e)
2. Representasikan message sbg integer M dalam interval [0,n-1]
3. Hitung C = M ^ e (mod n)
4. Kirim C ke A

Untuk mendekripsi, A melakukan :
Gunakan kunci pribadi d untuk menghasilkan M = C ^ d (mod n)

Contoh Penerapan :

Misalkan :

Dalam contoh ini dipilih bilangan yg kecil agar memudahkan perhitungan, namun dalam aplikasi nyata pilih bilangan prima besar untuk meningkatkan keamanan.

p = 3

q = 11

n = 3 * 11 = 33

m = (3-1) * (11-1) = 20

e = 2 => gcd(e, 20) = 2

e = 3 => gcd(e, 20) = 1 (yes)

n = 0 => d = 1 / 3

n = 1 => d = 21 / 3 = 7 (yes)

Publik key : (3, 33)

Private key : (7, 33)

```
Let's check the math using numbers
---------------------------------
* Try encryption : message "2"

  C = 2 ^ 7 (mod 33)
    = 29

  Try to decrypt : ciphertext "29"
  M = 29 ^ 3 (mod 33)
    = 24389 (mod 33)
    = 2

** Encrypt : message " " (ASCII=20)
  C = 20 ^ 7 (mod 33)
    = 8000 (mod 33)
    = 26

  Decrypt : ciphertext 26
  M = 26 ^ 3 (mod 33)
    = 17576 (mod 33)
    = 20
```

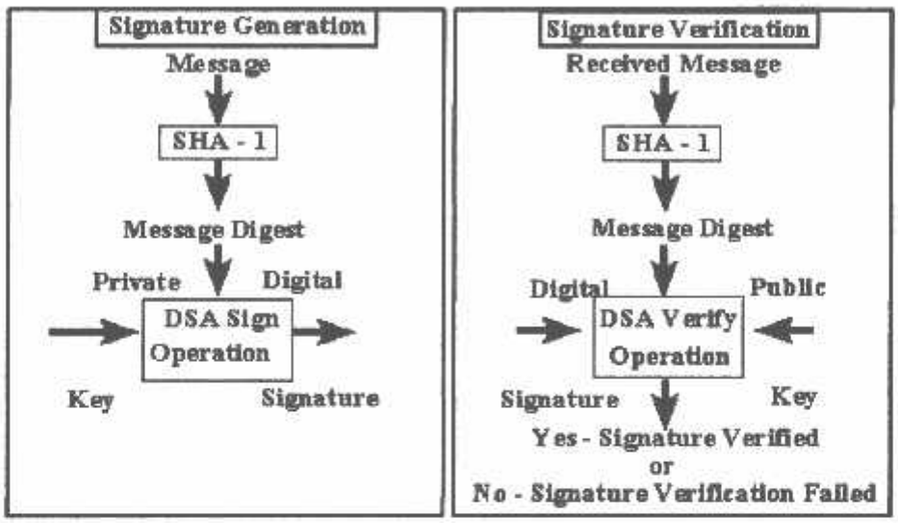
2.4 Fungsi *One-Way Hash* SHA-1

Secure Hash Algorithm, SHA-1 ini dikembangkan oleh NIST (*National Institute of Standard and Technology*). SHA-1 dapat diterapkan dalam penggunaan *Digital Signature Algorithm* (DSA) yang dispesifikasikan dalam *Digital Signature Standard* (DSS) dan SHA tersebut dapat diterapkan untuk aplikasi federal.

Untuk suatu pesan yang panjangnya < 2 ^ 64, SHA-1 akan menghasilkan keluaran sebanyak 160 bit dari pesan tersebut dan pesan keluaran itu disebut *message digest*. Panjang jarak *message digest* dapat berkisar antara 160 sampai

512 bit tergantung algoritmanya. Berdasarkan cirinya SHA-1 dapat digunakan dengan algoritma kriptografi lainnya seperti *Digital Signature Algorithms* atau dalam generasi angka yang acak (*bits*).

Bentuk penggunaan SHA-1 dengan DSA dapat dilihat pada gambar di bawah ini:



Gambar 2.4 Bentuk Penggunaan SHA-1 dengan DSA

SHA-1 dikatakan aman karena proses SHA-1 dihitung secara infisibel untuk mencari pesan yang sesuai untuk menghasilkan *message digest* atau dapat juga digunakan untuk mencari dua pesan yang berbeda yang akan menghasilkan *message digest* yang sama.

Menurut jenisnya SHA dapat dispesifikasikan menjadi 4 bagian yaitu: SHA-1, SHA-256, SHA-384, dan SHA-512. Berikut ini merupakan daftar-daftar properti dari keempat SHA.

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security[2] (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

Gambar 2.5 Daftar-daftar properti dari keempat SHA

Untuk SHA-1 ukuran blok pesan -m bit- dapat ditentukan tergantung dari algoritmanya. Pada SHA-1 masing-masing blok pesan mempunyai 512 bit dimana dapat dilakukan dengan 16 urutan sebesar 32 bit.

SHA-1 digunakan untuk menghitung *message digest* pada pesan atau *file* data yang diberikan sebagai *input*. Tujuan pengisian pesan adalah untuk menghasilkan total dari pesan yang diisi menjadi perkalian dari 512 bits.

Beberapa hal yang dilakukan dalam pengisian pesan :

a. Panjang dari pesan,M adalah k bits dimana panjang $k < 2^{64}$. Tambahkan bit "1" pada akhir pesan. Misalkan pesan yang asli adalah "01010000" maka setelah diisi menjadi "010100001".

b. Tambahkan bit "0", angka bit "0" tergantung dari panjang pesan. Misalnya :Pesan asli yang merupakan bit string : abcde

01100001 01100010 01100011 01100100 01100101.

Setelah langkah (a) dilakukan

01100001 01100010 01100011 01100100 0110010 1.

Panjang k = 40 dan angka bit di atas adalah 41dan 407 ditambah bit "0" (448 – (40+1) = 407). Kemudian diubah dalam hex:

```
61626364  65800000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000
```

c. Untuk memperoleh 2 kata dari k,angka bit dalam pesan asli yaitu jika $k < 2^{32}$ maka kata pertama adalah semua bit "0". Maka gambaran dari 2 kata dari k = 40 dalam hex adalah 00000000 00000028.

```
61626364  65800000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000028
```

SHA-1 menggunakan urutan fungsi logika yang dilambangkan dengan f_0, f_1,..., f_{79}. Untuk masing-masing f_t, dimana $0 \leq t < 79$ akan menghasilkan output sebanyak 32 bit.

Fungsinya adalah sebagai berikut:

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D) & 0 \leq t \leq 19 \\ B \oplus C \oplus D & 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & 40 \leq t \leq 59 \\ B \oplus C \oplus D & 60 \leq t \leq 79 \end{cases}$$

Konstanta kata yang digunakan pada SHA-1 yang disimbolkan secara berurutan dari K(0), K(1),..., K(79) dalam bentuk *hex* adalah sebagai berikut :

$$K_t = \begin{cases} 5A827999 & 0 \leq t \leq 19 \\ 6ED9EBA1 & 20 \leq t \leq 39 \\ 8F1BBCDC & 40 \leq t \leq 59 \\ CA62C1D6 & 60 \leq t \leq 79 \end{cases}$$

Algoritma SHA-1 dapat diringkas sebagai berikut:

a. Penghitungan menggunakan dua *buffer* dimana masing-masing *buffer* terdiri dari lima sebesar 32 bit kata dan urutan 80 juga sebesar 32 bit kata. Lima kata pertama pada *buffer* kata diberi nama A, B, C, D, E sedangkan lima kata kedua diberi nama H_0, H_1, H_2, H_3, dan H_4. Kemudian pada 80 kata yang berurutan diberi nama W_0, W_1, ..., W_{79} dan pada penghitungan ini juga memakai TEMP.
b. Lakukan pengisian pesan, M dan kemudian parsingkan pesan tersebut ke dalam N 512 bit blok pesan, $M^{(1)}$, $M^{(2)}$, ..., $M^{(n)}$. Caranya : 32 bit pertama dari blok pesan ditunjukkan ke $M_0^{(i)}$, lalu 32 bit berikutnya adalah $M_1^{(i)}$ dan selanjutnya berlaku hingga $M_{15}^{(i)}$.

c. Inisialisasi Nilai *Hash* (dalam bentuk hex) :

H_0 = 67452301 H_3 = 10325476

H_1 = EFCDAB89 H_4 = C3D2E1F0

H_2 = 98BADCFE

d. Lakukan proses M_1, M_2, ..., M_n dengan cara membagi M_i ke dalam 16 kata W_0, W_1, ..., W_{15} dimana W_0 merupakan *left most*.

e. Hitung : For t = 16 to 79

$$W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$$

f. Inisialisasi 5 variabel A, B, C, D, dan E dengan nilai *Hash* :

$$A = H_0 ; B = H_1; C = H_2; D = H_3; E = H_4.$$

g. Hitung : For t = 0 to 79

$$TEMP = S^5(A) + f_t(B,C,D) + E + Wt + Kt$$

$$E = D; D = C; C = S^{30}(B); B = A; A = TEMP.$$

h. Hitung Nilai *Hash* :

$H_0 = H_0 + A$; $H_1 = H_1 + B$;

$H_2 = H_2 + C$; $H_3 = H_3 + D$;

$H_4 = H_4 + E$.

Hasil dari *message digest* sebesar 160 bit dari pesan, M adalah : H_0 H_1 H_2 H_3 H_4.

2.5 Protokol Kriptografi

2.5.1 Pengertian Protokol

Suatu protokol adalah serangkaian langkah yang melibatkan dua pihak atau lebih dan dirancang untuk menyelesaikan suatu tugas. Dari definisi ini dapat diambil beberapa arti sebagai berikut :

a. protokol memiliki urutan dari awal hingga akhir;
b. setiap langkah harus dilaksanakan secara bergiliran;
c. suatu langkah tidak dapat dikerjakan bila langkah sebelumnya belum selesai;
d. diperlukan dua pihak atau lebih untuk melaksanakan protokol;
e. protokol harus mencapai suatu hasil;

Selain itu, suatu protokol pun memiliki karakteristik yang lain, yaitu :

a. setiap orang yang terlibat dalam protokol harus mengetahui terlebih dahulu mengenai protokol dan seluruh langkah yang akan dilaksanakan;
b. setiap orang yang terlibat dalam protokol harus menyetujui untuk mengikutinya;
c. protokol tidak boleh menimbulkan kerancuan;
d. protokol harus lengkap;

Cryptographic protocol adalah suatu protokol yang menggunakan kriptografi. Protokol ini melibatkan sejumlah algoritma kriptografi, namun secara umum tujuan protokol lebih dari sekedar kerahasiaan. Pihak-pihak yang berpartisipasi mungkin saja ingin membagi sebagian rahasianya untuk menghitung sebuah nilai, menghasilkan urutan random, atau pun menandatangani kontrak secara bersamaan. Penggunaan kriptografi dalam sebuah protokol terutama ditujukan untuk mencegah atau pun mendeteksi adanya *eavesdropping* dan *cheating*.

2.5.2 Fungsi Protokol

Dalam kehidupan kita sehari-hari terdapat banyak sekali protokol tidak resmi, misalnya saja dalam permainan kartu, pemungutan suara dalam pemilihan umum. Akan tetapi tidak ada seorang pun yang memikirkan mengenai protokol-protokol ini, protokol-protokol ini terus berkembang, semua orang mengetahui bagaimana menggunakannya.

Saat ini, semakin banyak interaksi antar manusia dilakukan melalui jaringan komputer. Komputer ini tentu saja memerlukan suatu protokol formal agar dapat melakukan hal yang biasa dilakukan manusia tanpa berpikir. Bila kita berpindah dari satu daerah ke daerah lain dan mengetahui bahwa kartu pemilihan suaranya berbeda dengan yang biasa kita gunakan, kita dapat beradaptasi dengan mudah. Akan tetapi kemampuan ini belum dimiliki oleh komputer, sehingga diperlukan suatu protokol.

Protokol digunakan untuk mengabtraksikan proses penyelesaian suatu tugas dari mekanisme yang digunakan. Protokol komunikasi adalah sama meskipun diimplementasikan pada PC atau VAX. Bila kita yakin bahwa kita memiliki protokol yang baik, kita dapat mengimplementasikannya dalam segala benda mulai dari telepon hingga pemanggang roti cerdas.

2.5.3 Beberapa Protokol Kriptografi

2.5.3.1 Protokol Pembagian Rahasia

Jika Beni memiliki rahasia, ia dapat memberikan 'separuh' rahasia itu kepada Huda dan 'separuh' rahasia itu kepada Sofyan. Huda, yang menerima paruh pertama rahasia Beni, tidak bisa mengetahui apa isi rahasia itu. Demikian pula dengan Sofyan. Namun, jika Huda dan Sofyan menggabungkan potongan-potongan rahasia itu, maka akan tergambar rahasia Beni. Pembagian rahasia *(secret splitting)* dapat dilakukan dengan cara:

1. Beni membuat seuntai string acak R yang panjangnya sama dengan pesan rahasia M.
2. Beni melakukan operasi XOR antara M dengan R, sehingga menghasilkan S.
3. Beni memberikan R kepada Huda dan S kepada Sofyan
4. Jika Huda dengan Sofyan bertemu, maka mereka sanggup mendapatkan pesan rahasia M dengan cara melakukan operasi XOR antara S dengan R.

2.5.3.2 Protokol Komitmen-Bit

Protokol ini bermanfaat kalau misalnya Beni hendak membuat suatu pernyataan atau komitmen (katakanlah suatu string binari 1000), namun Beni tak ingin agar Huda mengetahui isi pernyatan tersebut sebelum saatnya. Huda harus merasa yakin bahwa Beni pada saatnya nanti, benar-benar mengeluarkan isi

pernyataan yang sebenarnya saat melakukan komitmen, dan tidak mengeluarkan pernyataan yang sudah diubah (misalnya mengubah string tadi menjadi 1001). Ada beberapa jenis protokol komitmen-bit, namun di bawah ini hanya dijelaskan salah satu diantaranya, yakni dengan fungsi *hash* satu arah:

1. Beni membuat dua buah string secara acak, yakni R_1 dan R_2
2. Beni menggabungkan kedua string acak itu ke dalam pernyataannya (b) yang akan dikomitmenkan menjadi (R_1, R_2, b)
3. Beni menghitung *hash* dari gabungan string itu, $Hash(R_1, R_2, b)$.
4. Beni kemudian mengirimkan *hash* tersebut beserta R_1 kepada Huda. Huda akan menyimpannya untuk pemeriksaan nanti.
5. Jika sudah tiba saatnya untuk menunjukkan pernyataannya, Beni memberikan seluruh string (R_1, R_2, b) kepada Huda.
6. Huda memeriksa fungsi *hash* dari (R_1, R_2, b). Jika cocok dengan *hash* yang diperiksanya dulu, maka pernyataan Beni tidak diubah.

2.5.3.3 Tanda Tangan Buta

Huda disodori 100 amplop tertutup oleh Beni. Amplop itu berisi secarik pesan dan kertas karbon. Huda membuka 99 amplop secara acak. Jika seluruh amplop yang dibuka ternyata berisi pesan yang mirip, maka Huda dapat merasa bahwa amplop ke-100 juga berisi pesan yang mirip pula. Namun, jika satu saja dari 99 amplop tadi ada yang isi berbeda dari yang lain, maka Huda dapat mencurigai bahwa isi amplop ke-100 bisa saja juga tidak mirip dengan isi ke-98 amplop lainnya.

Dalam kasus dimana ternyata ke-99 amplop yang dibuka secara acak tadi berisi pesan yang mirip, maka dengan keyakinan yang cukup tinggi Huda berani menandatangani amplop terakhir

yang belum dibuka. Tanda tangan Huda akan menembus amplop dan kertas karbon, sehingga pesan dalam amplop akan tertandatangani oleh Huda. Huda kurang lebih tahu apa isi pesan di amplop ke-100 itu. Protokol tanda tangan buta (*blind signature*) bekerja sebagai berikut:

1. Beni 'mengalikan' dokumen (yang akan ditandatangani) dengan sebuah faktor pembuta.
2. Beni mengirimkan dokumen itu kepada Huda
3. Huda menandatangani dokumen itu
4. Huda mengembalikan dokumen yang sudah ditandatangani tadi kepada Beni
5. Beni membaginya dengan faktor pembuta, sehingga mendapatkan dokumen yang asli sudah tertandatangani oleh Huda.

2.5.3.4 Protokol Uang Digital

David Chaum, memiliki beberapa paten atas protokol uang digital yang diciptakannya. Berikut ini dijelaskan salah satu protokol uang digital:

1. Beni menyiapkan n lembar uang dengan nilai tertentu. Setiap uang diberi nomor seri acak X yang cukup panjang, sehingga kemungkinan 2 bilangan acak sama kecil sekali. Dalam setiap uang juga ada n (I_1, I_2, ..., I_n) string identifikasi yang berguna untuk memberikan informasi mengenai pemilik uang, yakni Beni. Beni kemudian memecah tiap-tiap string identitas diri itu tadi menjadi dua bagian dengan menggunakan protokol pemecahan rahasia. Lantas Beni melakukan bit-komitmen pada setiap pecahan. Contoh uang yang disiapkan adalah:

   Nilai: Rp.1.000,-

   Nomor seri acak: X

   String identitas: $I_1 = (I_{1L}, I_{1R})$

$I_2 = (I_{2L}, I_{2R})$

....

$I_n = (I_{nL}, I_{nR})$

2. Beni memasukkan uang itu kedalam yang juga disisipi kertas karbon amplop (mengalikan uang dengan faktor pembuta), lalu memberikannya kepada bank.
3. Bank akan meminta Beni untuk membuka n - 1 amplop itu secara acak. Bank memeriksa apakah semua uang tersebut memiliki nilai yang sama. Bank juga meminta kepada Beni untuk membuktikan kejujuran dirinya saat menuliskan string identifikasi pada uang itu, dengan cara menggabungkan pasangan-pasangan string identifikasi.
4. Jika bank merasa bahwa Beni tidak melakukan kecurangan, maka bank akan menandatangani uang terakhir yang masih di dalam amplop itu dan menyerahkannya kepada Beni. Tanda tangan bank akan menembus amplop dan kertas karbon sehingga uang di dalamnya tertandatangani.
5. Beni membuka amplop. Uang siap dipakai.
6. Beni menyerahkan uang kepada Huda. Huda sebagai penerima uang, akan memeriksa apakah tanda tangan bank pada uang itu absah.
7. Huda akan menyuruh Beni untuk membuka salah satu sisi dari setiap string identifikasi di setiap uang dengan cara memberikan string pemilih sepanjang n-bit. Artinya, jika string pemilih itu $b_1, b_2, ..., b_n$ maka Beni harus membuka sisi kiri atau kanan dari I_i, tergantung apakah b_i itu 0 atau 1.
8. Setelah itu Huda membawa uang tersebut ke bank. Bank akan memeriksa apakah nomor seri uang tersebut sudah pernah diterima oleh bank. Kalau belum ada, maka uang tersebut dinyatakan sah.

9. Jika nomor seri uang itu sudah pernah diterima oleh bank, maka bank akan memeriksa string identitas yang sudah terbuka pada uang itu dan membandingkannya dengan string identitas pada uang dengan nomor seri sama yang pernah diterima bank sebelumnya. Jika ternyata string identitas itu sama, maka berarti Huda yang menggandakan uang tersebut. Namun jika berbeda, maka berarti Beni yang menggandakan uang digital tersebut.

2.5.4 Penyerangan Terhadap Protokol

Penyerangan *cryptographic* dapat ditujukan pada beberapa hal berikut :

a. algoritma *cryptographic* yang digunakan dalam protokol;
b. teknik *cryptographic* yang digunakan untuk mengimplementasikan algoritma dan protokol;
c. protokol itu sendiri;

Seseorang dapat mencoba berbagai cara untuk menyerang suatu protokol. Mereka yang tidak terlibat dalam protokol dapat menyadap sebagian atau seluruh protokol. Tindakan ini disebut penyerangan pasif, karena si penyerang tidak mempengaruhi atau mengubah protokol, ia hanya mengamati protokol dan berusaha untuk memperoleh informasi.

Selain itu, seorang penyerang dapat berusaha untuk mengubah protokol demi keuntungannya sendiri. Ia dapat mengirimkan pesan dalam protokol, menghapus pesan, atau bahkan mengubah informasi yang ada di dalam suatu komputer. Tindakan-tindakan ini disebut sebagai penyerangan aktif, karena ia membutuhkan suatu campur tangan aktif.

Seorang penyerang tidaklah hanya berasal dari lingkungan luar protokol, namun ia mungkin juga berasal dari dalam protokol itu sendiri, ia dapat merupakan salah satu pihak yang terlibat dalam protokol. Tipe penyerang semacam ini disebut sebagai *cheater*. *Passive cheater* mengikuti protokol, tetapi berusaha memperoleh informasi lebih banyak

daripada yang diperbolehkan protokol bagi dirinya. *Active cheater* mengubah protokol dalam usahanya untuk berbuat curang.

Usaha untuk menjaga keamanan protokol akan semakin sulit apabila pihak-pihak yang terlibat umumnya merupakan active cheater, oleh karena itu suatu protokol yang baik harus mampu atau pun harus aman terhadap kemungkinan *passive cheating* maupun *active cheating*.

2.5.4.1 Jenis-jenis Pola Penyerangan

Proteksi data dan informasi dalam komunikasi komputer menjadi penting karena nilai informasi itu sendiri dan meningkatnya penggunaan komputer di berbagai sektor. Melihat kenyataan semakin banyaknya data yang diproses dengan komputer dan dikirim melalui perangkat komunikasi elektronik maka ancaman terhadap pengamanan data akan semakin meningkat. Beberapa pola ancaman atau serangan pada komunkasi data dalam komputer dapat diterangkan sebagai berikut :

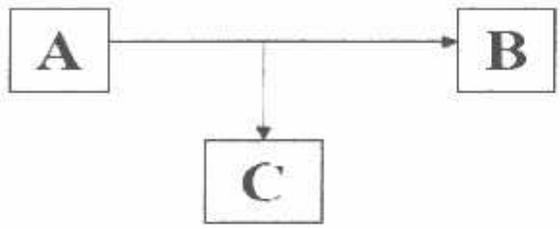
1. Interruption

Interruption terjadi bila data yang dikirimkan dari A tidak sampai pada orang yang berhak (B). *Interruption* merupakan pola penyerangan terhadap sifat *availability* (ketersediaan data).

Gambar 2.6 *Interruption*

2. Interception

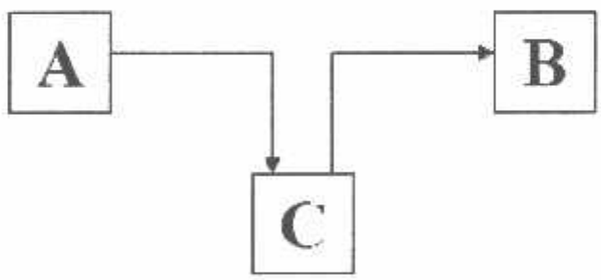
Serangan ini terjadi bila pihak ketiga C berhasil membaca data yang dikirimkan. *Interception* merupakan pola penyerangan terhadap sifat *confidentiality* (kerahasiaan data).



Gambar 2.7 *Interception*

3. Modification

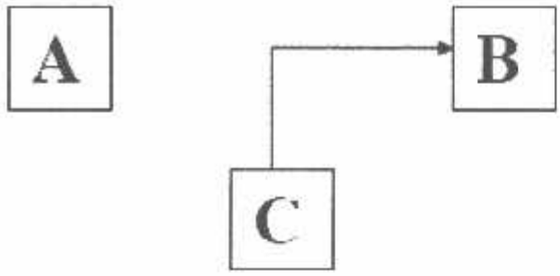
Pada serangan ini pihak ketiga C berhasil merubah pesan yang dikirimkan. *Modification* merupakan pola penyerangan terhadap sifat *integrity* (keaslian data).



Gambar 2.8 *Modification*

4. Fabrication

Pada serangan ini, penyerang berhasil mengirimkan data ke tujuan dengan memanfaatkan identitas orang lain. *Fabrication* meupakan pola penyerangan terhadap sifat *authenticity*.



Gambar 2.9 *Fabrication*

Beberapa metode penyadapan data:

1. *Wiretapping*

   Penyadap mencegat data yang ditransmisikan pada saluran kabel komunikasi dengan menggunakan sambungan perangkat keras.

2. *Electromagnetic eavesdropping*

   Penyadap mencegat data yang ditransmisikan melalui saluran *wireless*, misalnya radio dan *microwive*.

3. *Acoustic Eavesdropping*.

   Menangkap gelombang suara yang dihasilkan oleh suara manusia.

Yang dimaksud dengan serangan (*attack*) adalah usaha (*attempt*) atau percobaan yang dilakukan oleh kriptanalis.

Jenis-jenis serangan:

1. ***Exhaustive attack atau brute force attack***

   Percobaan yang dibuat untuk mengungkap plainteks atau kunci dengan mencoba semua kemungkinan kunci (*trial and error*).

   Asumsi yang digunakan:

   a. Kriptanalis mengetahui algoritma kriptografi
   b. Kriptanalis memiliki sebagian plainteks dan chiperteks yang bersesuaian.

Caranya: plainteks yang diketahui dienkripsikan dengan setiap kemungkinan kunci, dan hasilnya dibandingkan dengan chiperteks yang bersesuaian. Jika hanya chiperteks yang tersedia, chiperteks tersebut didekripsi dengan dengan setiap kemungkinan kunci dan plainteks hasilnya diperiksa apakah mengandung makna.

Misalkan sebuah sistem kriptografi membutuhkan kunci yang panjangnya 8 karakter, karakter dapat berupa angka (10 buah), huruf (26 huruf besar dan 26 huruf kecil), maka jumlah kunci yang harus dicoba adalah

$$62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 = 62^8 \text{ buah}$$

Secara teori, serangan secara *exhaustive* ini dipastikan berhasil mengungkap plainteks tetapi dalam waktu yang sangat lama (lihat Tabel 2.1).

Tabel 2.1 Waktu yang diperlukan untuk *exhaustive key search* (Sumber: William Stallings, *Data and Computer Communication Fourth Edition*)

Ukuran kunci	Jumlah kemungkinan kunci	Lama waktu untuk 10^6 percobaan per detik	Lama waktu untuk 10^{12} percobaan per detik
16 bit	$2^{16} = 65536$	32.7 milidetik	0.0327 mikrodetik
32 bit	$2^{32} = 4.3 \times 10^9$	35.8 menit	2.15 milidetik
56 bit	$2^{56} = 7.2 \times 10^{16}$	1142 tahun	10.01 jam
128 bit	$2^{128} = 4.3 \times 10^{38}$	5.4×10^{24} tahun	5.4×10^{18} tahun

Untuk menghadapi serangan ini, perancang kriptosistem (kriptografer) harus membuat kunci yang panjang dan tidak mudah ditebak.

2. Analytical attack

Pada jenis serangan ini, kriptanalis tidak mencoba-coba semua kemungkinan kunci tetapi menganalisis kelemahan algoritma kriptografi untuk mengurangi kemungkinan kunci yang tidak mungkin ada.

Analisis dilakukan dengan dengan memecahkan persamaan-persamaan matematika (yang diperoleh dari definisi suatu algoritma kriptografi) yang mengandung peubah-peubah yang merepresentasikan plainteks atau kunci. Asumsi yang digunakan: kriptanalis mengetahui algoritma kriptografi.

Untuk menghadapi serangan ini, kriptografer harus membuat algoritma kriptografi yang kompleks sedemikian sehingga plianteks merupakan fungsi matematika dari chiperteks dan kunci yang cukup kompleks, dan tiap kunci merupakan fungsi matematika dari chiperteks dan plainteks yang cukup kompleks. Metode *analytical attack* biasanya lebih cepat menemukan kunci dibandingkan dengan *exhaustive attack*.

2.5.4.2 *Man-in-the-Middle Attack*

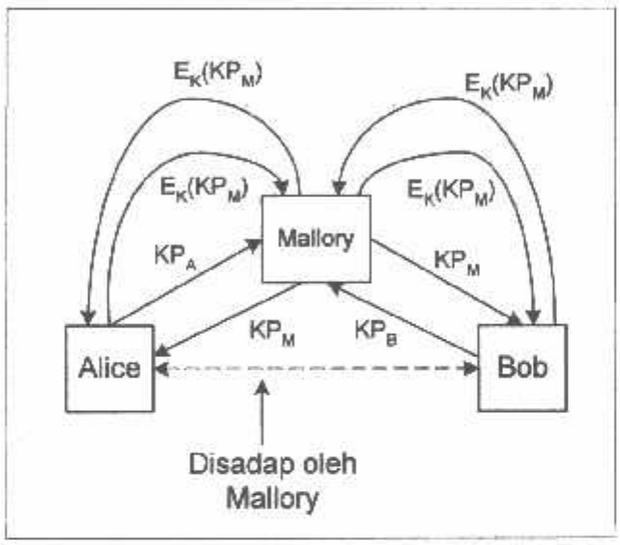
Dalam proses komunikasi data, walaupun data telah dienkripsi, terdapat kemungkinan data tersebut dapat diketahui oleh orang lain. Salah satu kemungkinan tersebut adalah orang tersebut menyadap media komunikasi yang digunakan oleh kedua orang yang sedang berkomunikasi tersebut. Hal inilah yang disebut dengan *man-in-the-middle-attack*.

Dalam keadaan ini, orang yang menyadap berada di antara kedua orang yang sedang berkomunikasi. Data-data yang dikirimkan oleh orang yang sedang berkomunikasi satu sama lain selalu melalui orang yang menyadap tersebut, sehingga orang yang

menyadap tersebut dapat mengetahui semua informasi yang dikirimkan satu sama lain.

Keadaan ini muncul karena kedua orang yang sedang berkomunikasi tersebut tidak dapat mem-verifikasi status dari orang yang berkomunikasi dengannya tersebut, dengan mengambil asumsi bahwa proses penyadapan tersebut tidak menyebabkan gangguan dalam jaringan.

Agar lebih jelas, perhatikan gambar berikut:



Gambar 2.10 Prosedur *Man-in-the-Middle Attack (Active Cheater)*

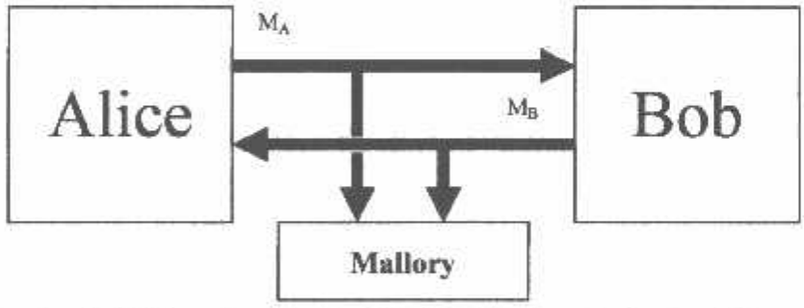
Gambar 2.10 di atas, menjelaskan bahwa Mallory dapat memodifikasi komunikasi antara Alice dan Bob, dengan cara mengubah kunci-kunci yang dikirim oleh Alice dan Bob, menjadi kunci Mallory. Sehingga ia dapat mengetahui apa pesan yang dikirim oleh mereka. Dan juga dapat memodifikasi pesan-pesan yang dikirim oleh Alice dan Bob. Dalam kasus seperti ini, Mallory (orang yang menyadap dan berada di antara saluran komunikasi Bob dan Alice) memegang kekuasaan penuh atas komunikasi yang terjadi.

Sesuai dengan gambar 2.10, maka cara kerja *man-in-the-middle-attack* adalah sebagai berikut :

1. Misalkan Alice dan Bob sedang berkomunikasi dan Mallory ingin menyadapnya (Alice dan Bob menggunakan metode kriptografi kunci publik).
2. Mallory melakukan penyadapan dengan salah satu jenis penyadapan (dibahas pada sub bab 2.5.4.1). Mallory berhasil melakukan penyadapan dan sekarang berada di pertengahan saluran komunikasi Alice dan Bob.
3. Ketika Alice mengirimkan kunci publiknya (KP_A) kepada Bob, Mallory dapat menangkap kunci ini dan mengirimkan kunci publiknya sendiri (KP_M) kepada Bob.
4. Kemudian, ketika Bob mengirimkan kunci publiknya (KP_B) kepada Alice, Mallory dapat menangkap kunci ini dan mengirimkan kunci publiknya sendiri (KP_M) kepada Alice.
5. Alice mengirimkan pesan kepada Bob yang dienkripsi dengan menggunakan kunci publik Mallory ($E_K(KP_M)$). Alice mengira kunci yang dipakai adalah kunci publik Bob dan tidak menyadari bahwa kunci tersebut sebenarnya adalah kunci publik Mallory.
6. Mallory dapat menyadap pesan tersebut. Karena pesan tersebut dienkripsi dengan menggunakan kunci publiknya, maka Mallory dapat mendekripsikan pesan tersebut dengan menggunakan kunci privatnya dan kemudian mengenkripsikan pesan tersebut kembali dengan menggunakan kunci publik dari Bob ($(E_K(KP_B))$ dan mengirimkannya kepada Bob.
7. Bob menerima pesan tersebut, mendekripsi pesan dengan menggunakan kunci privatnya dan mendapatkan pesan asli (*plain text*). Bob tidak akan menyadari bahwa pesan tersebut telah dibaca atau dimodifikasi oleh Mallory.

8. Hal yang sama terjadi ketika Bob mengirimkan pesan kepada Alice yang dienkripsi dengan menggunakan kunci publik Mallory ($E_K(KP_M)$). Bob mengira kunci yang dipakai adalah kunci publik Alice dan tidak menyadari bahwa kunci tersebut sebenarnya adalah kunci publik Mallory.
9. Mallory menyadap pesan tersebut. Karena pesan tersebut dienkripsi dengan menggunakan kunci publiknya, maka Mallory dapat mendekripsikan pesan tersebut dengan menggunakan kunci privatnya dan kemudian menenkripsikan pesan tersebut kembali dengan menggunakan kunci publik dari Alice ($E_K(KP_A)$) dan mengirimkannya kepada Alice.
10. Alice menerima pesan dari Bob, mendekripsikan pesan dengan menggunakan kunci privatnya dan mendapatkan pesan asli (*plain text*). Alice tidak akan menyadari bahwa pesan tersebut telah dibaca atau bahkan dimodifikasi oleh Mallory.

Gambar 2.11 di bawah ini menjelaskan bagaimana *Man-In-The-Middle-Attack* bertindak sebagai *passive cheater*.



Gambar 2.11 Prosedur *Man-In-The-Middle-Attack (Passive Cheater)*

Seperti gambar di atas, Mallory hanya sekedar membaca pesan yang dikirim tanpa mengubah apapun, atau tanpa memodifikasi (menciptakan, menghapus / meniadakan atau mengubah) pesan yang dikirim antara Alice dan Bob. Jadi, Mallory hanya bertindak sebagai penyadap untuk mengetahui pesan apa yang dikirim oleh mereka berdua.

2.6 *Interlock Protocol*

Problema *man-in-the-middle-attack* ini dapat atasi dengan menggunakan *interlock protocol*. *Interlock protocol* ini diciptakan oleh Ron Rivest dan Adi Shamir. Algoritma inti dari protokol ini yaitu protokol ini mengirimkan 2 bagian pesan terenkripsi.

Bagian pertama dapat berupa hasil dari fungsi *hash* satu arah (*one way hash function*) dari pesan tersebut dan bagian kedua berupa pesan terenkripsi itu sendiri.

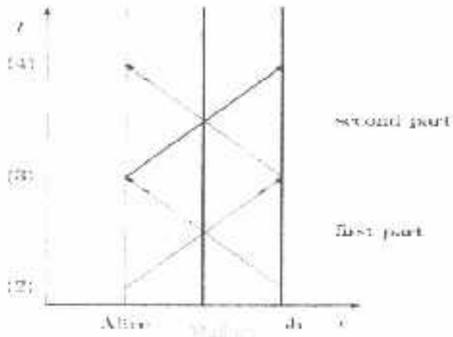
Hal ini menyebabkan orang yang menyadap tersebut tidak dapat mendekripsi pesan pertama dengan menggunakan kunci privatnya. Ia hanya dapat membuat sebuah pesan baru dan mengirimkannya kepada orang yang akan menerima pesan tersebut.

Secara singkat, cara kerja *interlock protocol* adalah sebagai berikut:

1. Alice mengirimkan kunci publiknya kepada Bob.
2. Bob mengirimkan kunci publiknya kepada Alice.
3. Alice mengenkripsi pesannya dengan menggunakan kunci publik Bob. Kemudian, mengirimkan sebagian pesan terenkripsi kepada Bob.
4. Alice mengirimkan bagian lainnya kepada Bob.
5. Bob menggabungkan kedua pesan Alice dan didekripsi dengan menggunakan kunci privatnya.
6. Bob mengenkripsi pesannya dengan menggunakan kunci publik Alice. Kemudian, mengirimkan sebagian pesan terenkripsi kepada Alice.
7. Bob mengirimkan bagian lainnya kepada Alice.
8. Alice menggabungkan kedua pesan Bob dan didekripsi dengan menggunakan kunci privatnya.

Pecahan pesan tersebut dapat berupa :

a. Pecahan pertama berupa *one-way function* dari pesan dan pecahan kedua berupa pesan terenkripsi.
b. Pecahan pertama berupa ½ dari pesan terenkripsi dan pecahan kedua merupakan sisa ½ lainnya.



Gambar 2.12 Cara Kerja *Interlock Protocol*

2.7 Konsep Faktorisasi, Modulo Bilangan Besar dan Tes Prima

Beberapa algoritma kriptografi yang menerapkan konsep faktorisasi, modulo bilangan besar dan tes prima adalah sebagai berikut :

1. RSA (Rivest, Shamir dan Adleman) merupakan algoritma kriptografi kunci publik. Algoritma RSA menggunakan bilangan *integer* besar modulo bilangan prima. RSA menggunakan dua buah bilangan prima acak yang cukup besar dalam proses pembentukan kuncinya. Hal ini dapat dicari dengan menggunakan algoritma GCD. Pada proses enkripsi dan dekripsi, RSA menggunakan konsep modulo bilangan besar yang dapat diselesaikan dengan menggunakan algoritma *Fast Exponentiation*.
2. SAFER (*Secure And Fast Encryption Routine*) menggunakan modulo bilangan besar dalam proses enkripsi dan dekripsinya yang dapat diselesaikan dengan menggunakan algoritma *Fast Exponentiation*.
3. *Blum-Micali Generator* untuk membangkitkan bilangan acak menggunakan dua buah bilangan prima yang cukup besar dan *modulo* kedua bilangan prima tersebut.
4. Pohlig-Hellman yang mirip dengan RSA. Algoritma ini bukan algoritma simetris karena menggunakan kunci yang berbeda untuk enkripsi dan dekripsi. Cara kerjanya hampir sama dengan RSA yaitu menggunakan dua buah bilangan prima acak yang cukup besar dan *modulo* bilangan besar yang dapat diselesaikan dengan menggunakan algoritma *Fast Exponentiation*.

5. Rabin menggunakan modulo bilangan besar pada proses dekripsinya. Algoritma ini menerapkan teorema *Chinese Remainder* untuk mendapatkan hasil pada proses dekripsinya.
6. ElGamal yang dapat digunakan pada *digital signature* (tanda tangan digital) dan enkripsi. Algoritma ini menghasilkan pasangan kunci dengan memilih sebuah bilangan prima p dan dua buah bilangan acak g dan x sehingga g dan x lebih kecil p. Kemudian menghitung nilai dari g^x modulo p.
7. *Digital Signature Algorithm* (DSA) menggunakan bilangan prima acak yang cukup besar, mencari faktorisasi prima dan menghitung modulo bilangan besar.
8. Diffie-Hellman merupakan algoritma pertukaran kunci. Algoritma ini menggunakan konsep modulo bilangan besar dalam proses pada protokolnya. Konsep modulo bilangan besar tersebut dapat diselesaikan dengan menggunakan algoritma *Fast Exponentiation.*

2.7.1 Faktorisasi Bilangan Besar

Dasar dari teori bilangan adalah bilangan itu sendiri dan blok dasar dari bilangan adalah bilangan prima, sehingga bilangan prima sering dikatakan sebagai pusat untuk teori bilangan. Bilangan prima adalah bilangan yang hanya mempunyai dua buah faktor yaitu bilangan satu dan bilangan itu sendiri. Bilangan yang memiliki lebih dari dua buah faktor disebut sebagai bilangan komposit (*composite number*). Bilangan satu hanya memiliki satu faktor dan dianggap bukan bilangan prima dan bilangan komposit.

Ketika sebuah bilangan komposit dituliskan sebagai produk dari semua faktor primanya, maka dikatakan sebagai faktorisasi prima (*prime factorization*) dari bilangan tersebut. Faktorisasi dari sebuah bilangan ke dalam bentuk unsur pokok primanya, disebut juga dekomposisi prima. Biasanya, faktor-faktor didaftarkan mulai dari yang terkecil hingga terbesar.

Dalam ilmu matematika, menentukan faktor-faktor prima merupakan suatu tantangan. Diberikan sebuah integer positif n ≥ 2, faktorisasi prima dituliskan sebagai berikut :

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$$

dimana :

p_i adalah k buah faktor-faktor prima, masing-masing dengan pangkat α_i.

Setiap faktor $p_i^{\alpha_i}$ disebut sebagai sebuah *primary*.

Faktorisasi prima merupakan sebuah permasalahan yang sulit. Banyak algoritma telah disarankan untuk menentukan faktor-faktor prima dari sebuah bilangan yang diberikan.

Salah satunya adalah dengan membagi bilangan tersebut secara berulang dengan bilangan prima hingga sisanya berupa bilangan prima. Metode ini dinamakan metoda *direct search factorization* atau *trial division*. Dalam metode ini, semua faktor-faktor prima yang mungkin dicoba secara sistematis dengan menggunakan percobaan pembagian untuk melihat apakah mereka dapat membagi oleh bilangan yang diberikan. Metode ini hanya dapat dipraktekkan untuk bilangan yang sangat kecil.

Contoh :

Faktorisasi prima dari 96 :

96 ÷ 2 = 48

48 ÷ 2 = 24

24 ÷ 2 = 12

12 ÷ 2 = 6

6 ÷ 2 = 3

3 ÷ 3 = 1

96 = 2 * 2 * 2 * 2 * 2 * 3

Faktorisasi prima dari 120 :

$120 \div 2 = 60$

$60 \div 2 = 30$

$30 \div 2 = 15$

$15 \div 3 = 5$

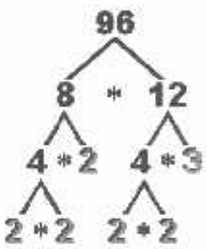
$5 \div 5 = 1$

$120 = 2 * 2 * 2 * 3 * 5$

Metoda lainnya adalah dengan memilih salah satu pasangan faktor dari bilangan tersebut dan kemudian membagi faktor-faktor tersebut hingga semua faktor adalah bilangan prima. Metoda ini dinamakan sebagai metoda pohon faktor (*factor trees*).

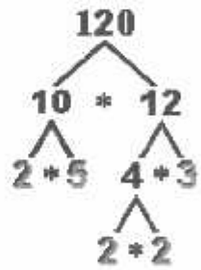
Contoh :

Faktorisasi prima dari 96 :



2*2*2*2*2*3=96

Faktorisasi prima dari 120 :



2*2*2*3*5=120

Pangkat besar dalam aritmatika modular juga dapat difaktorkan. Faktor yang digunakan dalam aritmatika modular adalah faktorisasi prima. Masalah pada perpangkatan bilangan besar adalah semakin besar R, semakin besar P dan Q yang ada dimana P dan Q adalah faktor dari R. Bahkan nilai yang relatif 9^(9^9) menghasilkan sebuah nilai dari panjang

350 juta digit desimal. Bagaimana akan dienkrip sesuatu tanpa membutuhkan penyimpanan *terabytes*?

Caranya adalah hanya menghitung nilai exponential modulo R. Seperti biasanya, aritmatika modular menyederhanakan dari yang besar menjadi kecil..

Adapun contoh dibawah ini adalah sebagai berikut :

28^23 (mod 55) – ?

Sebelum mulai menghitung, lihat biner Q, dimana 23 dalam biner adalah **10111**, yang artinya bahwa :

23 = 16 + 4 + 2 + 1, or 23 = 2^4 + 2^2 + 2^1 + 2^0.

Dapat dipecahkan perhitungan exponential dalam beberapa bagian kecil sebagai berikut :

28^23 = 28^(2^4 + 2^2 + 2^1 + 2^0)

= 28^(2^4) * 28^(2^2) * 28^(2^1) * 28^(2^0)

= 28^(2 * 2 * 2 * 2) * 28^(2 * 2) * 28^2 * 28

= (((28^2)^2)^2)^2 * (28^2)^2 * 28^2 * 28.

Ini kelihatan seperti tidak ada apanya akan tetapi merupakan suatu pembuktian.

Menghitung pangkat dua pertama dalam modular aritmatika :

28^2 = 784 = 14 (mod 55).

Dengan substitusi nilai ini dalam persamaan sebelumnya :

28^23 = (((28^2)^2)^2)^2 * (28^2)^2 * 28^2 * 28 (mod 55), didapat :

28^23 = ((14^2)^2)^2 * 14^2 * 14 * 28 (mod 55).

Sekarang menghitung pangkat 14^2 :

14^2 = 196 = 31 (mod 55), maka akan didapat :

28^23 = (31^2)^2 * 31 * 14 * 28 (mod 55).

Dan akhirinya :

31^2 = 961 = 26 (mod 55), and 26^2 = 676 = 16 (mod 55)

Dan kemudian :

28^23 = 16 * 31 * 14 * 28 (mod 55).

Didapat keuntungan dari modulus ketika melakuakn perkalian terakhir :

28^23 = 16 * 31 * 14 * 28 (mod 55)

= 16 * 31 * 392 (mod 55)

= 16 * 31 * 7 (mod 55)

= 16 * 217 (mod 55)

= 16 * 52 (mod 55)

= 832 (mod 55)

= 7 (mod 55)

Hasil :7, hasil yang sama ketika kita melakukan cara yang sulit.

Teknik biner ini tidak ada bedanya dibandingkan dengan bagaimana komputer secara normal menghitung integer. Bagaimanapun juga, kenyataannya bahwa kita dapat memecahkan proses tersebut dengan melakukan perkalian berturut-turut.

2.7.2 Algoritma Euclidean untuk Mencari GCD (*Greatest Common Divisor*)

Dua buah bilangan dikatakan relatif prima jika mereka tidak memiliki faktor prima yang sama kecuali 1. Dengan perkataan lain, jika *Greatest Common Divisor* (GCD) dari a dan n adalah sama dengan satu, maka a dan n adalah relatif prima. Bentuk ini dapat ditulis seperti berikut :

gcd(a,n) = 1

Salah satu metoda untuk menghitung GCD dari dua buah bilangan adalah dengan menggunakan algoritma *Euclid* yang ditulis dalam bukunya '*Elements*' sekitar 300 tahun sebelum masehi. Algoritma ini bukan hasil rancangannya. Para ahli sejarah menyatakan bahwa algoritma mungkin 200 tahun lebih tua. Algoritma ini merupakan algoritma *nontrivial* tertua yang masih eksis di dunia. Knuth mendeskripsikan algoritma ini dengan beberapa modifikasi modern seperti berikut :

```
Begin
P := A; Q := B;
While Q ≠ 0 Do Begin
R := P Mod Q;
```

```
P := Q;
Q := R;
End
GCD(A,B) := P
End
```

2.7.3 Algoritma Penguji Bilangan Prima Rabin-Miller

Salah satu metode yang paling banyak dipakai untuk menguji bilangan prima adalah metode Rabin-Miller. Algoritma Rabin-Miller menerima masukan berupa bilangan ganjil (n) yang lebih besar sama dengan 3 dan sebuah bilangan bulat (t) yang lebih kecil dari n. Bilangan t adalah parameter keyakinan, yaitu berapa kali pengujian dilakukan terhadap n untuk memastikan bahwa n adalah bilangan prima.

Adapun algoritma Rabin-Miller adalah sebagai berikut :

RABIN-MILLER (n,t)

1. $n-1 = 2^s r$; dengan s ganjil;
2. for $1 \leftarrow 1$ to t
3. do a ← random; dengan $2 \leq a \leq n-2$
4. $y \leftarrow a^r \bmod n$
5. if $y \neq 1$ and $y \neq n-1$
6. do $j \leftarrow 1$
7. while $j \leq s-1$ and $y \neq n-1$
8. do $y \leftarrow y^2 \bmod n$
9. if $y = 1$ return false; bukan prima
10. $j \leftarrow j + 1$
11. if $y \neq n - 1$ return false; bukan prima
12. return true; prima

2.7.4 Eksponensial Secara Cepat dengan *Fast Exponentiation*

Karena banyak algoritma enkripsi maupun dekripsi yang sering menggunakan ekponensial modulo n, maka untuk mempermudah pencariannya dilakukan dengan menggunakan algoritma untuk eksponensial secara cepat. Adapun algoritma tersebut adalah sebagai berikut:

Untuk menghasilkan nilai dari a^m (mod n)

Ubah m menjadi bentuk biner yaitu $b_k, b_{k-1}, \ldots, b_0$

hasil = a

for j = k downto 0

begin

If $b_j = 0$ then

  hasil = hasil ^ 2

Else if $b_j = 1$ then

    hasil = hasil ^ 2 * a

hasil = hasil (mod n)

End

Contoh : 3^6 (mod 5)

m = 6, dijadikan bilangan biner sehingga menjadi : 110 ($b_2 = 1$; $b_1 = 1$; $b_0 = 0$)

a = 3, n = 5, k = 2 dan hasil = 3

Pada saat j = 2

$b_2 = 1$ maka : hasil = $3^2 \times 3 = 27$

hasil = 27 (mod 5) ≡ 2 (mod 5)

Pada saat j = 1

$b_1 = 1$ maka : hasil = $2^2 \times 3 = 12$

hasil = 12 (mod 5) ≡ 2 (mod 5)

Pada saat j = 0

$b_0 = 0$ maka : hasil = $2^2 = 4$

hasil = 4 (mod 5) ≡ 4 (mod 5)

Sangat penting untuk memperoleh sebuah bilangan yang disajikan dalam bentuk yang efisien seperti algoritma eksponensial secara cepat. Jika GCD (a , n) = 1 dan mengandung sebuah integer b , 0<a<n, dimana a . b η 1 (mod n) maka b merupakan invers dari a. Contoh : GCD(7,11) = 1 dan b = 8 maka 7 . 8 η 1 (mod 11).

2.8 Bilangan Acak

Bilangan acak adalah deretan angka atau nilai yang acak dan tidak dapat diprediksi secara keseluruhan. Untuk menghasilkan bilangan acak merupakan hal yang sulit, kebanyakan pembangkit bilangan acak (*random number generator* = RNG) mempunyai beberapa bagian yang dapat diprediksi dan berhubungan dan biasanya kebanyakan RNG mengulang *string* yang sama setelah melakukan *n* putaran.

Suatu pembangkit bilangan acak atau *random number generator* (RNG) adalah suatu peralatan komputasional atau fisik yang dirancang untuk menghasilkan suatu urutan nilai yang tidak dapat mudah ditebak polanya, jadi urutan tersebut dapat diperlakukan sebagai suatu keadaan acak. Setiap *output* dari suatu RNG adalah berupa *true random numbers*, yang berarti nilai-nilai dimana secara keseluruhan bersifat acak, tidak dapat diprediksi dan tidak bergantung pada nilai-nilai yang dihasilkan sebelumnya. Untuk menghasilkan *true random numbers* merupakan hal yang sulit.

Sedangkan pembangkit bilangan acak semu atau *pseudo random number generator* (PRNG) merupakan suatu algoritma yang menghasilkan suatu urutan nilai dimana elemen-elemennya bergantung pada setiap nilai yang dihasilkan. Bilangan acak yang dihasilkan adalah bilangan acak semu (*pseudo*), karena pembangkitan bilangannya dapat diulang kembali. Oleh karena itu, pembangkit deret bilangan acak semacam itu disebut pembangkit bilangan acak semu (*pseudo random number generator* atau PRNG).

2.9 *VISUAL BASIC*

Microsoft Visual Basic (sering disingkat sebagai VB saja) merupakan sebuah bahasa pemrograman yang menawarkan *Integrated Development Environment* (IDE) visual untuk membuat program perangkat lunak berbasis sistem operasi *Microsoft Windows* dengan menggunakan model pemrograman (COM). *Visual Basic* merupakan turunan bahasa pemrograman *BASIC* dan menawarkan pengembangan perangkat lunak komputer berbasis grafik dengan cepat.

Beberapa bahasa skrip seperti *Visual Basic for Applications* (VBA) dan *Visual Basic Scripting Edition* (VBScript), mirip seperti halnya *Visual Basic*, tetapi cara kerjanya yang berbeda. Para programmer dapat membangun aplikasi dengan menggunakan komponen-komponen yang disediakan oleh Microsoft *Visual Basic* Program-program yang ditulis dengan *Visual Basic* juga dapat menggunakan *Windows API*, tapi membutuhkan deklarasi fungsi luar tambahan.

Dalam pemrograman untuk bisnis, *Visual Basic* memiliki pangsa pasar yang sangat luas. Sebuah survey yang dilakukan pada tahun 2005 menunjukkan bahwa 62% pengembang perangkat lunak dilaporkan menggunakan berbagai bentuk *Visual Basic*, yang diikuti oleh C++, *JavaScript*, C#, dan *Java*.

Visual Basic adalah pengembangan dari bahasa komputer *BASIC* (*Beginner's All-purpose Symbolic Instruction Code*). Bahasa *BASIC* diciptakan oleh Professor John Kemeny dan Thomas Eugene Kurtz dari Perguruan Tinggi Dartmouth pada pertengahan tahun 1960-an. Bahasa program tersebut tersusun mirip dengan bahasa Inggris yang biasa digunakan oleh para programer untuk menulis program-program komputer sederhana yang berfungsi sebagai pembelajaran bagi konsep dasar pemrograman komputer.

Sejak saat itu, banyak versi *BASIC* yang dikembangkan untuk digunakan pada berbagai *platform* komputer, seperti *Microsoft QBASIC, QUICKBASIC, GWBASIC, IBM BASICA, Apple BASIC* dan lain-lain.

Apple BASIC dikembangkan oleh Steve Wozniak, mantan karyawan Hewlett Packard dan teman dekat Steve Jobs (pendiri Apple Inc.). Steve Jobs pernah bekerja dengan Wozniak sebelumnya (mereka membuat game arcade

"*Breakout*" untuk Atari). Mereka mengumpulkan uang dan bersama-sama merakit PC, dan pada tanggal 1 April 1976 mereka secara resmi mendirikan perusahaan komputer Apple. Popularitas dan pemakaian *BASIC* yang luas dengan berbagai jenis komputer turut berperan dalam mengembangkan dan memperbaiki bahasa itu sendiri, dan akhirnya berujung pada lahirnya *Visual Basic* yang berbasis GUI (*Graphic User Interface*) bersamaan dengan *Microsoft Windows*. Pemrograman *Visual Basic* begitu mudah bagi pemula dan programer musiman karena ia menghemat waktu pemrograman dengan tersedianya komponen-komponen siap pakai.

Hingga akhirnya *Visual Basic* juga telah berkembang menjadi beberapa versi, sampai yang terbaru, yaitu *Visual Basic* 2010. Bagaimanapun juga *Visual Basic* 6.0 tetap menjadi versi yang paling populer karena mudah dalam membuat programnya dan ia tidak menghabiskan banyak memori.

Sejarah *BASIC* di tangan *Microsoft* sebagai bahasa yang diinterpretasi (BASICA) dan juga bahasa yang dikompilasi (BASCOM) membuat *Visual Basic* diimplementasikan sebagai gabungan keduanya. Programmer yang menggunakan *Visual Basic* bisa memilih kode bahasa pemrograman yang dikompilasi atau kode yang harus bahasa pemrograman yang diinterpretasikan sebagai hasil porting dari kode VB.

BAB III
ANALISA DAN DESAIN SISTEM

3.1 Analisa Sistem

Pada bab ini dijelaskan mengenai analisa dan desain sistem aplikasi. Analisis ditujukan untuk memberikan gambaran secara umum terhadap aplikasi. Hal ini berguna untuk menunjang desain sistem aplikasi yang akan dikerjakan sehingga kebutuhan akan aplikasi tersebut dapat diketahui sebelumnya. Kemudian hasil analissa akan menjadi dasar untuk melakukan desain sistem atau perancangan aplikasi sesuai kebutuhan sistem.

Untuk dapat berkomunikasi antara *client* dan *server*, maka pihak *server* harus membuka koneksi untuk *client* dan menentukan *port* berapa yang akan digunakan untuk berkomunikasi. Sehinggga *client* dapat memulai koneksi kepada *server* dengan menginputkan *IP* dan *port* yang ada pada *server*. Data komunikasi yang terjadi akan terekam pada aplikasi yang telah dibuat pada aplikasi berbasis windows, lebih khususnya dikembangkan dengan menggunakan pemrograman *Visual Basic*.

Agar dapat saling berkomunikasi dengan bertukar pesan *client* dan *server* terlebih dahulu menginputkan sebuah kunci yang dibutuhkan untuk proses enkripsi dan dekripsi pesan pada algoritma RSA yang dihasilkan secara acak oleh computer. Pada saat *client* dan *server* telah terhubung secara otomatis *client* akan memngirimkan kunci publiknya kedapa *server*, begitu juga sebaliknya *server* juga akan mengirim kunci publiknya pada *client*. Cara yang digunakan untuk mengirim pesan antara *client* dan *server* adalah sebelum pesan akan dikirim terlebih dahulu pesan atau plain teks akan dienkripsi terlebih dahulu menggunakan kunci public penerima pesan agar menjadi *chipper* teks, setelah sampai pada penerima *chipper* teks tersebut akan di dekripsi menggunakan kunci privat penerima agar kembali menjadi *plain* teks atau pesan asli.

3.1.1 Algoritma-algoritma pendukung yang digunakan

Algoritma pendukung yang digunakan pada aplikasi enkripsi pesan dengan kriptografi metode *interlock protocol* untuk mengatasi *man-in-the-middle-attack* yang akan dicodekan pada *Visual Basic* sebagai modul. Algoritma-algoritma tersebut antara lain algoritma tes *Rabbin-Miller*, algoritma *Greatest Common Divisor (GCD)*, algoritma *Fast Eksponentiation*, algoritma *Extended Euclidean*. Penjelasan dan *pseudocode* dari algoritma-algoritma tersebut adalah sebagai beikut:

1. Algoritma tes prima *Rabin-Miller*.

   Algoritma ini dipakai untuk menguji apakah suatu bilangan merupakan bilangan prima atau bukan. Algoritma ini dipakai dalam algoritma RSA untuk menentukan sifat prima dari bilangan acak p dan q. Algoritma tes prima Rabin-Miller ditampilkan dalam bentuk *pseudocode* adalah sebagai berikut:

```
'Tes Prima Rabin Miller terhadap bilangan P dengan
nilai A
Fungsi IsRabinMiller(pnP, pnA) As Boolean

Set pnC ← pnP - 1
'------- CARI nilai b
Set nTemp ← 0
Selama (pnC Mod (2 ^ nTemp)) ← 0 DAN ((2 ^ nTemp) <
 pnP)
   Set nTemp ← nTemp + 1
    End Selama
    Set pnB ← nTemp - 1
        '------- CARI nilai m
    Set pnM ← (pnC / (2 ^ pnB))
    Set pnJ ← 0
```

```
    Set pnZ ← FastExp(pnA, pnM, pnP)

    Jika (pnZ ← 1) ATAU (pnZ ← (pnP - 1)) Maka
        IsRabinMiller ← True (pnP adalah bilangan
prima)
        KELUAR DARI FUNGSI
    End Jika

lAgain:

    Jika pnJ > 0 DAN pnZ ← 1 Maka
        IsRabinMiller ← False (pnP bukan bilangan
prima)
        KELUAR DARI FUNGSI
    End Jika

    Set pnJ ← pnJ + 1
    Jika (pnJ < pnB) And (pnZ <> pnP - 1) Maka
        pnZ ← FastExp(pnZ, 2, pnP)
        GoTo lAgain
    End Jika

    Jika pnZ ← (pnP - 1) Maka
        IsRabinMiller ← True (pnP adalah bilangan
prima)
        KELUAR DARI FUNGSI
    End Jika

    Jika (pnJ ← pnB) DAN pnZ <> (pnP - 1) Maka
```

```
        IsRabinMiller ← False (pnP bukan bilangan
prima)
        KELUAR DARI FUNGSI
    End jika
End Fungsi
```

2. Algoritma *Greatest Common Divisor* (GCD).

Algoritma ini dipakai untuk mencari nilai faktor persekutuan terbesar dari dua bilangan. Algoritma ini dipakai dalam algoritma RSA untuk menentukan nilai GCD dari beberapa nilai. Algoritma GCD ditampilkan dalam bentuk *pseudocode* adalah sebagai berikut:

```
'Fungsi mengembalikan nilai GCD(A, B)
FUNGSI GCD(A, B) As Long

    Set P ← A
    Set Q ← B
    Selama Q <> 0
        Set R ← P mod Q
        Set P ← Q
        Set Q ← R
    End Selama

    'Kembalikan nilai GCD
    Set GCD ← P
End FUNGSI
```

3. Algoritma *Fast Exponentiation*.

Algoritma ini dipakai untuk mencari nilai dari persamaan (a^b mod c) secara cepat. Algoritma *Fast Exponentiation* dibutuhkan, karena perhitungan dengan persamaan eksponensial dengan nilai variabel yang

besar sulit dihitung manual dan bahkan dengan kalkulator, karena nilai pangkat biasanya akan mengakibatkan nilai menjadi *overflow*. Algoritma *Fast Exponentiation* ditampilkan dalam bentuk *pseudocode* adalah sebagai berikut:

```
'Fungsi mengembalikan nilai (A^B) mod C
FUNGSI FastExp(A, B, C) As Double

    Set A1 ← A
    Set B1 ← B
    Set Product ← 1
    Selama (B1 <> 0)
        Selama B1 mod 2 ← 0
            Set B1 ← B1 div 2
            Set A1 ← (A1 * A1) mod C
        End Selama
        Set B1 ← B1 - 1
        Set Product ← (Product * A1) mod C
    End Selama

    'Kembalikan nilai fast exponentiation
    Set FastExp ← Product
End FUNGSI
```

4. Algoritma *Extended Euclidean.*

   Algoritma ini dipakai untuk mencari inversi modulo (misalnya untuk persamaan e^{-1} mod n). Algoritma *Extended Euclidean* ditampilkan dalam bentuk *pseudocode* adalah sebagai berikut:

```
'Fungsi mengembalikan nilai (NilaiX^-1) mod pnValueA11
FUNGSI ExtendedEuclidean(NilaiX, pnValueA11) As Double

    Set bSelesai ← False

    'Bentuk Array
    Set A(1, 1) ← pnValueA11
    Set A(1, 2) ← NilaiX
    'Matriks Identitas
    Set A(2, 1) ← 1
    Set A(2, 2) ← 0
    Set A(3, 1) ← 0
    Set A(3, 2) ← 1

'Lakukan looping
    Selama bSelesai ← False
        'Hitung nilai m
        Set nM ← A(1, 1) div A(1, 2)
        Untuk nI ← 1 Sampai 3
            'Hitung nilai x
            Set nX ← A(nI, 1) - nM * A(nI, 2)
            'Ubah nilai
            Set A(nI, 1) ← A(nI, 2)
            Set A(nI, 2) ← nX
            Jika (nI ← 1) DAN (nX ← 0) Maka
                Set bSelesai ← True
            End Jika
        Next nI
    End Selama
```

```
    Jika A(3, 1) >= 0 Maka,
        'Kembalikan hasil extended euclidean
        Set ExtendedEuclidean ← A(3, 1)
    Jika tidak,
        'Kembalikan hasil extended euclidean
        Set ExtendedEuclidean ← A(3, 1) + pnValueAll
    End Jika
End FUNGSI
```

3.1.2 Mekanisme Enkripsi dan Dekripsi

Sebelum pesan atau *plain* teks yang dinputkan di enkripsi menggunakan algoritma RSA yaitu perhitungan *Fast Exponetiation*, pertama *plain* teks harus dirubah ke dalam bentuk binner. Perubahan karakter pada *plain* teks ke dalam bentuk binner dapat dilihat pada tabel 3.1 berikut ini.

Tabel 3.1 Perubahan karakter ASCII ke binner

Karakter	Binner	Letter	Binner
a	01100001	A	01000001
b	01100010	B	01000010
c	01100011	C	01000011
d	01100100	D	01000100
e	01100101	E	01000101
f	01100110	F	01000110
g	01100111	G	01000111
h	01101000	H	01001000
i	01101001	I	01001001
j	01101010	J	01001010
k	01101011	K	01001011
l	01101100	L	01001100
m	01101101	M	01001101
n	01101110	N	01001110
o	01101111	O	01001111
p	01110000	P	01010000
q	01110001	Q	01010001
r	01110010	R	01010010

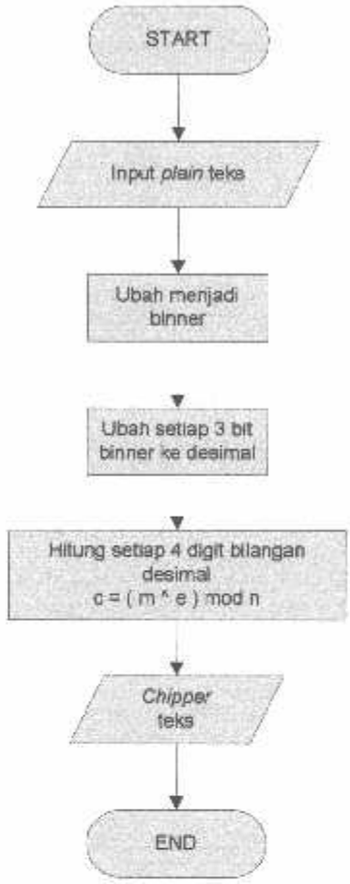
s	01110011	S	01010011
t	01110100	T	01010100
u	01110101	U	01010101
v	01110110	V	01010110
w	01110111	W	01010111
x	01111000	X	01011000
y	01111001	Y	01011001
z	01111010	Z	01011010

Setelah itu untuk setiap 3 bit binner dari total hasil perubahan karakter ke binner, dirubah kebentuk desimal. Perubahan 3 bit binner ke dalam bentuk desimal dapat dilihat pada tabel 3.2 berikut ini.

Tabel 3.2 Perubahan 3 bit binner ke desimal

3 bit binner	Desimal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Jika semua bit binner telah dirubah ke dalam bentuk desimal, lalu hitung setiap 4 digit bilang desimal tersebut pada fungsi enkripsi c = (m ^ e) mod n, dimana (c) adalah *chipper* teks, (m) adalah nilai desimal dari *plain* teks, (e) adalah kunci publik penerima, dan (n) adalah kunci publik pengirim. Hasil dari perhitungan tersebut akan jadi *chipper* teks yang nantinya akan dijadikan pesan yang akan dikirim.



Gambar 3.1 Flowchart proses enkripsi

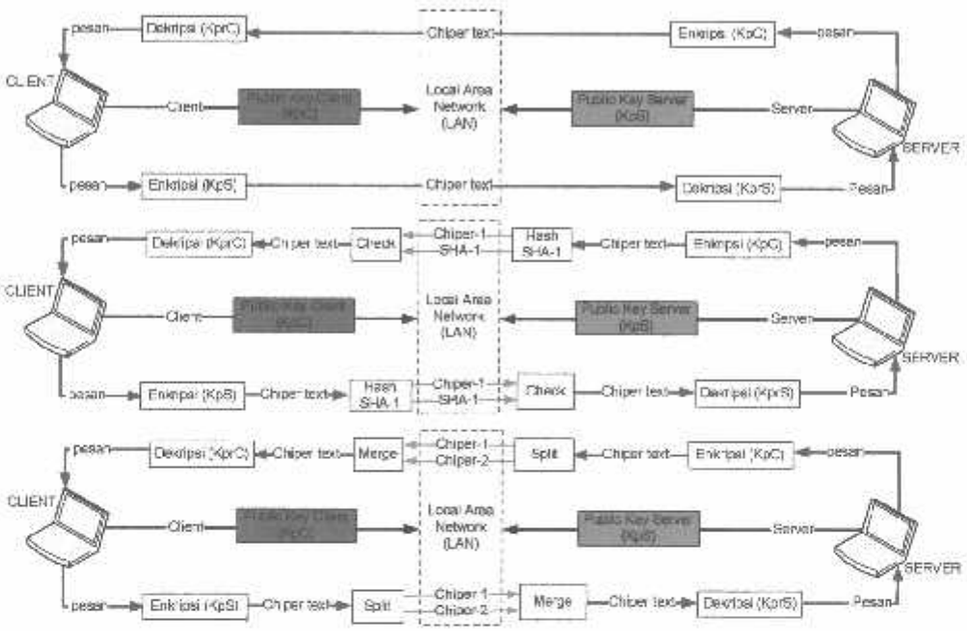
Kebalikan dari proses enkripsi, pada proses dekripsi *chipper* teks yang diterima akan dihitung dengan fungsi dekripsi m = (c ^ d) mod n, dimana (m) adalah nilai desimal dari *plain* teks, (c) adalah *chipper* teks, (d) adalah kunci privat penerima dan (n) adalah kunci publik penerima. Setelah proses perhitungan selesai, maka akan dihasilkan nilai desimal dari *plain* teks yang selanjutnya akan dirubah ke dalam bentuk binner dan dari binner akan dirubah ke bentuk karakter ASCII.

Gambar 3.2 Flowchart proses dekripsi

3.2 Desain Sistem

Sistem yang akan dibuat pada penelitian ini adalah sebuah aplikasi yang dapat mengenkripsi pesan yang dikrimkan pada proses percakapan yang dilakukan pada jaringan lokal dengan menggunakan kriptografi metode *Interlock Protocol* yang memiliki 2 variasi pengiriman pesan, sehingga memberikan keamanan pada pesan yang dikirim pada percakapan yang terjadi antara kedua belah pihak, untuk menghindari pengintaian antara komunikasi yang sedang berlangsung (*Man In The Middle Attack)*.

Aplikasi yang dirancang untuk komunikasi pada jaringan secara point to point. Ada dua macam aplikasi yang akan dirancang yaitu untuk bertindak sebagai *server* dan *client*, yang nantinya akan dijalankan pada dua komputer yang terhubung.



Gambar 3.3 Desain Sistem

Keterangan gambar 3.3 dari desain sistem:

- Pada saat *client* dan *server* terhubung keduanya saling bertukar kunci Publik *Client* (KpC), kunci publik *Server* (KpS), masing-masing digunakan untuk mengenkripsi pesan. Sedangkan kunci privat *Client* (KprC) dan kunci privat *Server* (KprS), masing-masing digunakan untuk mendekripsi pesan.
- Pada proses pengiriman pesan yang pertama tanpa menggunakan *Interlock Protocol*, pesan yang dikirim dienkripsi menggunakan kunci publik penerima. Setelah sampai pada penerima, pesan tersebut akan didekripsi menggunakan kunci privat penerima.

- Pada proses pengiriman pesan yang kedua menggunakan *Interlock Protocol* tipe-1, yaitu pesan akan dikirim 2 kali yang pertama dikirim adalah *chipper* teks hasil enkripsi menggunakan kunci publik penerima yang telah di *digest* menggunakan fungsi *One way Hash SHA-1*, pesan berikutnya adalah *chipper* teks itu sendiri. Pesan pertama yang terkirim akan disimpan, selanjutnya pesan kedua akan di *digest* terlebih dahulu dan disamakan nilainya dengan pesan pertama, jika nilainya sama maka *chipper* teks akan didekripsi menggunakan kunci privat penerima.
- Pada proses pengiriman pesan yang ketiga menggunakan *Interlock Protocol* tipe-2, yaitu pesan akan dienkripsi menggunakan kunci publik penerima, setelah itu *chipper* teks hasil enkripsi akan dipecah menjadi 2 bagian yang sama besar. Pesan akan dikirim secara bergantian, pesan pertama yang terkirim akan disimpan, selanjutnya pesan kedua yang terkirim akan disimpan lalu digabungkan dengan pesan pertama sehingga menjadi *chipper* teks utuh yang akan didekripsi menggunakan kunci privat penerima.

3.2.1 Desain Aplikasi

Aplikasi kriptografi untuk enkripsi pesan menggunakan metode *interlock protocol* untuk mengatasi *man-in-the-middle-attack* dirancang dengan menggunakan bahasa pemrograman *Microsoft Visual Basic 6.0* dengan beberapa komponen *standard* seperti:

1. *Command Button*, sebagai tombol.
2. *Text Box*, sebagai tempat *input*.
3. Label, untuk menampilkan tulisan.
4. *Image* atau *Picture Box*, sebagai komponen untuk menampilkan gambar.
5. *Winsock* atau *Windows Socket*, sebagai komponen penghubung jaringan
6. *Timer*, untuk melakukan proses *delay*.

7. *Check Button*, sebagai komponen untuk memilih.

Aplikasi ini memiliki beberapa buah *form*, yaitu:

1. *Form Splash Screen*.
2. *Form* Utama.
3. *Form* Input Kunci.
4. *Form About*.

3.2.1.1 Form Splash Screen

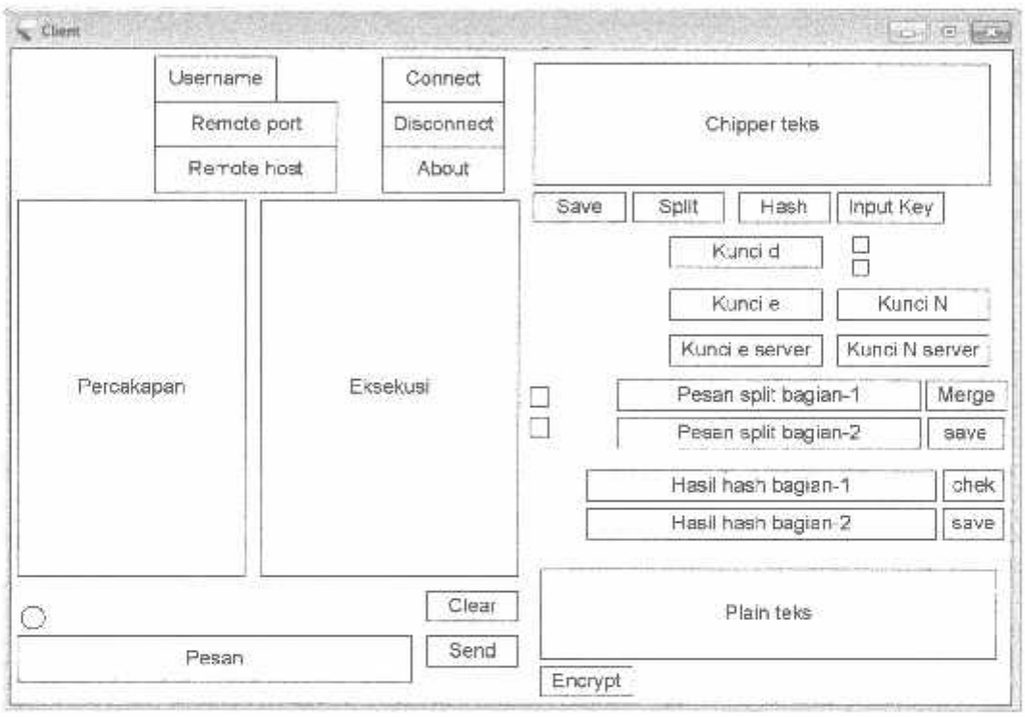
Form Splash Screen adalah form yang pertama kali dibuka saat aplikasi dijalankan, berisi nama aplikasi, logo serta identitas pembuat aplikasi. Rancangan *form* Splash Screen dapat dilihat pada gambar 3.4 berikut:

Gambar 3.4 Rancangan *Form* Splash Screen

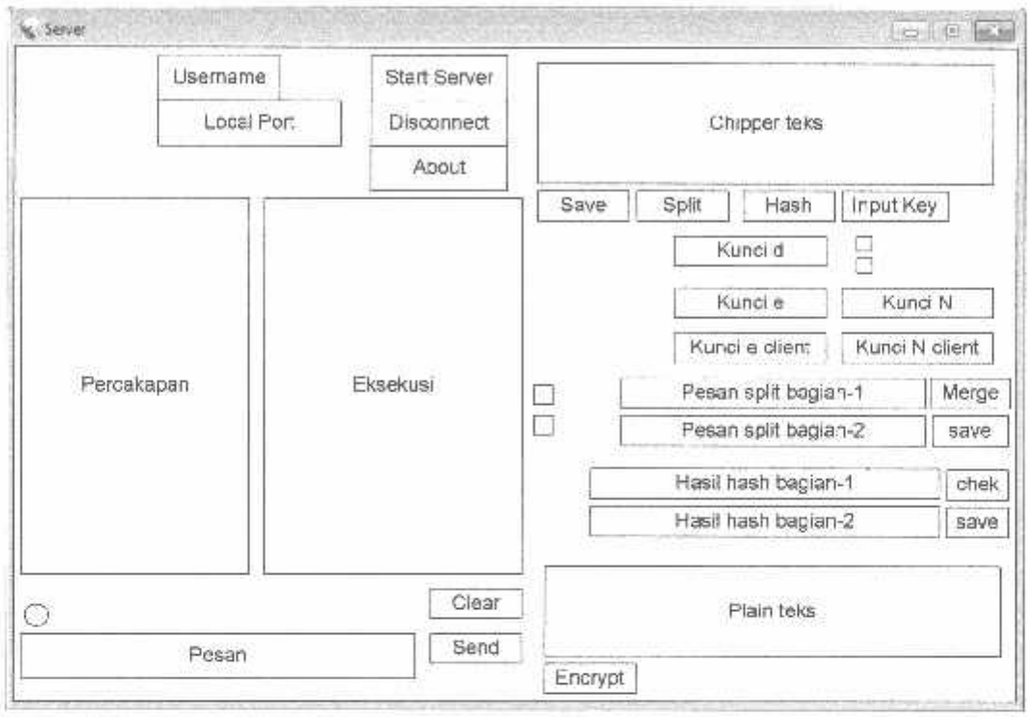
3.2.1.2 Form Utama

Form utama adalah form paling penting, berisi kolom-kolom teks yang digunakan untuk menginputkan nilai dan menampilkan proses eksekusi serta tombol-tombol perintah yang dibutuhkan untuk koneksi antara *client* dan *server*, proses enkripsi, proses dekripsi, menginputkan dan mengirim pesan, untuk membuka *form input* kunci dan *form About*. Juga terdapat kolom teks

dan tombol perintah yang digunakan untuk mengaplikasikan *Interlock Protocol* diantaranya untuk membagi *chipper* teks menjadi 2 bagian dan menggabungkannya, serta untuk menghasilkan nilai Hash dan membandingkan nilainya. Rancangan *form* Utama terdapat dua macam yaitu untuk *client* pada gambar 3.5 dan pada *server* gambar 3.6 berikut:



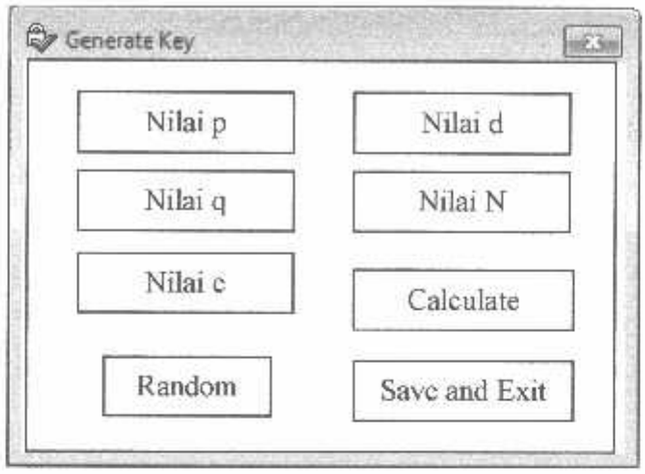
Gambar 3.5 Rancangan *Form* Utama *Client*



Gambar 3.6 Rancangan *Form* Utama *Server*

3.2.1.3 Form Input Kunci

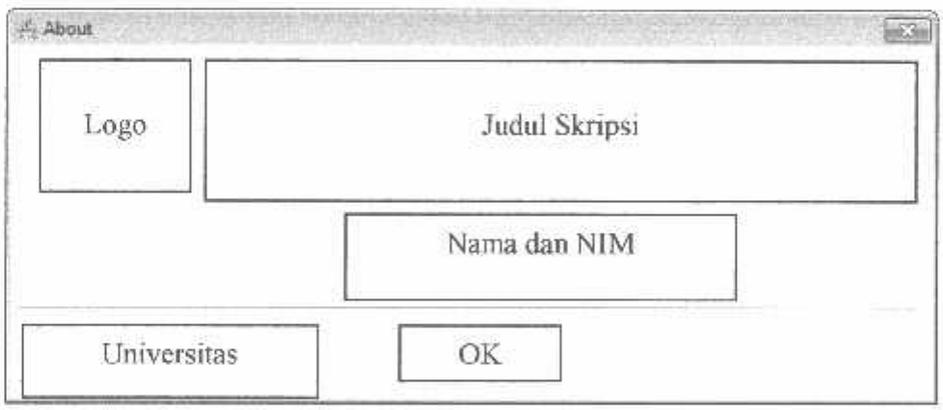
Form Input Kunci berfungsi sebagai *form* untuk memasukkan dan menghitung nilai-nilai pembentuk kunci publik dan kunci privat dari *client* dan *server*. Rancangan *form* Input Kunci dapat dilihat pada gambar 3.7 berikut:



Gambar 3.7 Rancangan *Form* Input Kunci

3.2.1.4 Form About

Form About adalah *form* yang berisi nama mahasiswa, NIM, universitas, judul skripsi dan logo. Rancangan *form* About dapat dilihat pada gambar 3.8 berikut.



Gambar 3.8 Rancangan *Form* About

3.2.2 *Flowchart* Sistem

Berdasarkan desain sistem yang dibuat, maka terdapat empat diagram alir dari aplikasi kriptografi untuk enkripsi pesan dengan metode *Interock Protocol* adalah sebagai berikut :

1. *Flowchart* proses koneksi antara *client* dan *server*

Gambar 3.9. Flowchart koneksi antara *Client* dan *Server*

Pada gambar 3.9 *Flowchart* menggambarkan bagaiamna proses koneksi antara *Client* dan *Server* mulai dari mengisikan *username*, *remote host*, *remote port*, dan *local port*, setelah itu pembentukan kunci pada *Client* dan *Server*. Sehingga antara *Client* dan *Server* dapat terkoneksi dan dapat bertukar kunci publik.

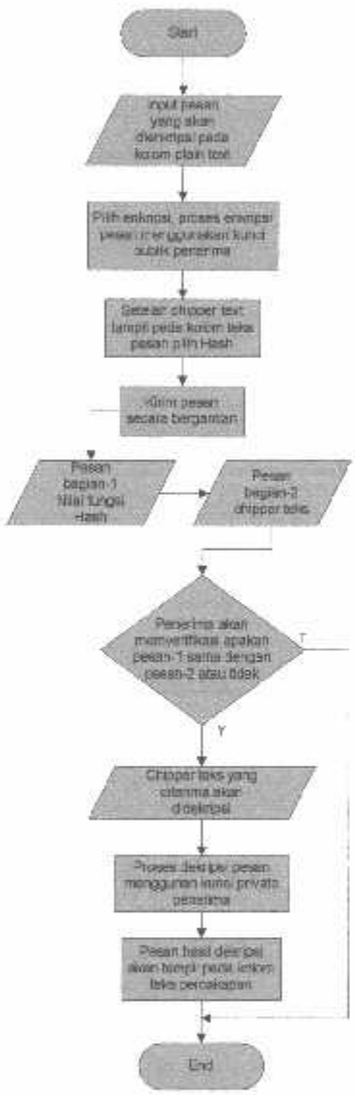
2. *Flowchart* pengiriman pesan tanpa menggunakan *Interlock Protocol*

Gambar 3.10. Flowchart proses pengiriman pesan tanpa *Interlock Protocol*

Pada gambar 3.10 *Flowchart* menggambarkan proses pengiriman pesan antara *Client* dan *Server* tanpa menggunakan *Interlock Protocol*, dimana pesan yang akan dikirim ke penerima dienkripsi menggunakan kunci

publik penerima, setelah *chipper* teks sampai ke penerima akan didekripsi menggunakan kunci privat yang dimilikinya.

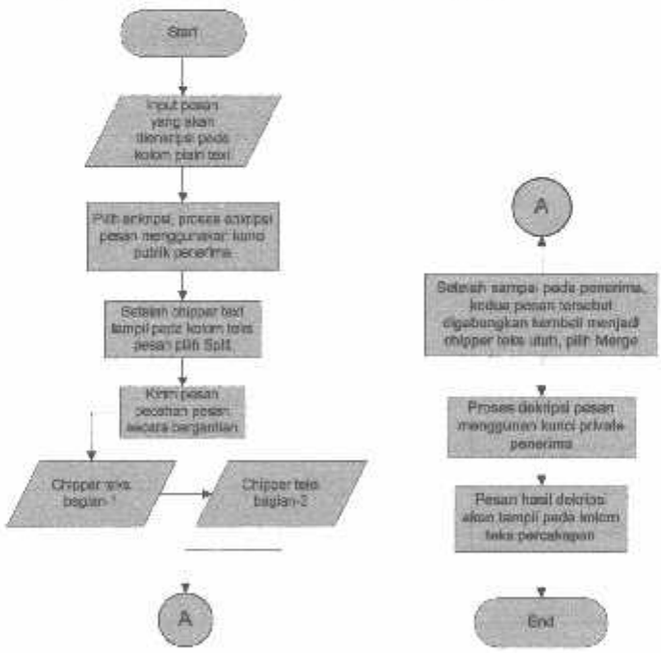
3. Flowchart pengiriman menggunakan *Interlock Protocol* memakai fungsi *One Way Hash SHA-1*



Gambar 3.11. Flowchart proses pengiriman pesan menggunakan *Interlock Protocol* memakai fungsi *One Way Hash SHA-1*

Pada gambar 3.11 *Flowchart* menggambarkan proses pengiriman pesan antara *Client* dan *Server* menggunakan *Interlock Protocol* memakai fungsi *One Way Hash SHA-1*, dimana pesan yang akan dikirim ke penerima akan dienkripsi menggunakan kunci publik penerima setelah itu *chipper* teks hasil enkripsi akan di *digest* menggunakan fungsi *Hash SHA-1*. Pengirim akan mengirim dua pesan, yang pertama adalah nilai hasil fungi *Hash* dari *chipper* teks dan yang kedua yaitu *chipper* teks itu sendiri. Setelah pesan nilai hasil fungsi *Hash* diterima oleh penerima, maka pesan tersebut akan disimpan dan jika pesan kedua diterima, pesan tersebut akan di *digest* dengan fusngi *Hash* oleh penerima dan hasilnya akan dibandingkan dengan pesan pertama, jika hasilnya sama maka pesan akan didekripsi menggunakan kunci privat penerima.

4. Flowchart pengiriman menggunakan *Interlock Protocol* dengan membagi *chipper* teks menjadi 2 bagian yang sama besar.



Gambar 3.12. Flowchart proses pengiriman pesan menggunakan *Interlock Protocol* dengan membagi *chipper* teks menjadi 2 bagian yang sama besar.

Pada gambar 3.12 *Flowchart* menggambarkan proses pengiriman pesan antara *Client* dan *Server* menggunakan *Interlock Protocol* dengan membagi *chipper* teks menjadi 2 bagian, dimana pesan yang akan dikirim ke penerima akan dienkripsi menggunakan kunci publik penerima setelah itu *chipper* teks hasil enkripsi akan di pecah menjadi 2 bagian yang sama besar. Pengirim akan mengirim dua bagian *chipper* teks secara bergantian. Setelah kedua bagian *chipper* teks diterima, maka *chipper* teks tersebut akan digabungkan kembali menjadi *chipper* teks utuh dan hasilnya akan akan didekripsi menggunakan kunci privat penerima.

BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi sistem

Implementasi dilakukan dengan menerapkan hasil desain yang telah dibuat kedalam bahasa pemrograman *(coding) Microsoft Visual Basic* 6.0, schingga algoritma-algoritma yang telah dibuat dapat dimengerti oleh mesin dan menghasilkan keluaran seperti apa yang diharapkan. Berikut ini adalah perlengkapan dan kebutuhan yang digunakan dalam implementasi sistem :

Tabel 4.1 Spesifikasi Perlengkapan Implementasi

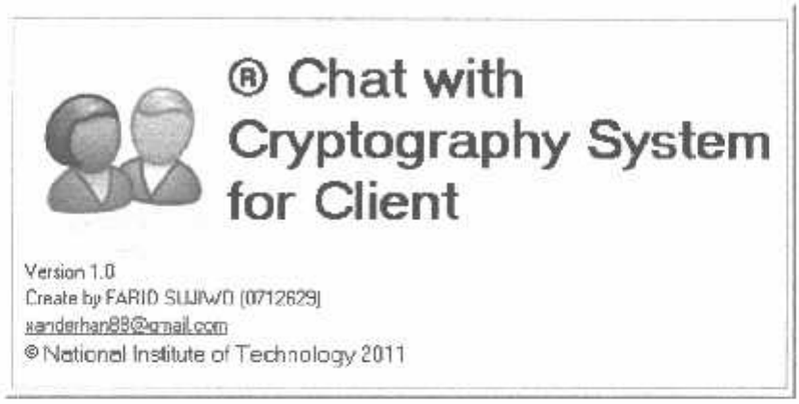
No	Perlengkapan	Spesifikasi	Keterangan
1	Software PC	Sistem Operasi	Windows XP
2	Personal Komputer	Processor	P4. 3.0 Ghz
		Memory	512 Mb DDR1
		Harddisk	80 GB
3	Kabel Jaringan	UTP (crossover)	Konektor RJ-45

Dalam implementasi tersebut kami memakai dua buah *personal computer* yang dihubungkan menggunakan kabel jaringan secara langsung (*point to point*).

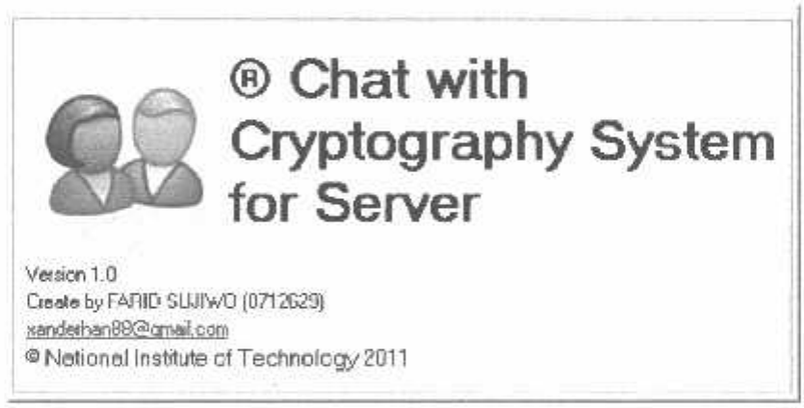
4.2 Aplikasi Enkripsi pesan menggunakan metode *Interlock Protocol*

4.2.1 Tampilan Aplikasi Kriptografi untuk enkripsi pesan pada *client* dan *server*

1. *Form Splash screen*



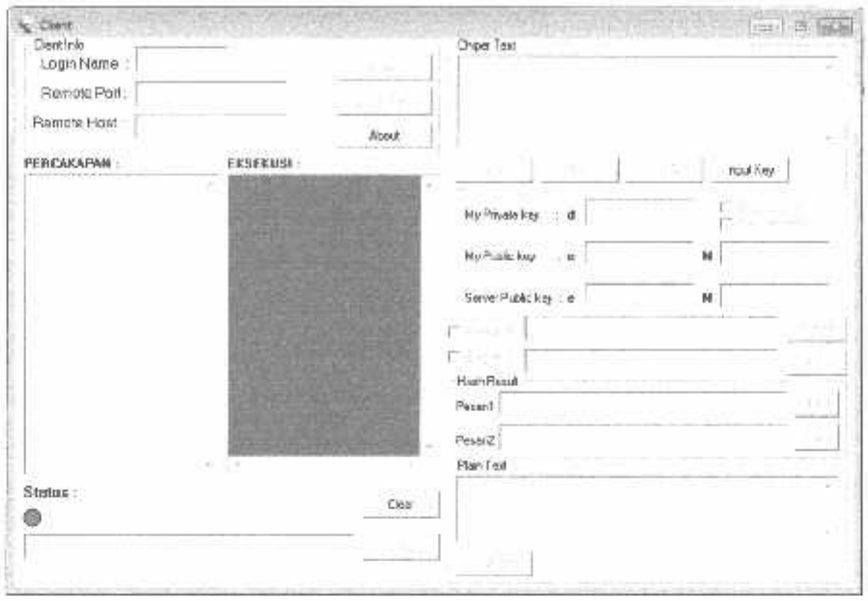
Gambar 4.1. *Form splash screen client*



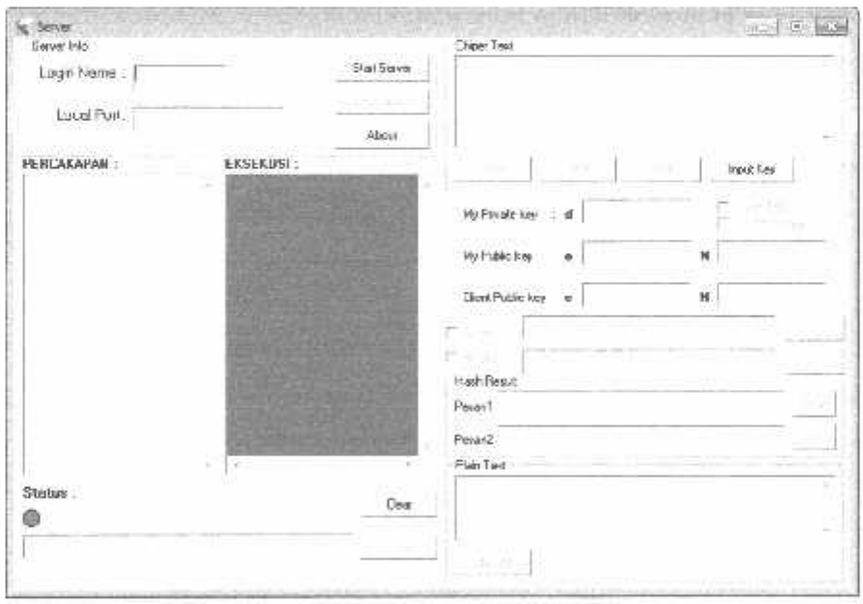
Gambar 4.2. *Form splash screen server*

Pada gambar 4.1 dan gambar 4.2 *form splash screen* adalah tampilan yang pertama kali muncul pada saat aplikasi dijalankan, yang berisi nama aplikasi dan identitas penyusun aplikasi.

2. *Form* utama



Gambar 4.3. *Form* utama *client*

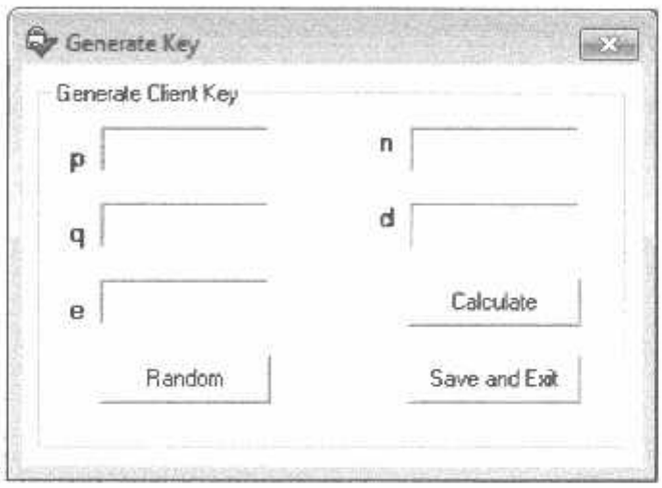


Gambar 4.4. *Form* utama *server*

Dari gambar 4.3 dan 4.4 *Form* utama *Server* serta *Client* bisa kita lihat terdapat beberapa tombol-tombol perintah dan kolom-kolom teks yang digunakan untuk menyambung koneksi antara *Server* dan

Client, tombol input key untuk membuat kunci pada *Client* dan *Server*, serta pilihan-pilihan untuk mengaktifkan *Interlock Protocol*.

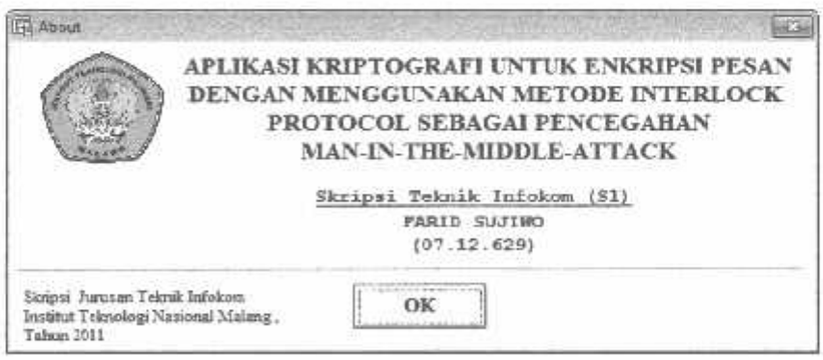
3. *Form* Generate Key



Gambar 4.5. *Form Generate Key*

Pada gambar 4.5 *form Generate Key* terdiri dari kolom-kolom yang nantinya digunakan untuk mengisi, menghitung dan menyimpan nilai-nilai pembentuk kunci publik dan kunci privat yang digunakan pada proses enkripsi dan dekripsi pesan.

4. *Form About*



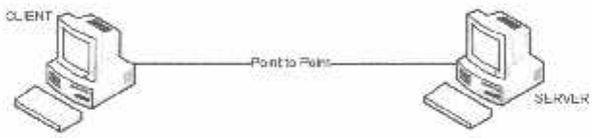
Gambar 4.6. *Form About*

Pada gambar 4.6 *form about* adalah tampilan yang berisi tampilan judul skripsi, nama dan NIM mahasiswa penyusun skripsi dan nama Universitas.

4.3 Pengujian Sistem

Pengujian ini dilakukan untuk mengetahui kinerja aplikasi Kriptografi untuk enkripsi pesan beserta opsi-opsi pendukungnya. Dalam pengujian ini dilakukan beberapa percobaan diantaranya pembentukan kunci secara acak, koneksi antara *Client* dan *Server*, proses enkripsi, dekripsi, *digest* message, dan pembagian *chipper* teks.

4.3.1 Pengujian menggunakan jaringan lokal secara point to point.



Gambar 4.7. Desain Pengujian LAN

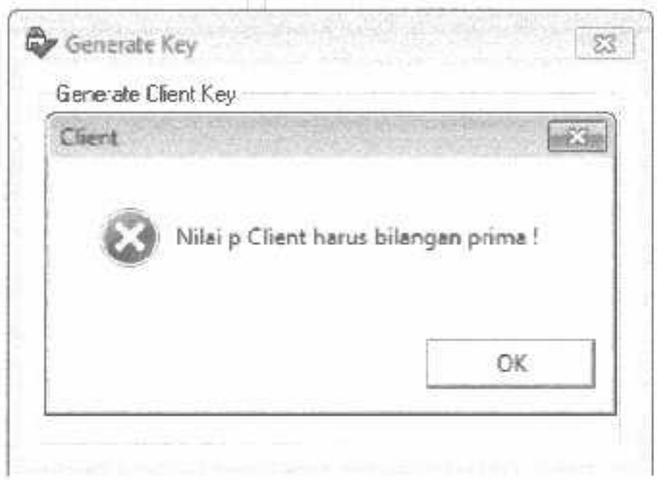
Pada pengujian kali ini akan dilakukan koneksi antara *Client* dan *Server* yang telah terhubung secara point to point.

4.3.2. Pengujian Pada proses pembentukan kunci

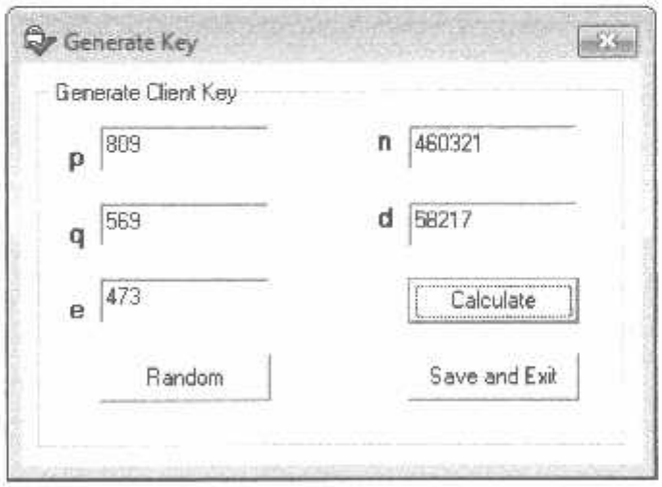
Pada pengujian penmbentukan kunci pada form generate key kita bisa menginputkan sendiri nilai-nilai untuk proses pembentukan kunci, tapi yang perlu diingat adalah nilai-nilai tersebut harus bilangan prima dan relatif antar bilangan yang dibutuhkan dalam algoritma RSA pada proses enkripsi dan dekripsi, maka pada proses pembentukan kunci dibutuhkan algoritma pendukung, diantaranya yaitu Metode *Rabin-Miller* untuk menghasilkan bilangan prima, Algoritma GCD untuk mengecek prima relatif antar bilangan, Algoritma *Fast Exponentiation* untuk menghitung perpangkatan

modulo bilangan besar, *Generator Geffe* untuk membangkitkan bilangan acak semu.

Jika kita salah memasukkan nilai maka yang terjadi adalah proses pembentukan kunci tersebut akan gagal, karena nilai yang dimasukkan bukanlah bilangan prima.



Gambar 4.8. Pengujian pembentukan kunci tanpa pengacakan bilangan prima



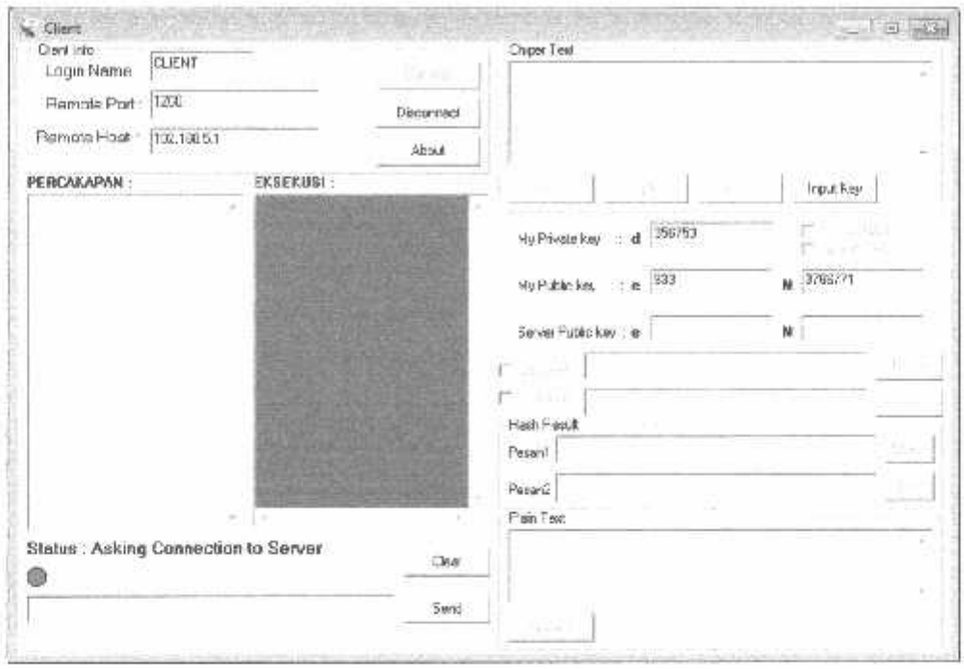
Gambar 4.9. Pengujian pembentukan kunci dengan pengacakan bilangan prima

Dari gambar 4.8 dan gambar 4.9 bisa dilihat bahwa jika akan menginputkan sendiri bilangan yang akan dibentuk menjadi kunci tidak bisa sembarang bilangan, bilangan yang bisa dibentuk menjadi kunci adalah bilangan prima dan prima relatif antar bilangan tersebut. Pada pengujian pembentukan kunci secara acak atau *random*, deret bilangan yang keluar antara 3 sampai 5 deret bilangan yang dihasilkan.

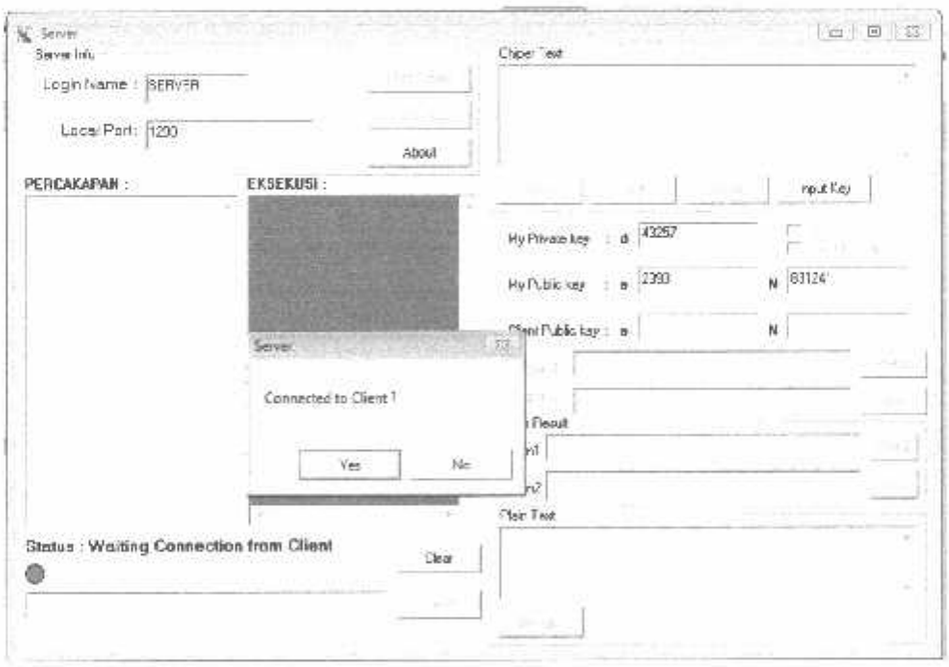
4.3.3. Pengujian Pada proses koneksi antara *Client* dan *Server*

Pada pengujian koneksi *Client* dengan *Server*, terlebih dahulu kita harus menginputkan username, remote port pada *Client* yang harus sama dengan local port pada *Server*, setelah itu remote host pada *Client*, yang isinya adalah alamat IP yang dimiliki *Server*. Jika semua sudah terisi dengan benar maka selanjutnya adalah *Server* mulai membuka koneksi, diikuti dengan *Client* meminta koneksi pada *Server*. Setelah *Server* menerima permintaan koneksi dari *Client*, maka yang selanjutnya terjadi proses pertukaran kunci publik antara *Client* dan *Server*.

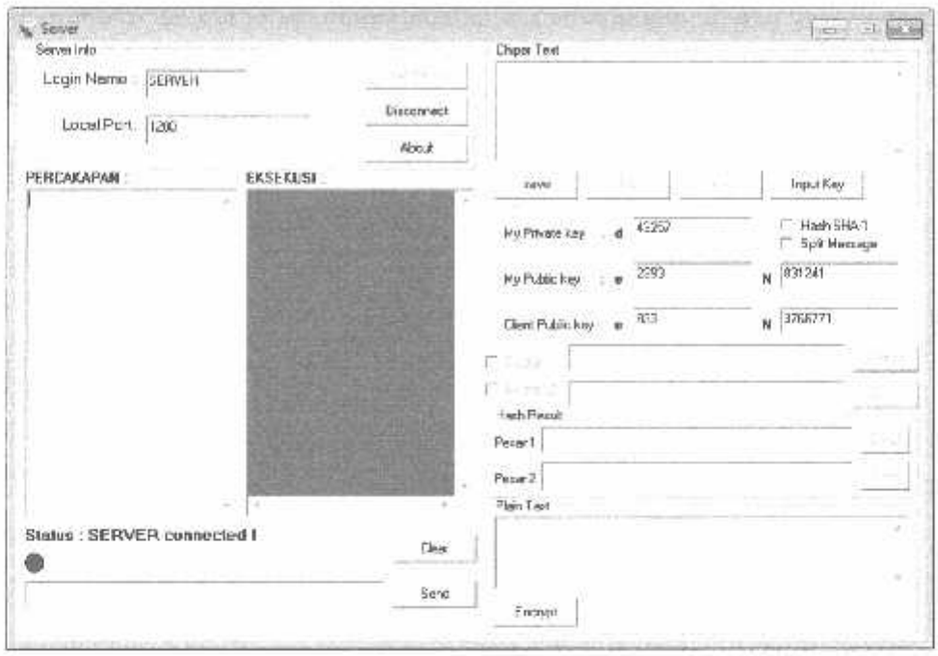
Setelah itu *Client* dan *Server* sudah selesai terkoneksi, ditandai dengan status dari *Client* dan *Server* connected dan indikator dari warna merah berubah menjadi warna hijau.



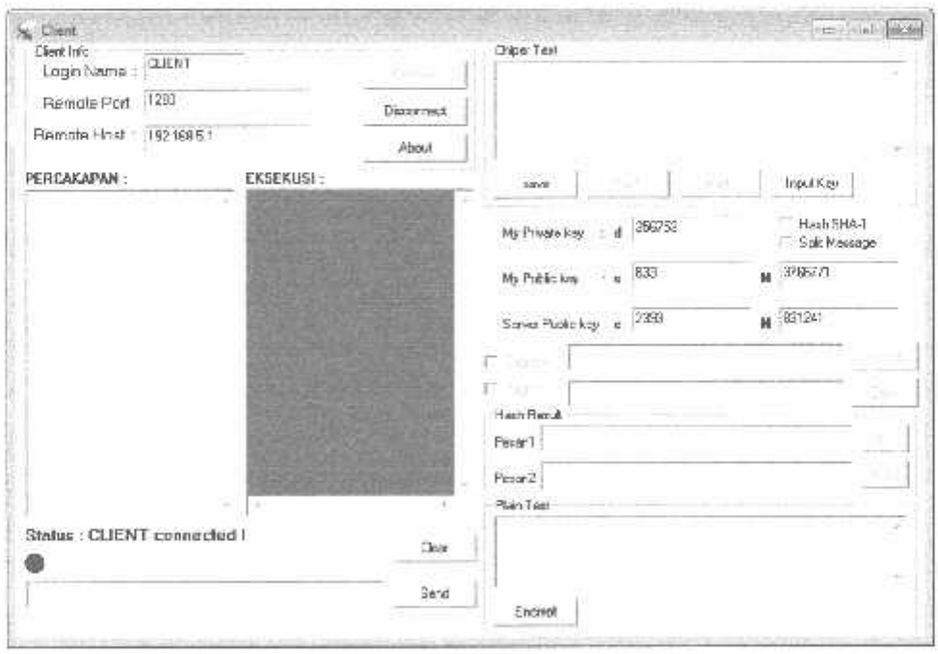
Gambar 4.10. Pengujian koneksi dari *Client* pada *Server*



Gambar 4.11. Permintaan koneksi dari *Client* pada *Server*



Gambar 4.12. *Server* yang telah terkoneksi dengan *Client*

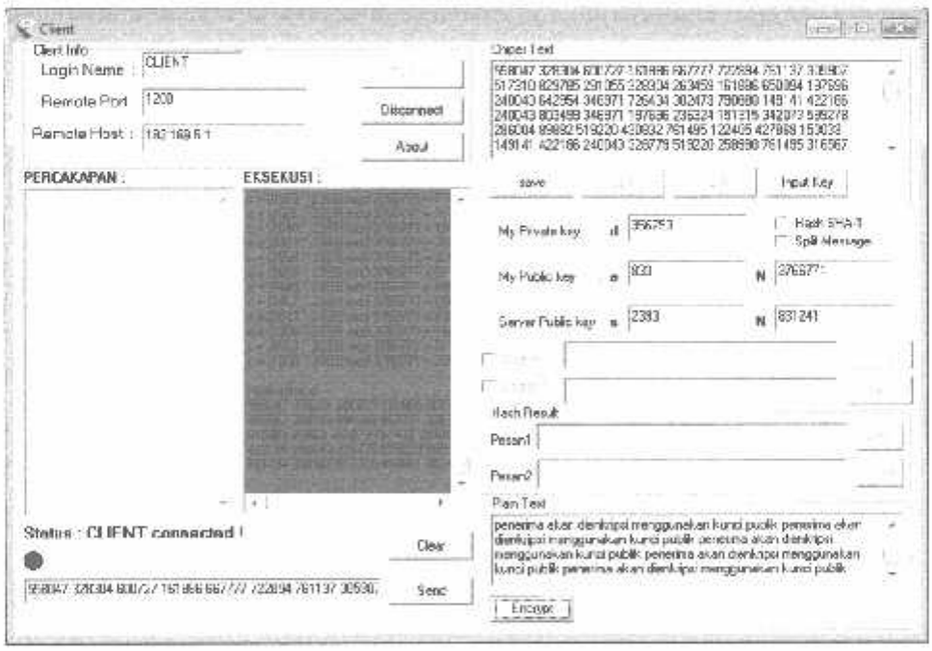


Gambar 4.13. *Client* yang telah terkoneksi dengan *Server*

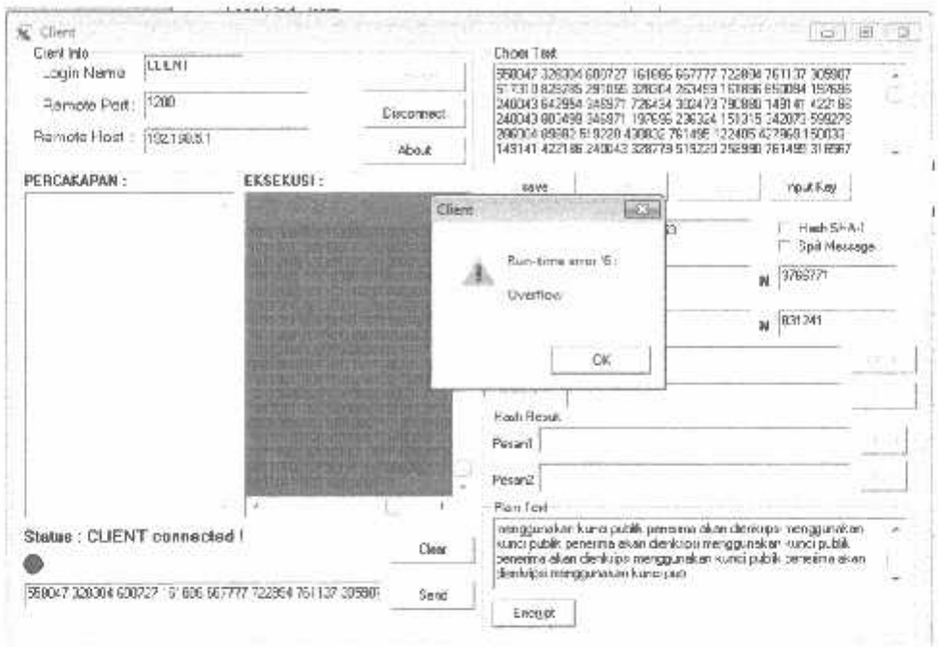
Dari gambar 4.10, gambar 4.11, gambar 4.12 dan gambar 4.13 bisa dilihat sebelum terjadi koneksi antara *Client* dan *Server* indikator status masih berwarna merah, setelah permintaan koneksi disetujui maka indikator status akan berwarna hijau. Berikutnya secara otomatis *Client* dan *Server* akan saling bertukar kunci publik yang masuk pada kolomnya masing-masing.

4.3.4. Pengujian Pada proses enkripsi pesan

Pada pengujian proses enkripsi pesan ini akan dilakukan pengujian seberapa cepat prose enkripsi pesan yang dapat dilakukan dan berapa karakter yang dapat di enkripsi untuk satu kali pengiriman pesan. Untuk melakukannya kita mencoba menginputkan pada plain teks 1000 karakter, setelah itu kita akan mengenkripsikannya dengan memilih tombol *Encrypt*.



Gambar 4.14. Pengujian enkripsi pesan dengan 1000 karakter

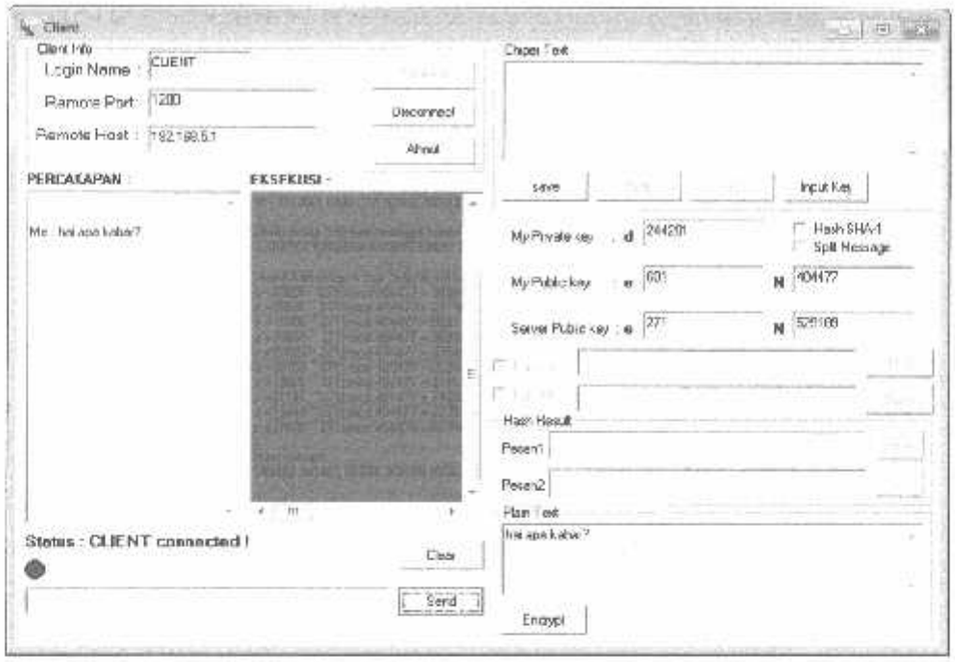


Gambar 4.15. Pengujian enkripsi pesan dengan 4096 karakter

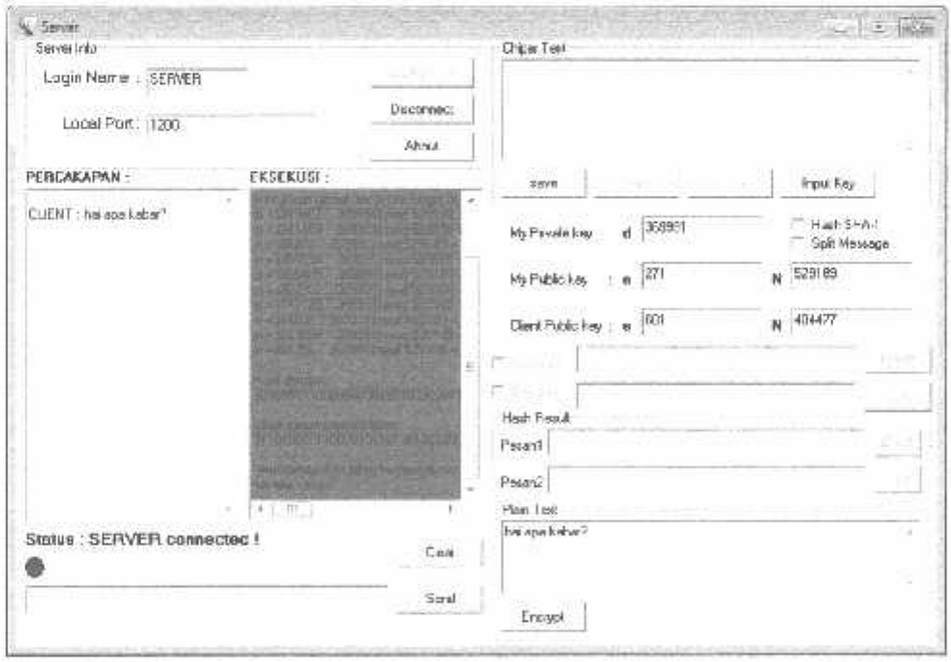
Dari gambar 4.14 proses enkripsi yang dilakukan menggunakan plain teks sebanyak 1000 karakter membutuhkan waktu 2 detik untuk komputer dapat menyelesaikannya. Dan pada gambar 4.15 saat pengujian menggunakan karakter sebanyak 4096 karakter, proses *error* karena data yang diproses *over flow*.

4.3.5. Pengujian Pada proses pengiriman pesan tanpa *Interlock Protocol*

Pada pengujian proses pengiriman pesan ini akan dilakukan proses pengiriman, pengenkripsian dan dekripsi pesan yang akan dikirimkan dan diterima antara *Client* dan *Server* tanpa menggunakan *Interlock Protocol*.



Gambar 4.16. Pengujian pengiriman pesan pada *Client*

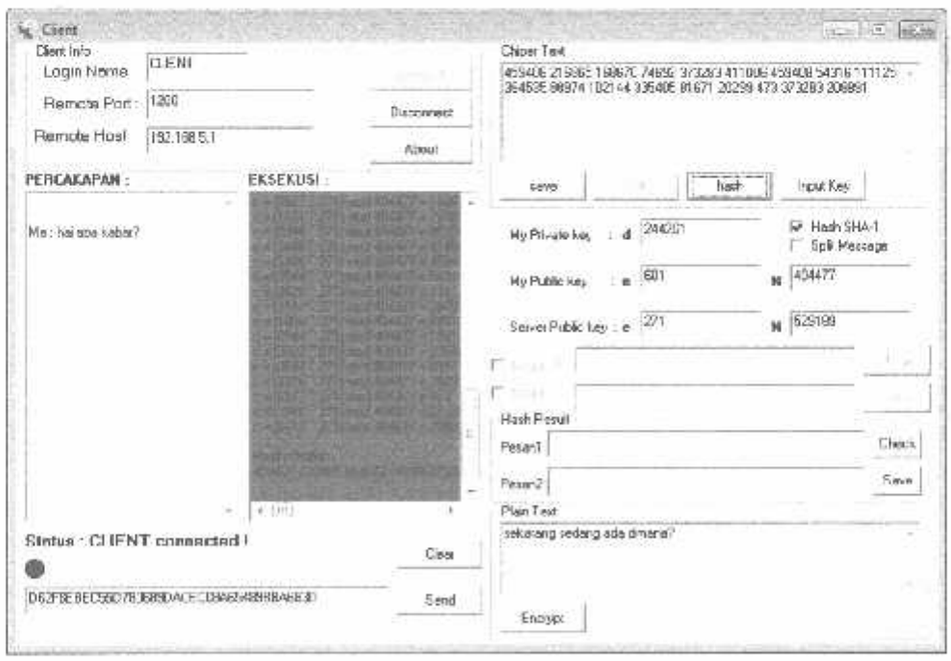


Gambar 4.17. Pengujian penerimaan pesan pada *Server*

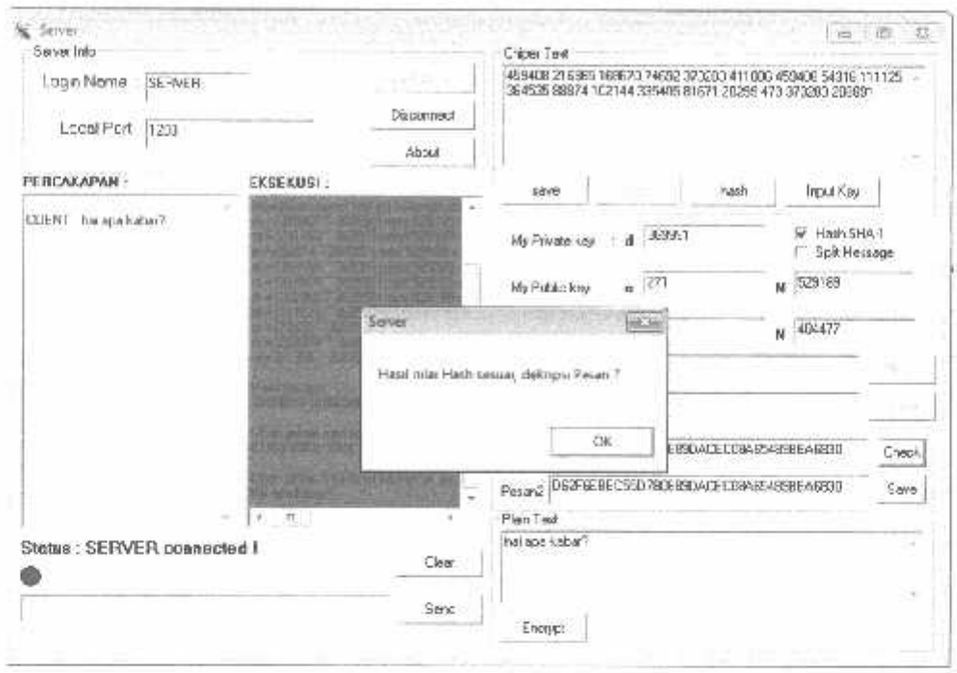
Dari gambar 4.16 dapat dilihat proses enkripsi pesan yang dilakukan pada *Client* menggunakan kunci publik *Server*, setelah itu pesan berupa *chipper* teks dikirim dan pada gambar 4.17 begitu sampai kepada penerima yaitu *Server* akan didekripsi menggunakan kunci privat *Server*.

4.3.6. Pengujian Pada proses pengiriman pesan dengan fungsi *One Way Hash SHA-1*

Pada pengujian proses pengiriman pesan ini akan dilakukan proses pengiriman pesan antara *Client* dan *Server* menggunakan *Interlock Protocol* memakai fungsi *One Way Hash SHA-1*, dimana pesan yang akan dikirim ke penerima akan dienkripsi menggunakan kunci publik penerima setelah itu *chipper* teks hasil enkripsi akan di *digest* menggunakan fungsi *Hash SHA-1*. Pengirim akan mengirim dua pesan, yang pertama adalah hasil *digest* dari *chipper* teks dan yang kedua yaitu *chipper* teks itu sendiri. Setelah pesan hasil *digest* diterima oleh penerima, maka pesan tersebut akan disimpan dan jika pesan kedua diterima pesan tersebut akan di *digest* oleh penerima dan hasilnya akan dibandingkan dengan pesan pertama.



Gambar 4.18. Pengujian pengiriman pesan pada *client* menggunakan fungi *Hash*

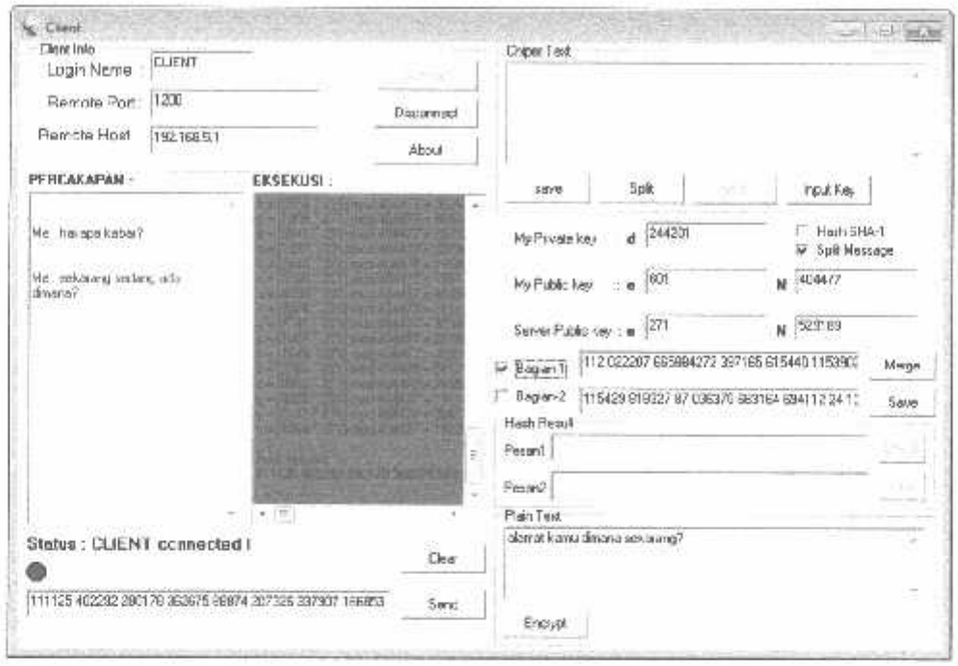


Gambar 4.19. Pengujian penerimaan pesan pada *server* menggunakan fungi *Hash*

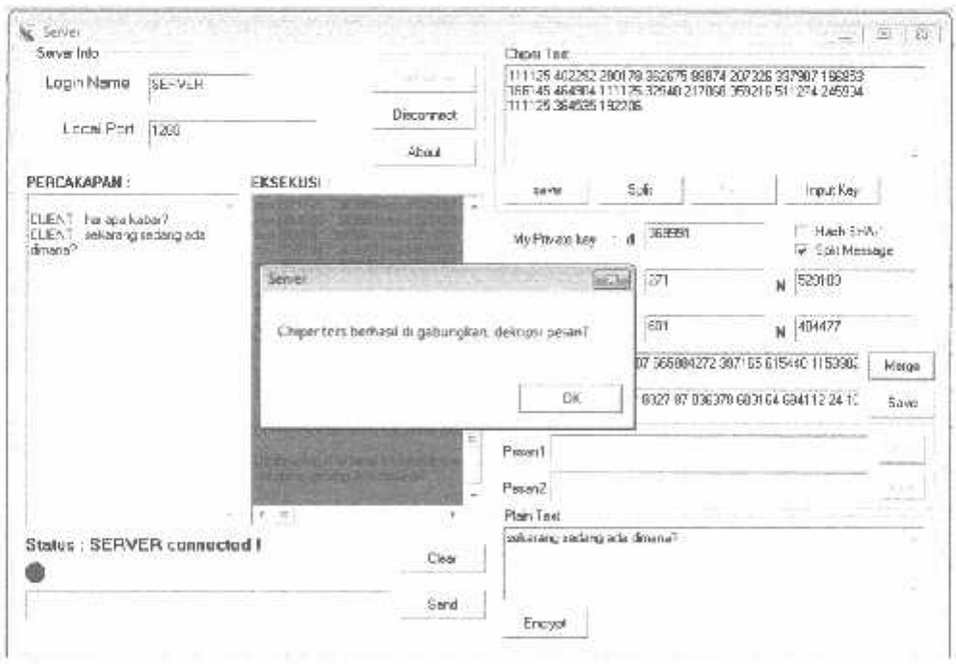
Dari gambar 4.18 dapat dilihat proses enkripsi pesan dan pengiriman pesan secara berurutan, pesan pertama yang dikirim adalah pesan nilai *Hash* dan pesan kedua adalah *chipper* teks. Setelah itu pada gambar 4.19 oleh penerima akan dilakukan pengecekan, jika hasil nilai *Hash* pesan kedua dengan pesan pertama sama maka pesan akan didekripsi.

4.3.7. Pengujian Pada proses pengiriman pesan dengan membagi *chipper* teks.

Pada pengujian proses pengiriman pesan ini akan dilakukan proses pengiriman pesan antara *Client* dan *Server* menggunakan *Interlock Protocol* dengan membagi *chipper* teks menjadi 2 bagian, dimana pesan yang akan dikirim ke penerima akan dienkripsi menggunakan kunci publik penerima setelah itu *chipper* teks hasil enkripsi akan di pecah menjadi 2 bagian yang sama besar. Pengirim akan mengirim dua bagian *chipper* teks secara bergantian.



Gambar 4.20. Pengujian pengiriman pesan pada *Client* dengan membagi pesan menjadi 2 bagian



Gambar 4.21. Pengujian penerimaan pesan pada *Server* dengan menggabungkan 2 bagian *chipper* teks

Dari gambar 4.20 dapat dilihat proses enkripsi pesan dan pengiriman pesan secara berurutan, pesan pertama adalah *chipper* teks bagian-1 dan selanjutnya *chipper* teks bagian-2 yang selajutnya pada gambar 4.21 pesan akan digabungkan kedua bagian tersebut oleh penerima sehingga menjadi *chipper* teks yang utuh dan didekripsikan.

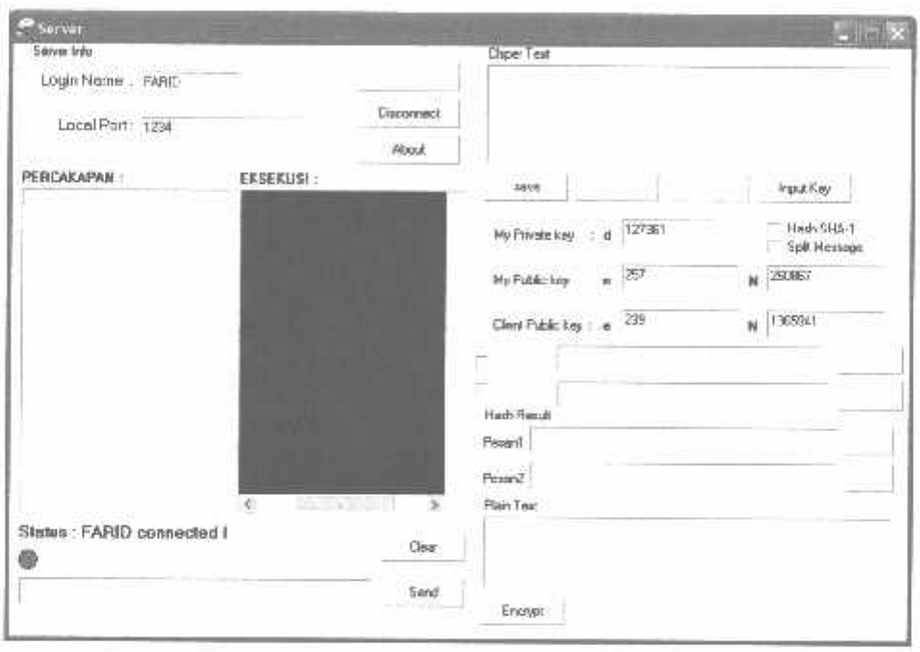
Berikut ini adalah tabel hasil pengujian proses enkripsi pesan pada aplikasi.

Tabel 4.2 Hasil pengujian proses enkripsi

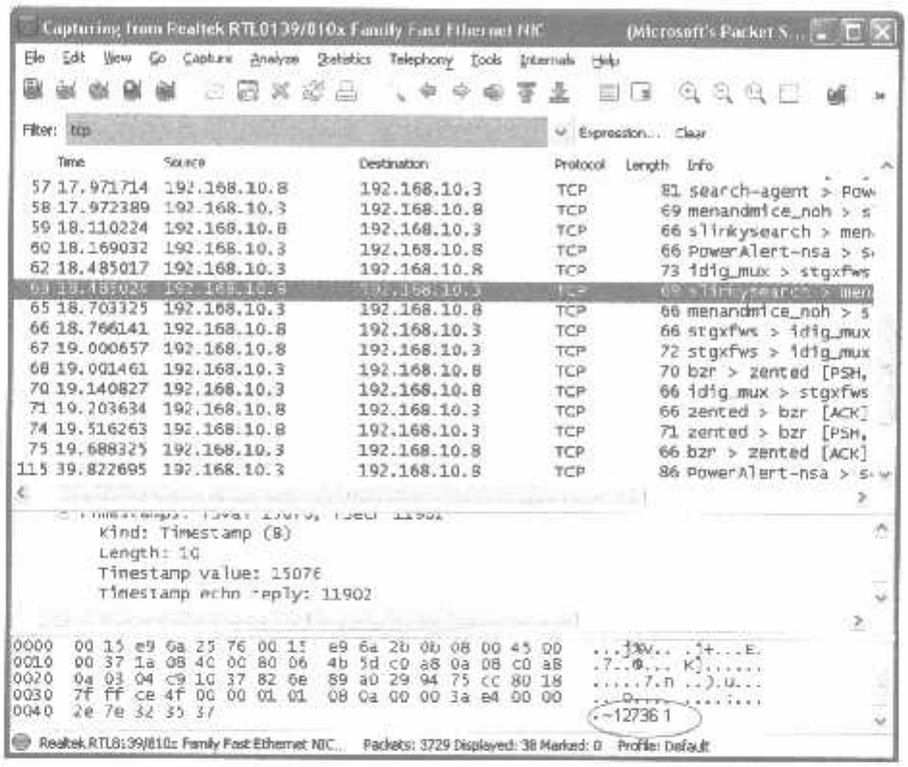
No	Jumlah Karakter	Waktu yang dibutuhkan	Keterangan
1	1000	2 detik	sukses
2	2000	4 detik	sukses
3	3000	6 detik	sukses
4	4000	8 detik	sukses
5	4095	8 detik	sukses
6	4096	0	*Data Over Flow*

4.3.8. Pengujian Jika Data disadap

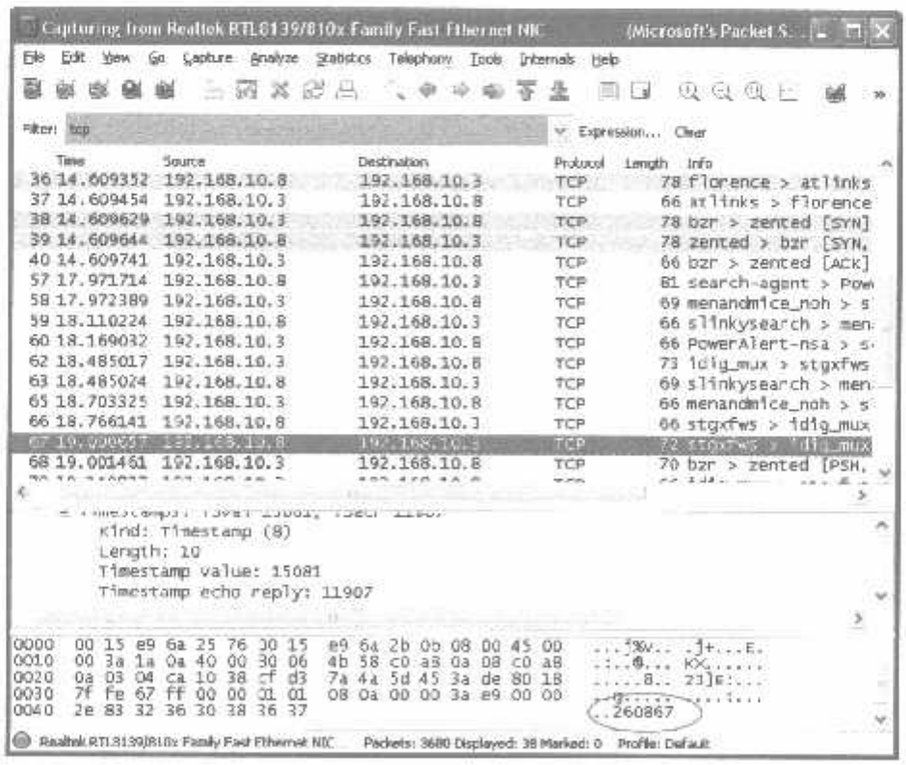
Pada pengujian ini akan dilakukan penyadapan pada data yang dikirim oleh *Client* dan *Server*, apakah data tersebut aman atau tidak jika menggunakan metode *Interlock Protocol*. Pada proses penyadapan ini menggunakan 3 komputer dimana satu komputer bekerja sebagai penyadap dengan menggunakan software Wireshark v1.6. Dengan software tersebut komputer penyadap akan dapat memantau paket-paket yang dikirim dan diterima baik dari *Client* maupun *Server*.



Gambar 4.22. Form Server yang akan disadap

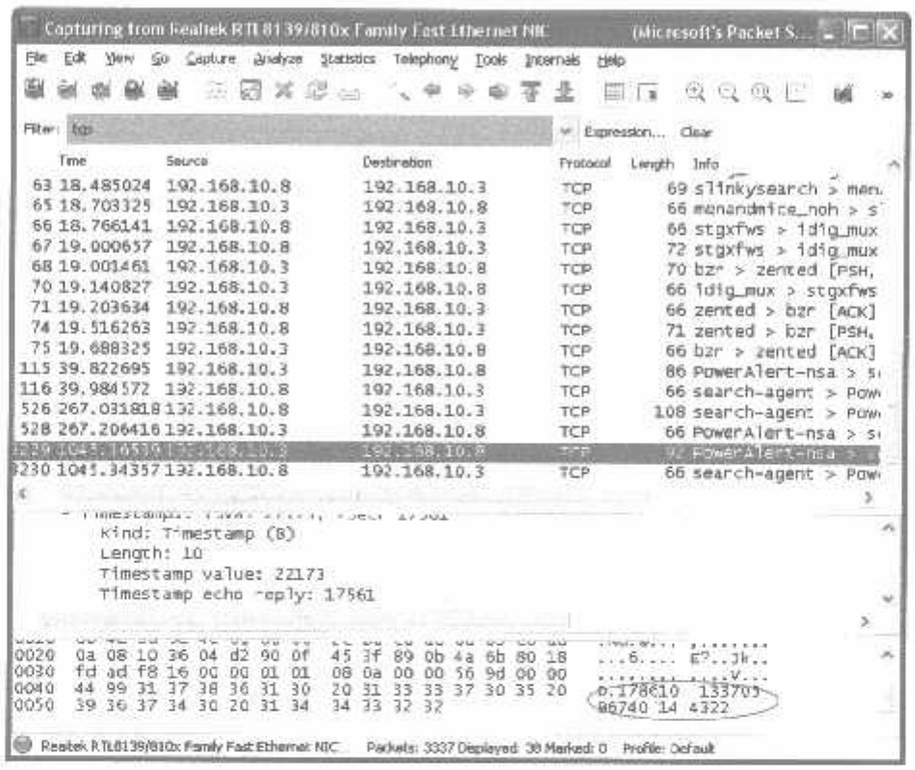


Gambar 4.23. Penyadapan kunci e dari Server

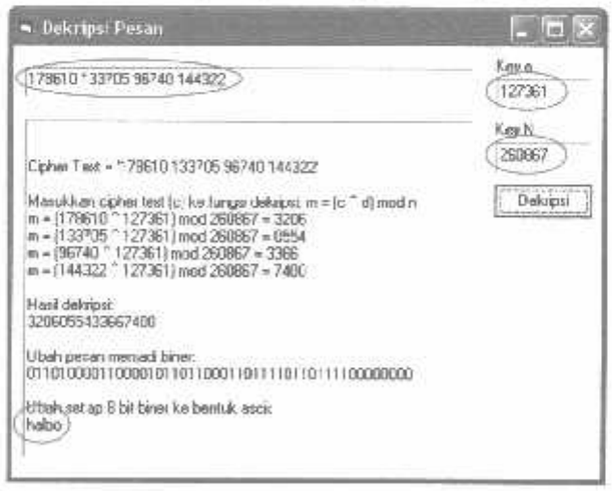


Gambar 4.24. Penyadapan kunci d dari Server

Pada gambar 4.23 dan gambar 4.24 terjadi penyadapan pada kunci yang dikirim oleh *Server* pada *Client*. Kunci tersebut digunakan untuk mendekripsikan *Chipper* teks yang dikirimkan, jadi jika penyadap memiliki kunci tersebut maka penyadap dapat membaca semua pesan yang dikirimkan.



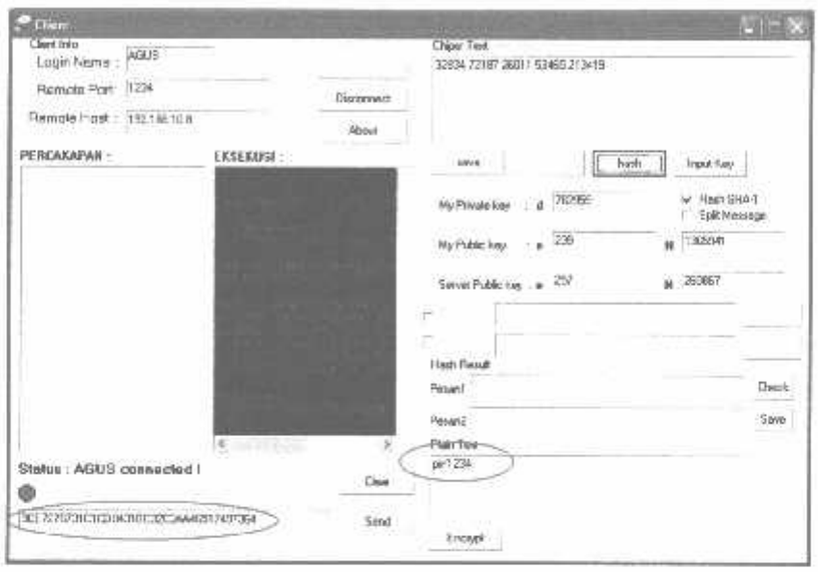
Gambar 4.25. Penyadapan *Chipper* teks yang dikirim



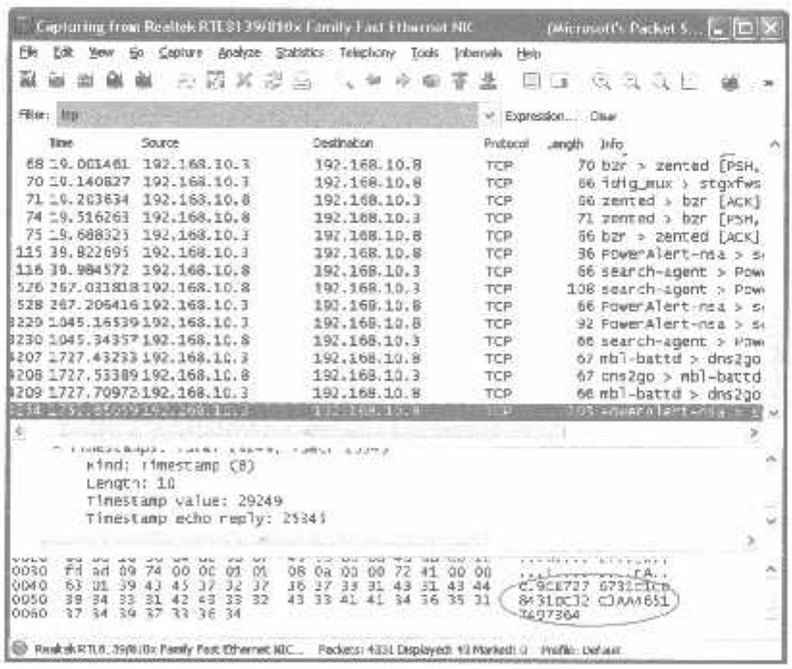
Gambar 4.26. Dekripsi pesan oleh penyadap

4.3.8.1. Pengujian penyadapan menggunakan *Interlock Protocol* dengan fungsi Hash

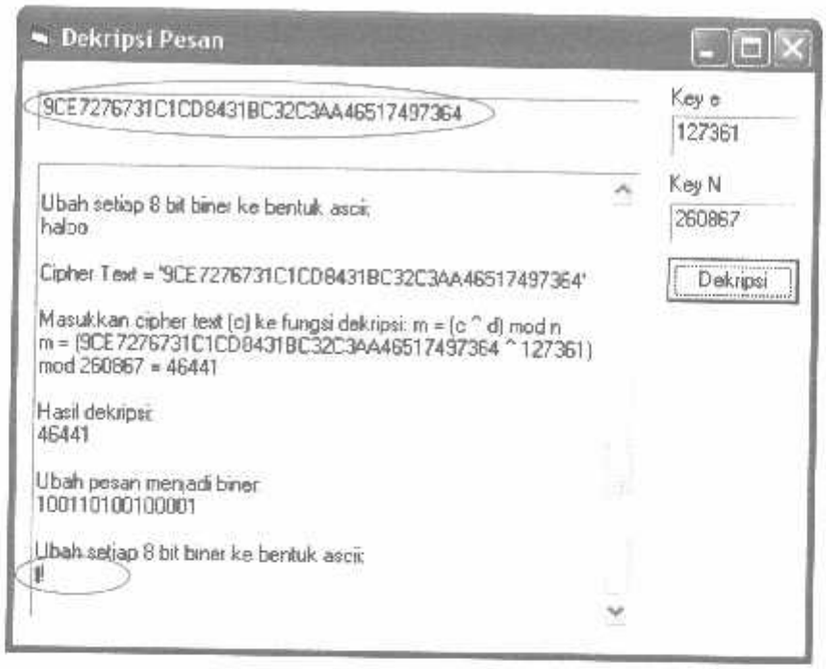
Pada pengujian akan dilakukan penyadapan data, namun kali ini pengiriman pesan pada aplikasi menggunakan metode *Interlock Protocol* dengan fungsi *Hash*.



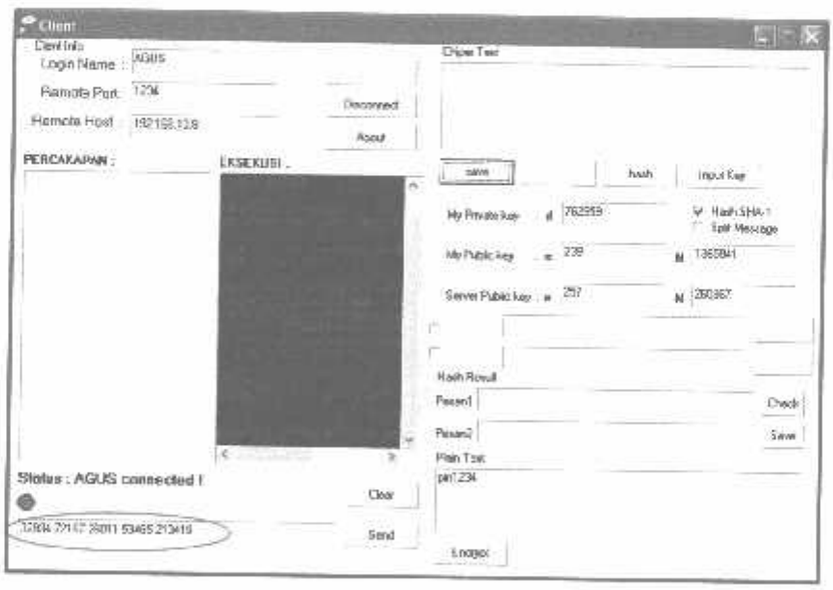
Gambar 4.27. Pengiriman kode *Hash* dari *Client*



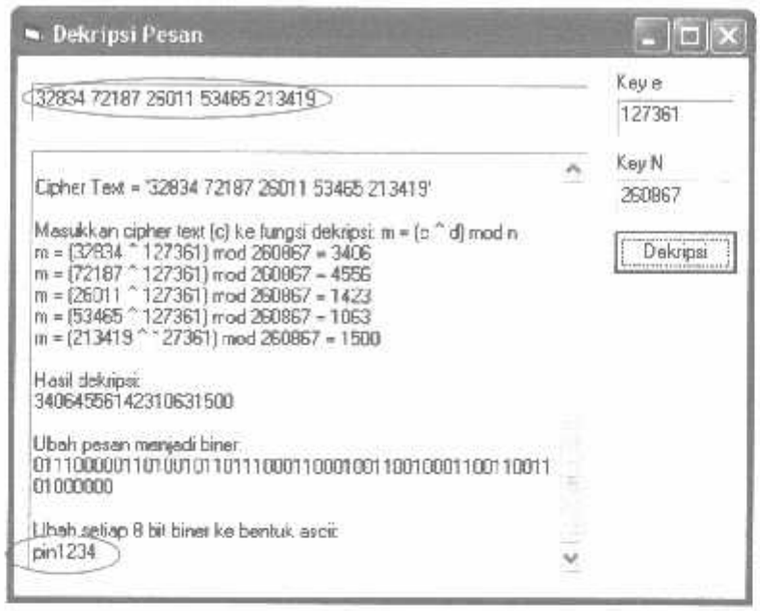
Gambar 4.28. Penyadapan kode *Hash*



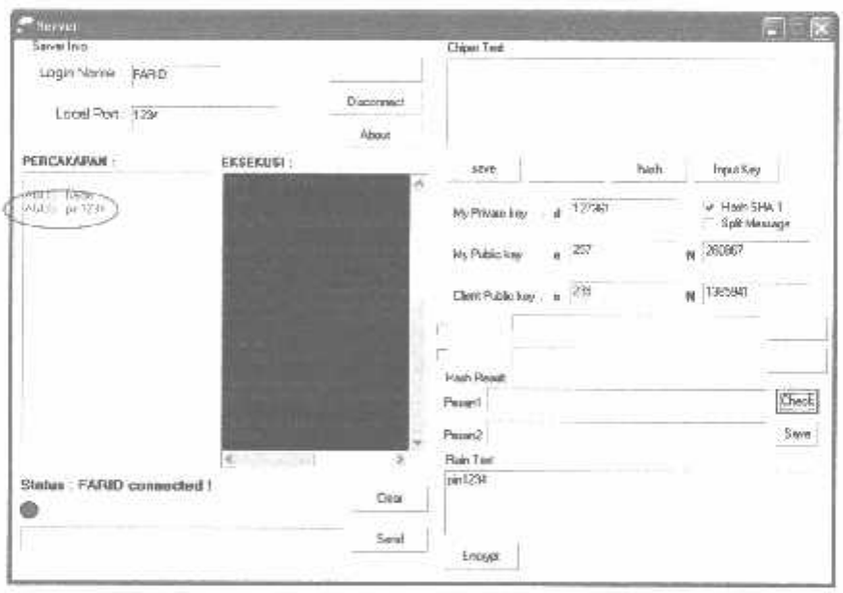
Gambar 4.29. Dekripsi kode *Hash* oleh penyadap



Gambar 4.30. Pengiriman *chipper* teks dari Client



Gambar 4.31. Dekripsi pesan oleh penyadap

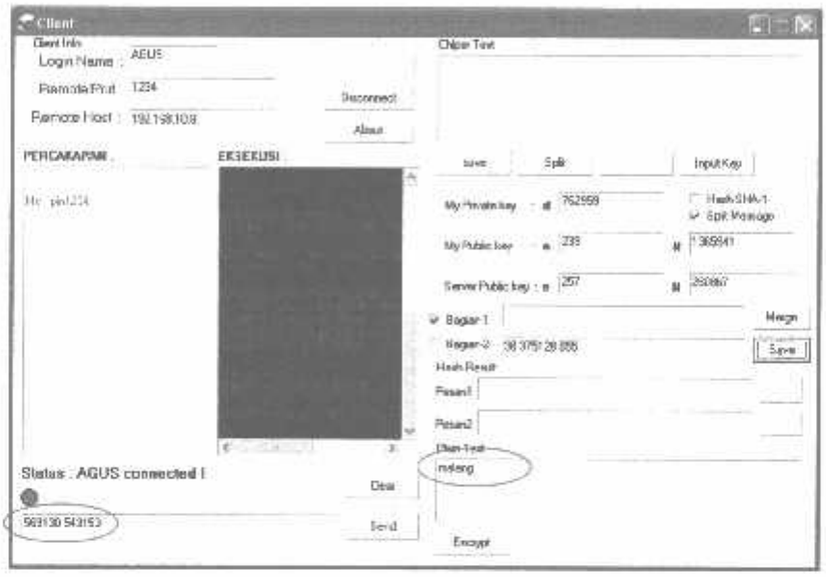


Gambar 4.32. Dekripsi pesan oleh Server

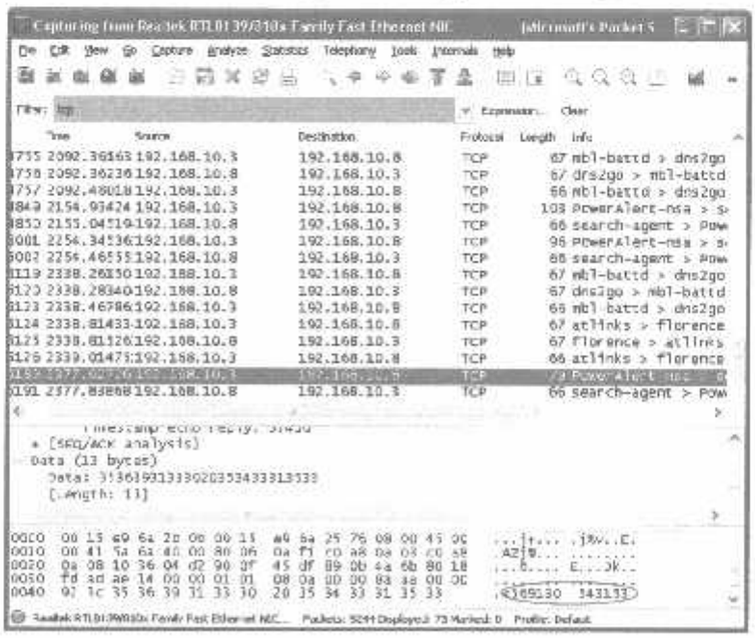
Pada pengujian yang dilakukan penyadap tidak dapat mendekripsikan pesan pertama yaitu kode *Hash*, dikarenakan kode tersebut bukanlah sebuah *chipper* teks. Penyadap dapat mendekripsikan pesan kedua yaitu *chipper* teks, akan tetapi tidak dapat mengubah isi dari pesan tersebut, karena pesan tersebut sudah terlebih dahulu dilakukan fungsi *Hash* oleh pengirim.

4.3.8.2. Pengujian penyadapan menggunakan *Interlock Protocol* dengan membagi *chipper* teks

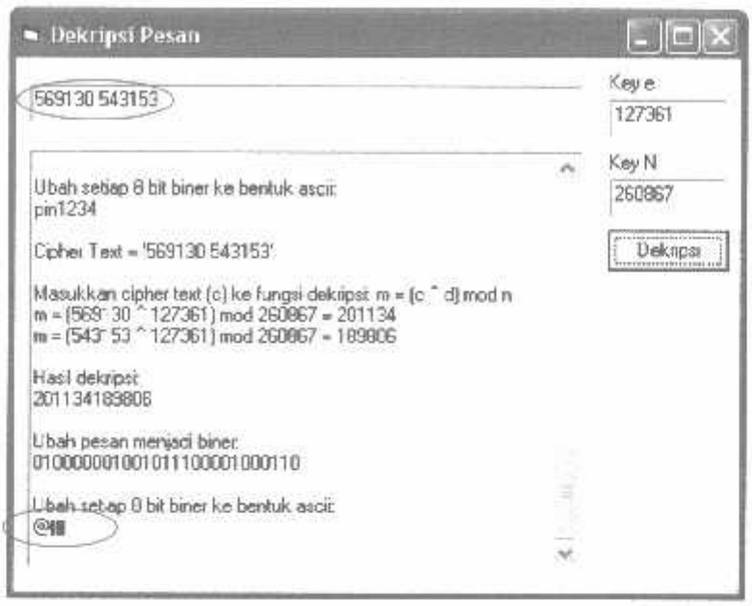
Pada pengujian akan dilakukan penyadapan data, proses pengiriman pesan menggunakan *Interlock Protocol* dengan membagi *chipper* teks menjadi dua bagian yang sama besar dan dikirim secara bergantian.



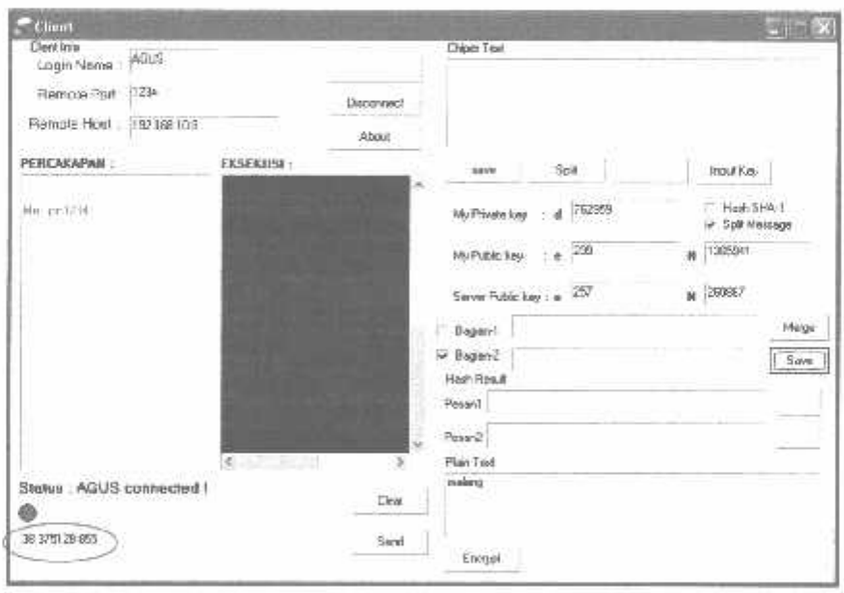
Gambar 4.33. Pengiriman *chipper* teks bagian-1



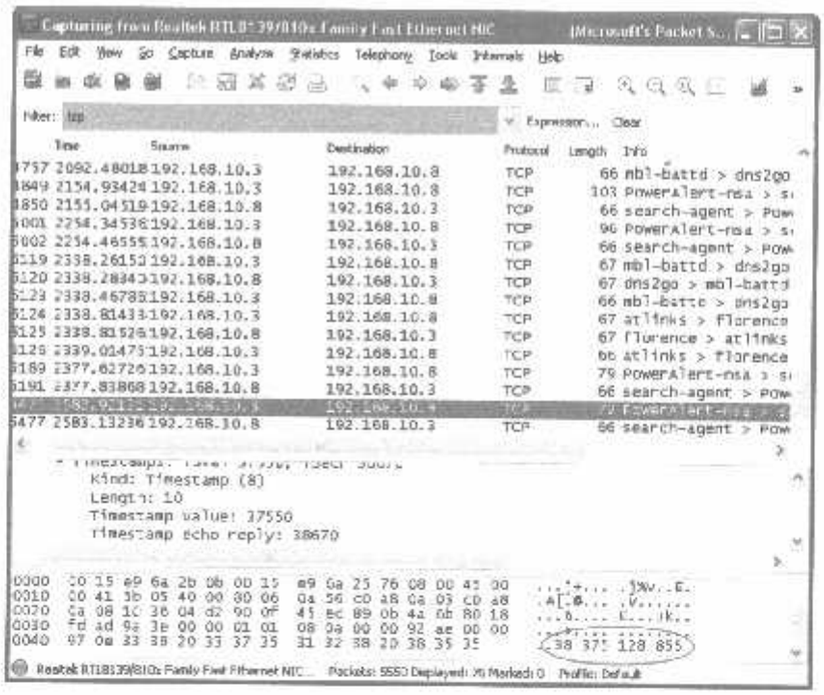
Gambar 4.34. Penyadapan *chipper* teks bagian-1



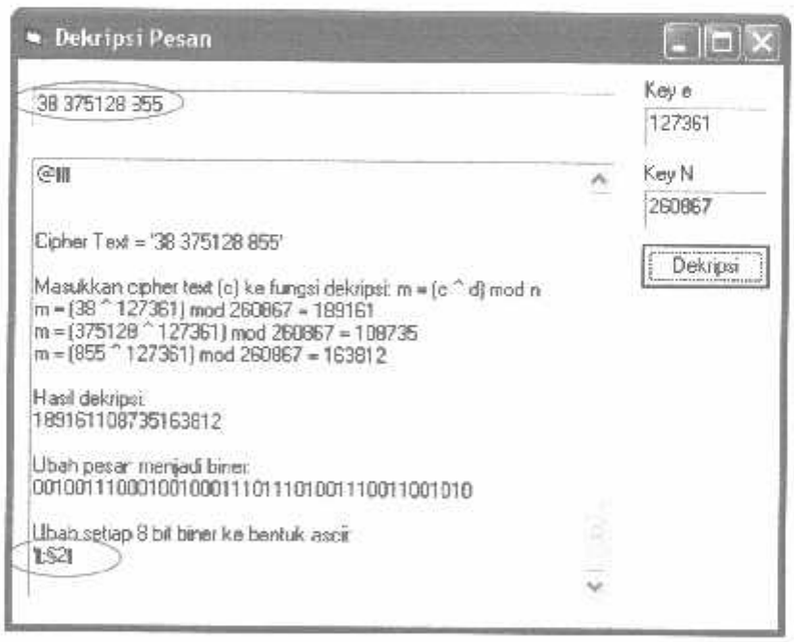
Gambar 4.35. Dekripsi pesan bagian-1 oleh penyadap



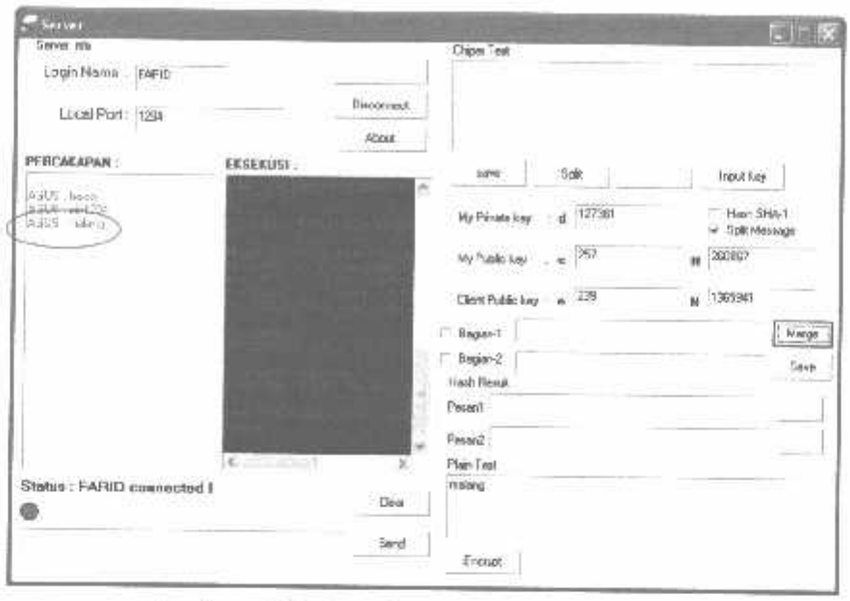
Gambar 4.36. Pengiriman *chipper* teks bagian-2



Gambar 4.37. Penyadapan pesan bagian-2



Gambar 4.38. Dekripsi pesan bagian-2 oleh penyadap



Gambar 4.39. Penerimaan pesan pada *Server*

Pada pengujian yang dilakukan penyadap tidak dapat mendekripsikan pesan bagian-1 dan pesan bagian-2, karena pesan yang dikirimkan bukan *chipper* teks utuh hasil enkripsi melainkan setengah bagian dari *chipper* teks tersebut.

Berikut ini adalah tabel hasil pengujian proses penyadapan pada aplikasi.

Tabel 4.3 Hasil pengujian proses penyadapan

No	Metode	Pesan	Dekripsi	Manipulasi
1	Hash	Kode Hash	Gagal	Gagal
2		*Chipper teks*	Sukses	Gagal
3	Split	Pesan bagian-1	Gagal	Gagal
4		Pesan bagian-2	Gagal	Gagal

BAB V
PENUTUP

5.1 Kesimpulan

Dari pengujian yang dilakukan, diperoleh kesimpulan sebagai berikut :

1. Pencegahan *Man-In-The-Middle-Attack* berhasil dilakukan, pesan yang terkirim dan diterima tidak dapat dibaca dan dimanipulasi pada proses penyadapan.
2. Pada proses pengujian pembentukan kunci didapatkan tidak semua bilangan bisa dijadikan nilai untuk membentuk kunci, hanya bilangan prima yang bisa dibetuk sebagai kunci.
3. Kunci yang terbentuk menggunakan proses random menghasilkan 3 sampai 5 deret panjang bilangan.
4. Pada proses pengujian enkripsi pesan, didapatkan bahwa komputer dapat menyelesaikan proses enkripsi pada aplikasi ini dengan kecepatan 500 karakter/detik.
5. Dari hasil proses enkripsi pesan didapatkan bahwa aplikasi hanya dapat memproses enkripsi pesan dengan jumlah karakter maksimal 4095.

5.2 Saran-saran

Berikut ini adalah saran yang diberikan untuk penegembangan selanjutnya.

1. Untuk pengembangan selanjutnya dapat ditambahkan beberapa fitur pendukung untuk aplikasi *chatting* pada umumnya, seperti *smilely*, *instant message*, dan fitur pendukung lainnya.
2. Enkripsi dengan kriptografi tidak hanya dilakukan sebatas pada pesan saja tetapi dikembangkan enkripsi pada gambar dan file teks.
3. Untuk pengembangan aplikasi *Client* dan *Server* selanjutnya bisa ditambahkan agar dapat digunakan untuk multi*Client*.
4. Ditambahkan penggunaan *database* yang difungsikan untuk penyimpanan hasil percakapan.

DAFTAR PUSTAKA

1. Ario Suryokusumo, ***Microsoft Visual Basic 6.0***, PT. Elex Media Komputindo, Jakarta, 2001.
2. Bruce Schneier, ***Applied Crytography***, *Second Edition*, 1996.
3. Jusuf Kurniawan, **Kriptografi Keamanan *Internet* dan Jaringan Komunikasi**, Penerbit Informatika, Bandung, 2004.
4. Rahadian Hadi, **Pemrograman *Windows API* dengan *Microsoft Visual Basic***, PT. Elex Media Komputindo, Jakarta, 2002.
5. Wardana, **Pembuatan Kontrol *Active X* di *Visual Basic 6.0***, PT. Elex Media Komputindo, Jakarta, 2006
6. Rahmat Putra, ***Inovative Source Code Visual Basic***, Dian Rakyat, Jakarta, 2006
7. Suryanto Thabrani, **Mudah dan Cepat Menguasai Visual Basic**, MediaKita, Surabaya, 2007.
8. William Stallings, ***Cryptography and Network Security***, *Third Edition.*
9. Ridwan,M.B.A. Drs, **Metode & Teknik Penyusunan Proposal Penelitian**, Alfabeta, Bandung, 2009.
10. Sugito Yogi, **Metode Penelitian Metode Percobaan dan Penulisan Karya Ilmiah**, UB Press, Malang, 2009
11. *http://id.wikipedia.com/visual_basic* 23 Oktober 2011, 08:32Am
12. *http://id.wilkipedia.org/wiki/man in the middle attack* 02 Agustus 2011, 06:03Pm
13. *http://www.planetsourcecode.com/vb.classic/* 16 Oktober 2011, 09:45Am

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting) Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI

FAKULTAS TEKNOLOGI INDUSTRI

Nama	:	FARID SUJIWO
Nim	:	07.12.629
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika S-1
Judul	:	**APLIKASI KRIPTOGRAFI UNTUK ENKRIPSI PESAN MENGGUNAKAN METODE INTERLOCK PROTOCOL SEBAGAI PENCEGAHAN MAN-IN-THE-MIDDLE-ATTACK**

Dipertahankan dihadapan Tim Pengujian Skripsi jenjang Program Strata Satu (S-1)

Pada Hari : Selasa

Tanggal : 23 Agustus 2011

Dengan Nilai : 84,3 (A)

Panitia Ujian Skripsi :

Ketua Majelcis Penguji

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

Sekretaris Penguji

Dr. Eng. Aryuanto Soetedjo, ST.MT
NIP.Y.1030800417

Anggota Penguji :

Penguji I

Sonny Prasetio, ST, MT
NIP.P.10301000433

Penguji II

Ahmad Faisol, ST
NIP.P.1031000431

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Karanglo KM.2 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi Teknik Komputer & Inforamtika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

Nama	:	FARID SUJIWO
Nim	:	07.12.629
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Komputer & Informatika
Judul	:	**APLIKASI KRIPTOGRAFI UNTUK ENKRIPSI PESAN MENGGUNAKAN METODE INTERLOCK PROTOCOL SEBAGAI PENCEGAHAN MAN-IN-THE-MIDDLE-ATTACK**

No.	Penguji	Tanggal	Uraian	Paraf
1	Penguji I	23/08/2011	1. Tambahkan Kesimpulan pada Abstraksi 2. Ganti kata "pada" dengan "dalam" pada penunjukkan gambar	
2	Penguji II	23/08/2011	1. Tambahkan batasan masalah dekripsi pesan yang dikirim 2. Tambahkan option Help pada aplikasi 3. Tambahkan file yang dapat dienkripsi pada saran	

Disetujui :

Penguji I

Sonny Prasetio, ST.MT
NIP.Y.10301000433

Penguji II

Ahmad Faisol, ST
NIP.P.1031000431

Mengetahui :

Dosen Pembimbing I

Joseph Dedy Irawan, ST, MT
NIP. 19740416 200501 1 002

Dosen Pembimbing I

Sotyohadi, ST
NIP.Y.1039700309

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Farid Sujiwo
NIM : 0712 629
Perbaikan meliputi :

- Keterangan Gambar ditambahkan.
- Abstrak

Malang, 28/8/2011

(SONNY PRASETIO, ST,MT)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

Formulir Perbaikan Ujian Skripsi

Dalam pelaksanaan Ujian Skripsi Janjang Strata 1 Jurusan Teknik Elektro Konsentrasi T. Energi Listrik / T. Elektronika / T.Infokom, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Fani d Sujiwo
NIM : 0712679
Perbaikan meliputi :

~~Petunjuk~~

- Tambahkan petunjuk penggunaan pada aplikasi (Help ??)
- Tambahkan Batasan Permasalahan (teks, file, atau gambar) ???

Malang, 13 Agustus 2014

(A Dafid)

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : ~~Teknik Energi Listrik/Teknik Elektronika~~/ Teknik Komputer & Informatika*)

1.	Nama Mahasiswa: [illegible] Sujito			Nim: 0712624
2.	Keterangan Pelaksanaan	Tanggal: 24 - 07 - 2011	Waktu	Tempat Ruang:
3.	Spesifikasi Judul (berilah tanda silang)**)			
	a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri		e. Elektronika & Komponen (f.) Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya	
4.	Judul Proposal yang diseminarkan Mahasiswa	Aplikasi Kriptografi untuk Enkripsi Pesan Menggunakan Metode Interlock Protocol Sebagai Pencegahan MAN IN THE MIDDLE ATTACK		
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian	....		
6.	Catatan:			
7.	Catatan:			
	Persetujuan Judul Skripsi			
	Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II	Disetujui, Dosen Keahlian III	
	Mengetahui Ketua Jurusan. Ir. Yusuf Ismail Nakhoda,MT NIP.Y. 1018800189	Disetujui, Calon Dosen Pembimbing ybs Pembimbing I	Pembimbing II Setyohadi, ST	

Perhatian:

1. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ... atau g sesuai bidang keahlian

Form S-3c

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

LEMBAR PENGAJUAN JUDUL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik / Teknik Elektronika /Teknik Komputer & Informatika /Teknik Komputer / Teknik Telekomunikasi*)

1.	Nama Mahasiswa: FARID SUJIWO			Nim: 0712629
2.	Waktu Pengajuan	Tanggal: 25	Bulan: Mei	Tahun: 2011
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi (h) lainnya ..Kriptografi			
4.	Konsultasikan judul sesuai materi bidang ilmu kepada Dosen*) Dr. Ary'anto, ST, MT	Ketua Jurusan Ir. Yusuf Ismail Nakhoda, MT NIP. Y. 1018800189		
5.	Judul yang diajukan mahasiswa:	Aplikasi Kriptografi untuk Enkripsi Pesan Menggunakan Metode Interlock Protokol Sebagai Rancangbuku Man-IN-The-Middle Attack		
6.	Perubahan judul yang disetujui Dosen sesuai materi bidang ilmu	.....		
7.	Catatan: .... Persetujuan Judul skripsi yang dikonsultasikan kepada Dosen materi bidang ilmu	Disetujui 26/5/ 2011 Dosen		

Perhatian:

1. Formulir pengajuan ini harap dikembalikan kepada jurusan paling lambat satu minggu setelah disetujui kelompok dosen keahlian dengan dilampirkan proposal skripsi beserta persyaratan skripsi sesuai form S-1
2. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ... atau g sesuai bidang keahlian

Form S-2

INSTITUT TEKNOLOGI NASIONAL

Jln. Raya Karanglo, Km. 2 Malang

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa/i:

Nama : Farid Sujiwo

NIM : 07.12.629

Semester : VIII (Delapan)

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Komputer dan Informatika

Dengan ini menyatakan bersedia/ ~~tidak bersedia~~*) Membimbing skripsi dari mahasiswa tersebut, dengan judul:

" Aplikasi Kriptografi Untuk Enkripsi Pesan Menggunakan Metode Interlock Protocol Sebagai Pencegahan Man In The Middle Attack "

Demikian surat pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang,

Kami yang membuat pernyataan

Joseph Dedy Irawan ,ST,MT
NIP. 19740416 200501 1 002

Catatan :

Setelah disetujui agar formulir ini

Diserahkan mahasiswa/I yang bersangkutan

Kepada Jurusan untuk diproses lebih lanjut.

*) Coret yang tidak perlu

Form S-3b

INSTITUT TEKNOLOGI NASIONAL

Jln. Raya Karanglo, Km. 2 Malang

PERNYATAAN KESEDIAAN DALAM PEMBIMBINGAN SKRIPSI

Sesuai permohonan dari mahasiswa/i:

Nama : Farid Sujiwo

NIM : 07.12.629

Semester : VIII (Delapan)

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Komputer dan Informatika

Dengan ini menyatakan bersedia/ ~~tidak bersedia~~*) Membimbing skripsi dari mahasiswa tersebut, dengan judul:

" Aplikasi Kriptografi Untuk Enkripsi Pesan Menggunakan Metode Interlock Protocol Sebagai Pencegahan Man In The Middle Attack "

Demikian surat pernyataan ini kami buat agar dapat dipergunakan seperlunya.

Malang,

Kami yang membuat pernyataan

Sotyohadi, ST
NIP.Y. 1039700309

Catatan :

Setelah disetujui agar formulir ini

Diserahkan mahasiswa/i yang bersangkutan

Kepada Jurusan untuk diproses lebih lanjut.

*) Coret yang tidak perlu

Form S-3b

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting) Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 20 Juli 2011

Nomor : ITN-402/I.TA/2/11
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Sdr./i . **JOSEPH DEDY IRAWAN, ST, MT**
Dosen Institut Teknologi Nasional Malang

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: FARID SUJIWO
Nim	: 0712629
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik **KOMPUTER & INFORMATIKA**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu (enam) 6 bulan, terhitung mulai tanggal :

09 Juli 2011 s/d 09 Januari 2012

Sebagai satu syarat untuk menempuh ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih

Ketua Jurusan
Teknik Elektro S-1

Ir. Yusef Ismail Nakhoda, MT
Nip. Y.1018800189

Tembusan Kepada Yth :
1. Mahasiswa Yang Bersangkutan
2. Arsip
3. Coret yang tidak perlu

Form. S 4a

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
F. KULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJ. .NA MAGISTER TEKNIK

T. BNI (PERSERO) MALANG
BANK NIAGA MALANG

Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting) Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 20 Juli 2011

Nomor : ITN-403/I.TA/2/11
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Sdr./i . **SOTYOHADI, ST**
Dosen Institut Teknologi Nasional Malang

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: FARID SUJIWO
Nim	: 0712629
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik **KOMPUTER & INFORMATIKA**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu (enam) 6 bulan, terhitung mulai tanggal :

09 Juli 2011 s/d 09 Januari 2012

Sebagai satu syarat untuk menempuh ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih

Ketua Jurusan
Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT
Nip. Y.1018800189

Tembusan Kepada Yth :
1. Mahasiswa Yang Bersangkutan
2. Arsip
3. Coret yang tidak perlu

Form. S 4a

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, KM 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Farid Sujiwo
Nim : 07.12.629
Masa Bimbingan : 5 Juli 2011 – 5 Januari 2012
Judul Skripsi : Aplikasi Kriptografi Untuk Enkripsi Pesan Menggunakan Metode Interlock Protocol Sebagai Pencegahan Man-In-The-Middle-Attack

No	Tanggal	Uraian	Paraf Pembimbing
1	14-07-2011	Ubah program dari simulasi ke pengiriman pesan pada jaringan	
2	27-07-2011	Hapus delay pada program saat eksekusi	
3	29-07-2011	Pisah kan antara kolom eksekusi dan kolom perekaman pada saat mengirim pesan	
4	03-08-2011	Acc Bab I, Bab II, Bab III, Bab IV dan Bab V	
5			
6			
7			
8			
9			
10			

Malang,
Dosen Pembimbing II

Sotyohadi, ST
NIP.Y.1039700309

Form S-4b

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo, KM 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Farid Sujiwo
Nim : 07.12.629
Masa Bimbingan : 5 Juli 2011 – 5 Januari 2012
Judul Skripsi : Aplikasi Kriptografi Untuk Enkripsi Pesan Menggunakan Metode Interlock Protocol Sebagai Pencegahan Man-In-The-Middle-Attack

No	Tanggal	Uraian	Paraf Pembimbing
1	21-07-2011	Ubah program dari simulasi ke bentuk percakapan pada jaringan	
2	26-07-2011	Jadikan Program percakapan secara point to point antara client dan server	
3	16-08-2011	ACC Bab I, Bab II, Bab III, Bab IV, Bab V	
4		ACC SEMINAR HASIL	
5		ACC KOMPRE	
6			
7			
8			
9			
10			

Malang,
Dosen Pembimbing I

Joseph Dedy Irawan, ST, MT
NIP. 19740416 200501 1 002

Form S-4b

Source Code (Visual Basic 6.0)

➢ Frmclient.frm

```
Option Explicit

Private A As Integer
Private Proses As Boolean
Private Speed As Integer

'Hasil enkripsi
Private HE As String

'Hasil dekripsi
Private HD As String

'Hasil Kiriman Blok
Private HB(2) As String

Private Sub chkB1_Click()
   If chkB1.Value = 1 Then
      chkB2.Value = 0
   End If
End Sub

Private Sub chkB2_Click()
   If chkB2.Value = 1 Then
      chkB1.Value = 0
   End If
End Sub

Private Sub chkHash_Click()
   If chkHash.Value = 1 Then
      chkSplit.Value = 0
      cmdHash.Enabled = True
      txtHas.Enabled = True
      txtHas2.Enabled = True
      cmdCek.Enabled = True
      cmdsv3.Enabled = True
   Else
      txtHas.Text = ""
      txtHas2.Text = ""
      cmdHash.Enabled = False
      txtHas.Enabled = False
      txtHas2.Enabled = False
      cmdCek.Enabled = False
      cmdsv3.Enabled = False
   End If
End Sub

Private Sub chkSplit_Click()
   If chkSplit.Value = 1 Then
      chkHash.Value = 0
      chkB1.Enabled = True
      chkB2.Enabled = True
      txtB1.Enabled = True
      txtB2.Enabled = True
      cmdCmp.Enabled = True
      cmdSplit.Enabled = True
      cmdSv2.Enabled = True
   Else
      chkB1.Value = 0
      chkB2.Value = 0
      txtB1.Text = ""
      txtB2.Text = ""
      chkB1.Enabled = False
      chkB2.Enabled = False
      txtB1.Enabled = False
      txtB2.Enabled = False
      cmdCmp.Enabled = False
      cmdSplit.Enabled = False
      cmdSv2.Enabled = False
   End If
End Sub

Private Sub cmdAbout_Click()
   frmAbout.Show vbModal
End Sub

Private Sub cmdCek_Click()
Dim ask As String

If txtHas.Text = txtHas2.Text Then
   MsgBox ("Hasil nilai Hash sesuai, dekripsi Pesan ?"),
vbOKOnly
   Call Dekripsi(txtCpr)
   txtHas.Text = ""
   txtHas2.Text = ""
End If
If txtHas.Text <> txtHas2.Text Then
   ask = MsgBox("Hasil nilai Hash tidak sesuai ! dekripsi
Pesan ?", vbYesNo, Me.Caption)
   If ask = vbYes Then
      Call Dekripsi(txtCpr)
      txtHas.Text = ""
      txtHas2.Text = ""
   Else
      Exit Sub
   End If
End If
End Sub

Private Sub cmdCmp_Click()
If txtB1.Text = "" And txtB2.Text = "" Then
      MsgBox "Nilai masih kosong !", vbCritical
   Else
      If txtB1.Text = "" And txtB2.Text = txtB2.Text Then
      MsgBox "Nilai Belum lengkap !", vbCritical
      txtB1.SetFocus
      Else
         If txtB1.Text = txtB1.Text And txtB2.Text = "" Then
      MsgBox "Nilai Belum lengkap !", vbCritical
      txtB2.SetFocus
         Else
   Dim N As Integer
   Dim x As String

   x = ""
   For N = 1 To Len(txtB1)
      x = x & Mid(txtB1, N, 1) & Mid(txtB2, N, 1)

   Next N
   txtCpr.Text = x
   MsgBox ("Chiper teks berhasil di gabungkan, dekripsi
pesan?"), vbOKOnly
   Call Dekripsi(x)
   txtB1.Text = ""
   txtB2.Text = ""
   Exit Sub
         End If
      End If
End If
End Sub

Private Sub CmdConnect_Click()
If txtLogin.Text = Empty Or txtLogin.Text = " " Then
MsgBox "Enter a Client Name.", vbInformation +
vbOKOnly, "Error"
txtLogin.SetFocus
Exit Sub
End If
If txtRemotePort.Text = Empty Then
MsgBox "Enter a Remote Port.", vbInformation +
vbOKOnly, "Error"
txtRemotePort.SetFocus
Exit Sub
End If
If txtRemoteHost.Text = Empty Then
MsgBox "Enter a Remote Host.", vbInformation +
vbOKOnly, "Error"
txtRemoteHost.SetFocus
Exit Sub
```

```
End If
If txtKad.Text = Empty Or txtKan.Text = Empty Or
txtKae.Text = Empty Then
MsgBox "Enter key first.", vbInformation + vbOKOnly,
"Error"
frmKey.Show vbModal

txtKad.Text = Akey.d
txtKae.Text = Akey.e
txtKan.Text = Akey.N
Exit Sub
End If
  tcpClient.RemoteHost = txtRemoteHost.Text
  tcpClient.RemotePort = txtRemotePort.Text
  sndKey1.RemoteHost = txtRemoteHost.Text
  sndKey2.RemoteHost = txtRemoteHost.Text
  Winsock1.RemoteHost = txtRemoteHost.Text
  Winsock2.RemoteHost = txtRemoteHost.Text
  tcpClient.Connect
  sndKey1.Connect
  sndKey2.Connect
  Winsock1.Connect
  Winsock2.Connect
  lblStatus.Caption = "Status : Asking Connection to
Server"
  CmdConnect.Enabled = False
  CmdDis.Enabled = True
  CmdSend.Enabled = True
End Sub

Private Sub cmdCr_Click()
  txtPro.Text = ""
  txtPsn.Text = ""
End Sub

Private Sub CmdDis_Click()
  lblStatus.Caption = "Status : You Disconnected !"
  CmdConnect.Enabled = True
  CmdDis.Enabled = False
  txtMsg.Enabled = False
  CmdSend.Enabled = False
  txtPln.Enabled = False
  cmdSv.Enabled = False
  cmdEn.Enabled = False
  chkHash.Enabled = False
  chkSplit.Enabled = False
  sndKey1.Close
  sndKey2.Close
  Winsock1.Close
  Winsock2.Close
  shpCon.FillColor = vbRed
  tcpClient.SendData "code rejected 0"
  Timer1.Interval = 2000
End Sub
Private Sub cmdEn_Click()
  If txtPln.Text = "" Then
    MsgBox "Pesan masih kosong !", vbCritical
    txtPln.SetFocus
    Exit Sub
  End If

  Msg2 = txtPln.Text

  Call Enkripsi(Msg2)
End Sub
Private Sub cmdHash_Click()
  Dim hsh As String
  Dim hslhash As String

  If txtCpr.Text = "" Then
    MsgBox "Pesan masih kosong !", vbCritical
    txtCpr.SetFocus
    Exit Sub
  End If

  hsh = txtCpr.Text
  hslhash = Sha1(hsh)

  If txtHas.Text = Empty And txtHas2.Text = Empty Then
    txtMsg.Text = hslhash
  Else
    txtHas.Text = hslhash
  End If
End Sub
Private Sub cmdInput_Click()
  frmKey.Show vbModal

  txtKad.Text = Akey.d
  txtKae.Text = Akey.e
  txtKan.Text = Akey.N
End Sub

Private Sub CmdSend_Click()
If txtMsg.Text = Empty Then
  MsgBox "Pesan yang akan dikirim kosong !", vbCritical
  txtMsg.SetFocus
  Exit Sub
End If
If txtMsg.Text = Empty Then Exit Sub
    myData = ""
    myData = txtMsg.Text
If chkHash.Value = 1 And txtCpr.Text = Empty Then
    tcpClient.SendData myData
    txtPsn.Text = txtPsn.Text & vbCrLf & vbCrLf & "Me :
" & txtPln & vbCrLf
    txtMsg.Text = ""
Else
  If chkHash.Value = 1 And txtCpr.Text = txtCpr Then
    tcpClient.SendData myData
    txtMsg.Text = ""
Else
  If chkSplit.Value = 1 And chkB1.Value = 1 Then
    tcpClient.SendData myData
    txtMsg.Text = ""
    chkB1.Value = 0
Else
    tcpClient.SendData myData
    txtPsn.Text = txtPsn.Text & vbCrLf & vbCrLf & "Me :
" & txtPln & vbCrLf
    txtMsg.Text = ""
    txtCpr.Text = ""
    chkB2.Value = 0
    End If
    End If
    End If
End Sub

Private Sub cmdSplit_Click()
  If txtCpr.Text = "" Then
    MsgBox "Pesan masih kosong !", vbCritical
    txtCpr.SetFocus
    Exit Sub
  End If

  Dim N As Integer
  Dim Blok(2) As String

  Blok(1) = ""
  Blok(2) = ""
  For N = 1 To Len(txtCpr) Step 2
    Blok(1) = Blok(1) & Mid(txtCpr, N, 1)
    Blok(2) = Blok(2) & Mid(txtCpr, N + 1, 1)
  Next N
  txtB1.Text = Blok(1)
  txtB2.Text = Blok(2)
  txtCpr.Text = ""
End Sub

Private Sub cmdSv_Click()
  If txtCpr.Text = "" Then
    MsgBox "Pesan masih kosong !", vbCritical
    txtCpr.SetFocus
    Exit Sub
  End If
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf &
"Chiper text : " & txtCpr
    txtMsg.Text = txtCpr
    txtCpr.Text = ""
```

```
End Sub

Private Sub cmdSv1_Click()
   If txtPln.Text = "" Then
      MsgBox "Pesan masih kosong !", vbCritical
      txtPln.SetFocus
      Exit Sub
   End If
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & txtPln
   txtMsg.Text = txtPln
   txtPln.Text = ""
End Sub
Private Sub cmdSv2_Click()
   If chkB1.Value = 1 And chkB2.Value = 0 Then
      If txtB1.Text = Empty Then
         MsgBox "Isi pesan bagian-1 kosong !", vbCritical
         txtB1.SetFocus
         Exit Sub
      End If
      txtMsg.Text = txtB1
      txtPro.Text = txtPro.Text & vbCrLf & "Chiper teks
bagian-1 :" & txtB1 & vbCrLf
      txtB1.Text = ""
   Else
      If chkB1.Value = 0 And chkB2.Value = 1 Then
         If txtB2.Text = Empty Then
            MsgBox "Isi pesan bagian-2 kosong !", vbCritical
            txtB2.SetFocus
            Exit Sub
         End If
         txtMsg.Text = txtB2
         txtPro.Text = txtPro.Text & vbCrLf & "Chiper teks
bagian-2 :" & txtB2 & vbCrLf
         txtB2.Text = ""
   Else
   If chkSplit.Value = 1 And txtB1.Text = Empty And
txtB2.Text = Empty Then
      txtB1.Text = txtCpr
      txtCpr.Text = ""
   Else
      If chkSplit.Value = 1 And txtB1.Text = txtB1 And
txtB2.Text = Empty Then
      txtB2.Text = txtCpr
      txtCpr.Text = ""
   Else
      If chkB1.Value = 0 And chkB2.Value = 0 And
txtHas.Text = Empty Then
         MsgBox "Tidak ada yang bisa disimpan atau dikirim
!", vbCritical
            Exit Sub
         End If
         End If
         End If
         End If
   End If
End Sub

Private Sub cmdsv3_Click()
If txtCpr.Text = Empty Then
   MsgBox "Pesan hash masih kosong !", vbCritical
   Exit Sub
End If
      If chkHash.Value = 1 And txtHas.Text = Empty And
txtHas2.Text = Empty Then
      txtHas2.Text = txtCpr
      txtCpr.Text = ""
   End If
End Sub

Private Sub Form_Load()
   lblStatus.Caption = "Status : "
   CmdDis.Enabled = False
   txtMsg.Enabled = False
   chkB1.Enabled = False
   chkB2.Enabled = False
   txtB1.Enabled = False
   txtB2.Enabled = False
   cmdCmp.Enabled = False
   cmdHash.Enabled = False
   cmdSplit.Enabled = False
   cmdSv2.Enabled = False
   txtHas.Enabled = False
   CmdSend.Enabled = False
   cmdCek.Enabled = False
   txtHas2.Enabled = False
   cmdsv3.Enabled = False
   txtPln.Enabled = False
   cmdSv.Enabled = False
   cmdEn.Enabled = False
   chkHash.Enabled = False
   chkSplit.Enabled = False
   txtPln.Enabled = False
   cmdSv.Enabled = False
   cmdEn.Enabled = False
   chkHash.Enabled = False
   chkSplit.Enabled = False
   shpCon.FillColor = vbRed
End Sub

Private Sub tcpClient_DataArrival(ByVal bytesTotal As
Long)
Dim strData As String
   tcpClient.GetData strData

If strData = "code rejected 0" Then
   lblStatus.Caption = "Status : Connection Refused !"
   CmdConnect.Enabled = True
   CmdDis.Enabled = False
   txtMsg.Enabled = False
   CmdSend.Enabled = False
   txtPln.Enabled = False
   cmdSv.Enabled = False
   cmdEn.Enabled = False
   chkHash.Enabled = False
   chkSplit.Enabled = False
   tcpClient.SendData "code rejected 0"
   tcpClient.Close
   sndKey1.Close
   sndKey2.Close
   Winsock1.Close
   Winsock2.Close
   shpCon.FillColor = vbRed
   Exit Sub
   Exit Sub
End If

If strData = "code accepted 1" Then
   sndKey1.SendData txtKae.Text
   Sleep 500
   sndKey2.SendData txtKan.Text
   lblStatus.Caption = "Status : " & txtLogin.Text & "
connected !"
   CmdConnect.Enabled = False
   CmdDis.Enabled = True
   txtMsg.Enabled = True
   CmdSend.Enabled = True
   txtPln.Enabled = True
   cmdSv.Enabled = True
   cmdEn.Enabled = True
   chkHash.Enabled = True
   chkSplit.Enabled = True
   shpCon.FillColor = vbGreen
   Exit Sub
End If

If chkSplit.Value = 1 Or chkHash.Value = 1 Then
   txtCpr.Text = ""
   txtCpr.Text = txtCpr.Text & strData
Else
   txtCpr.Text = ""
   txtCpr.Text = txtCpr.Text & strData
   Call Dekripsi(strData)
   End If
End Sub

Private Sub Timer1_Timer()
   tcpClient.Close
   sndKey1.Close
```

```
    sndKey2.Close
    Winsock1.Close
    Winsock2.Close
    Timer1.Interval = 0
End Sub

Private Sub txtCpr_Click()
txtCpr.SelStart = 0
txtCpr.SelLength = Len(txtCpr.Text)
End Sub

Private Sub txtLogin_Change()
If Len(txtLogin.Text) <= 0 Then
CmdConnect.Enabled = False
Else
CmdConnect.Enabled = True
End If
End Sub

Private Sub txtLogin_Click()
txtLogin.SelStart = 0
txtLogin.SelLength = Len(txtLogin.Text)
End Sub

Private Sub txtLogin_LostFocus()
txtLogin.Text = UCase(txtLogin.Text)
End Sub

Private Sub txtPln_Click()
txtPln.SelStart = 0
txtPln.SelLength = Len(txtPln.Text)
End Sub

Private Sub txtPro_Change()
txtPro.SelStart = Len(txtPro.Text)
End Sub
'Prosedur Enkripsi
Private Sub Enkripsi(pcText As String)
    Dim Biner As String
    Dim Desimal1 As String
    Dim Desimal2 As String
    Dim N As Integer
    Dim Temp As String
    Dim nTemp As Double

    'Pesan
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Pesan = '" & pcText & "'"

    'Sleep 500

    'Ubah menjadi biner
    Biner = ""
    For N = 1 To Len(pcText)
        Biner = Biner &
FormatS(DecToBiner(Asc(Mid(pcText, N, 1))), "0", 8)
    Next N
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Ubah pesan menjadi biner:"
    'Sleep 500
    txtPro.Text = txtPro.Text & vbCrLf & Biner
    'Sleep 500

    'Tambah Nol
    If Len(Biner) Mod 3 <> 0 Then
        Biner = Biner & String(3 - (Len(Biner) Mod 3), "0")
    End If

    'Penggal setiap 3 bit biner dan ubah menjadi desimal
    Desimal1 = ""
    For N = 1 To Len(Biner) Step 3
        Temp = Mid(Biner, N, 3)
        If Len(Temp) <> 3 Then
            Temp = FormatS(Temp, "0", 3)
        End If
        Desimal1 = Desimal1 & Format(BinerToDec(Temp))
    Next N
    'Tulis
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Ubah setiap 3 bit biner menjadi bentuk
desimal:"
    'Sleep 500
    txtPro.Text = txtPro.Text & vbCrLf & Desimal1
    'Sleep 500

    'Ambil 4 digit desimal, masukkan ke fungsi
enkripsi/dekripsi
    'Tulis
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
    "Masukkan setiap 4 digit desimal (m) ke fungsi " & _
    "enkripsi: c = (m ^ c) mod n"
    Desimal2 = ""
    For N = 1 To Len(Desimal1) Step 4

        'Fungsi enkripsi
        Temp = Mid(Desimal1, N, 4)
        If Len(Temp) <> 4 Then Temp = Temp & String(4 -
Len(Temp), "0")
        nTemp = FastExp(Val(Temp), Val(txtKma),
Val(txtKmn))

        'Hasil enkripsi
        If Desimal2 <> "" Then Desimal2 = Desimal2 & " "
        Desimal2 = Desimal2 & Format(nTemp)

        'Tulis
        txtPro.Text = txtPro.Text & vbCrLf & _
        "c = (" & Temp & " ^ " & _
        txtKma & ") mod " & txtKan & " = " & nTemp

        'Sleep 500
    Next N

    'Tulis
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
    "Hasil enkripsi: " & vbCrLf & Desimal2 & vbCrLf

    'Hasil enkripsi
    HE = Desimal2
    txtMsg.Text = HE
    txtCpr.Text = HE
End Sub

'Prosedur Dekripsi
Private Sub Dekripsi(pcText As String)
    Dim Biner As String
    Dim Desimal1 As String
    Dim Desimal2() As String
    Dim N As Integer
    Dim Temp As String
    Dim nTemp As Double

    'Pesan
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Cipher Text = '" & pcText & "'"

    'Sleep 500

    'Masukkan ke fungsi dekripsi
    'Tulis
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
    "Masukkan cipher text (c) ke fungsi dekripsi: m = (c ^ d)
mod n"
    Desimal1 = ""
    Desimal2 = Split(pcText, " ")
    For N = 0 To UBound(Desimal2)
        If Trim(Desimal2(N)) <> "" Then
            'Fungsi dekripsi
            nTemp = FastExp(Val(Desimal2(N)), Val(txtKad),
Val(txtKan))

            'Hasil dekripsi
            Temp = Format(nTemp)
            If Len(Temp) < 4 Then Temp = String(4 -
Len(Temp), "0") & Temp
            Desimal1 = Desimal1 & Temp

            'Tulis
```

```
        txtPro.Text = txtPro.Text & vbCrLf & _
        "m = (" & Desimal2(N) & " ^ " & _
        txtKad & ") mod " & txtKan & " = " & Temp

        'Sleep 500
      End If
    Next N

    'Tulis
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
    "Hasil dekripsi: " & vbCrLf & Desimal1
    'Sleep 500

    'Ubah ke biner
    Biner = ""
    For N = 1 To Len(Desimal1)
      Biner = Biner &
FormatS(DecToBiner(Val(Mid(Desimal1, N, 1))), "0", 3)
    Next N
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
          "Ubah pesan menjadi biner:"
    'Sleep 500
    txtPro.Text = txtPro.Text & vbCrLf & Biner
    'Sleep 500

    'Ubah ke ascii
    Temp = ""
    For N = 1 To Len(Biner) Step 8
      Temp = Temp & Chr(BinerToDec(Mid(Biner, N, 8)))
    Next N
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
          "Ubah setiap 8 bit biner ke bentuk ascii:"
    'Sleep 500
    txtPro.Text = txtPro.Text & vbCrLf & Temp & vbCrLf
    'Sleep 500

    'Hasil dekripsi
    HD = Temp
    txtPln.Text = HD
    txtPsn.Text = txtPsn.Text & vbCrLf & "SERVER :" &
HD & vbCrLf
    txtCpr.Text = ""
End Sub
Private Sub sndKey1_DataArrival(ByVal bytesTotal As
Long)
Dim kme As String
    sndKey1.GetData kme
    txtKma.Text = ""
    txtKma.Text = txtKma.Text & kme
End Sub

Private Sub sndKey2_DataArrival(ByVal bytesTotal As
Long)
Dim kmn As String
    sndKey2.GetData kmn

    txtKmn.Text = ""
    txtKmn.Text = txtKmn.Text & kmn
End Sub

Private Sub txtRemoteHost_Click()
txtRemoteHost.SelStart = 0
txtRemoteHost.SelLength = Len(txtRemoteHost.Text)
End Sub

Private Sub txtRemotePort_Click()
txtRemotePort.SelStart = 0
txtRemotePort.SelLength = Len(txtRemotePort.Text)
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As
Long)
    Dim dt As String
    Winsock1.GetData dt

If dt = "1" Then
    chkHash.Value = 1
Else
    chkHash.Value = 0
    End If
End Sub

Private Sub Winsock2_DataArrival(ByVal bytesTotal As
Long)
    Dim dts As String
    Winsock2.GetData dts

If dts = "1" Then
    chkSplit.Value = 1
Else
    chkSplit.Value = 0
    End If
End Sub
```

➢ frmServer.frm

```
Option Explicit

Private A As Integer
Private Proses As Boolean
Private Speed As Integer

'Hasil enkripsi
Private HE As String

'Hasil dekripsi
Private HD As String

'Hasil Kiriman Blok
Private HB(2) As String

Private Sub chkB1_Click()
    If chkB1.Value = 1 Then
      chkB2.Value = 0
    End If
End Sub

Private Sub chkB2_Click()
    If chkB2.Value = 1 Then
      chkB1.Value = 0
    End If
End Sub

Private Sub chkHash_Click()
    If chkHash.Value = 1 Then
      chkSplit.Value = 0
      cmdHash.Enabled = True
      txtHas.Enabled = True
      txtHas2.Enabled = True
      cmdCek.Enabled = True
      cmdsv3.Enabled = True
    Else
      txtHas.Text = ""
      txtHas2.Text = ""
      cmdCek.Enabled = False
      cmdHash.Enabled = False
      txtHas.Enabled = False
      txtHas2.Enabled = False
      cmdsv3.Enabled = False
    End If
End Sub

Private Sub chkSplit_Click()
    If chkSplit.Value = 1 Then
      chkHash.Value = 0
      chkB1.Enabled = True
      chkB2.Enabled = True
      txtB1.Enabled = True
      txtB2.Enabled = True
      cmdCmp.Enabled = True
      cmdSplit.Enabled = True
      cmdSv2.Enabled = True
```

```
    Else
        chkB1.Value = 0
        chkB2.Value = 0
        txtB1.Text = ""
        txtB2.Text = ""
        chkB1.Enabled = False
        chkB2.Enabled = False
        txtB1.Enabled = False
        txtB2.Enabled = False
        cmdCmp.Enabled = False
        cmdSplit.Enabled = False
        cmdSv2.Enabled = False
    End If
End Sub

Private Sub cmdAbout_Click()
    frmAbout.Show vbModal
End Sub

Private Sub cmdCek_Click()
Dim ask As String

If txtHas.Text = txtHas2.Text Then
   MsgBox ("Hasil nilai Hash sesuai, dekripsi Pesan ?"),
vbOKOnly
   Call Dekripsi(txtCpr)
   txtHas.Text = ""
   txtHas2.Text = ""
End If
If txtHas.Text <> txtHas2.Text Then
   ask = MsgBox("Hasil nilai Hash tidak sesuai ! dekripsi
Pesan ?", vbYesNo, Me.Caption)
   If ask = vbYes Then
      Call Dekripsi(txtCpr)
      txtHas.Text = ""
      txtHas2.Text = ""
   Else
      Exit Sub
   End If
End If
End Sub

Private Sub cmdCmp_Click()
 If txtB1.Text = "" And txtB2.Text = "" Then
      MsgBox "Nilai masih kosong !", vbCritical
   Else
      If txtB1.Text = "" And txtB2.Text = txtB2.Text Then
      MsgBox "Nilai Belum lengkap !", vbCritical
      txtB1.SetFocus
      Else
         If txtB1.Text = txtB1.Text And txtB2.Text = "" Then
      MsgBox "Nilai Belum lengkap !", vbCritical
      txtB2.SetFocus
         Else
   Dim N As Integer
   Dim x As String

   x = ""
   For N = 1 To Len(txtB1)
      x = x & Mid(txtB1, N, 1) & Mid(txtB2, N, 1)

   Next N
   txtCpr.Text = x
   MsgBox ("Chiper teks berhasil di gabungkan, dekripsi
pesan?"), vbOKOnly
   Call Dekripsi(x)
   txtB1.Text = ""
   txtB2.Text = ""
   Exit Sub
         End If
      End If
 End If
End Sub

Private Sub cmdCr_Click()
   txtPro.Text = ""
   txtPsn.Text = ""
End Sub

Private Sub CmdDis_Click()
   lblStatus.Caption = "Status : You Disconnected !"
   CmdDis.Enabled = False
   tcpServer.SendData "code rejected 0"
   txtMsg.Enabled = False
   CmdSend.Enabled = False
   txtPln.Enabled = False
   cmdSv.Enabled = False
   cmdEn.Enabled = False
   chkHash.Enabled = False
   chkSplit.Enabled = False
   shpCon.FillColor = vbRed
   Sleep 2000
   tcpServer.Close
   sndKey1.Close
   sndKey2.Close
   Winsock1.Close
   Winsock2.Close
   cmdStart.Enabled = True
End Sub
Private Sub cmdEn_Click()
   If txtPln.Text = "" Then
      MsgBox "Pesan masih kosong !", vbCritical
      txtPln.SetFocus
      Exit Sub
   End If

   Msg2 = txtPln.Text

   Call Enkripsi(Msg2)
End Sub
Private Sub cmdHash_Click()
   Dim hsh As String
   Dim hslhash As String

   If txtCpr.Text = "" Then
      MsgBox "Pesan masih kosong !", vbCritical
      txtCpr.SetFocus
      Exit Sub
   End If

   hsh = txtCpr.Text
   hslhash = Sha1(hsh)

   If txtHas.Text = Empty And txtHas2.Text = Empty Then
      txtMsg.Text = hslhash
   Else
   txtHas.Text = hslhash
   End If
End Sub

Private Sub cmdInput_Click()
   frmKey.Show vbModal

   txtKad.Text = Akey.d
   txtKae.Text = Akey.e
   txtKan.Text = Akey.N
End Sub

Private Sub CmdSend_Click()
If txtMsg.Text = Empty Then
   MsgBox "Pesan yang akan dikirim kosong !", vbCritical
   txtMsg.SetFocus
   Exit Sub
End If
If txtMsg.Text = Empty Then Exit Sub
      myData = ""
      myData = txtMsg.Text
If chkHash.Value = 1 And txtCpr.Text = Empty Then
      tcpServer.SendData myData
      txtPsn.Text = txtPsn.Text & vbCrLf & vbCrLf & "Me :
" & txtPln & vbCrLf
      txtMsg.Text = ""
Else
   If chkHash.Value = 1 And txtCpr.Text = txtCpr Then
      tcpServer.SendData myData
      txtMsg.Text = ""
Else
   If chkSplit.Value = 1 And chkB1.Value = 1 Then
```

```
        tcpServer.SendData myData
        txtMsg.Text = ""
        chkB1.Value = 0
    Else
        tcpServer.SendData myData
        txtPsn.Text = txtPsn.Text & vbCrLf & vbCrLf & "Me :
" & txtPln & vbCrLf
        txtMsg.Text = ""
        txtCpr.Text = ""
        chkB2.Value = 0
        End If
        End If
        End If
End Sub

Private Sub cmdSplit_Click()
    If txtCpr.Text = "" Then
        MsgBox "Pesan masih kosong !", vbCritical
        txtCpr.SetFocus
        Exit Sub
    End If

    Dim N As Integer
    Dim Blok(2) As String

    Blok(1) = ""
    Blok(2) = ""
    For N = 1 To Len(txtCpr) Step 2
        Blok(1) = Blok(1) & Mid(txtCpr, N, 1)
        Blok(2) = Blok(2) & Mid(txtCpr, N + 1, 1)
    Next N
    txtB1.Text = Blok(1)
    txtB2.Text = Blok(2)
    txtCpr.Text = ""
End Sub

Private Sub cmdStart_Click()
If txtLogin.Text = Empty Or txtLogin.Text = " " Then
MsgBox "Enter a Server Name.", vbInformation +
vbOKOnly, "Error"
txtLogin.SetFocus
Exit Sub
End If
If txtLocalPort.Text = Empty Then
MsgBox "Enter a Local Port.", vbInformation + vbOKOnly,
"Error"
txtLocalPort.SetFocus
Exit Sub
End If
If txtKad.Text = Empty Or txtKan.Text = Empty Or
txtKae.Text = Empty Then
MsgBox "Enter key first.", vbInformation + vbOKOnly,
"Error"
frmKey.Show vbModal

txtKad.Text = Akey.d
txtKae.Text = Akey.e
txtKan.Text = Akey.N
Exit Sub
End If
    tcpServer.LocalPort = txtLocalPort.Text
    tcpServer.Listen
    sndKey1.Listen
    sndKey2.Listen
    Winsock1.Listen
    Winsock2.Listen
    lblStatus.Caption = "Status : Waiting Connection from
Client"
    cmdStart.Enabled = False
End Sub

Private Sub cmdSv_Click()
    If txtCpr.Text = "" Then
        MsgBox "Pesan masih kosong !", vbCritical
        txtCpr.SetFocus
        Exit Sub
    End If
        txtPro.Text = txtPro.Text & vbCrLf & vbCrLf &
"Chiper text : " & txtCpr
        txtMsg.Text = txtCpr
        txtCpr.Text = ""
End Sub

Private Sub cmdSv1_Click()
    If txtPln.Text = "" Then
        MsgBox "Pesan masih kosong !", vbCritical
        txtPln.SetFocus
        Exit Sub
    End If
    txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & txtPln
    txtMsg.Text = txtPln
    txtPln.Text = ""
End Sub
Private Sub cmdSv2_Click()
    If chkB1.Value = 1 And chkB2.Value = 0 Then
        If txtB1.Text = Empty Then
            MsgBox "Isi pesan bagian-1 kosong !", vbCritical
            txtB1.SetFocus
            Exit Sub
        End If
        txtMsg.Text = txtB1
        txtPro.Text = txtPro.Text & vbCrLf & "Chiper teks
bagian-1 :" & txtB1 & vbCrLf
        txtB1.Text = ""
    Else
        If chkB1.Value = 0 And chkB2.Value = 1 Then
            If txtB2.Text = Empty Then
                MsgBox "Isi pesan bagian-2 kosong !", vbCritical
                txtB2.SetFocus
                Exit Sub
            End If
            txtMsg.Text = txtB2
            txtPro.Text = txtPro.Text & vbCrLf & "Chiper teks
bagian-2 :" & txtB2 & vbCrLf
            txtB2.Text = ""
    Else
    If chkSplit.Value = 1 And txtB1.Text = Empty And
txtB2.Text = Empty Then
        txtB1.Text = txtCpr
        txtCpr.Text = ""
    Else
        If chkSplit.Value = 1 And txtB1.Text = txtB1 And
txtB2.Text = Empty Then
        txtB2.Text = txtCpr
        txtCpr.Text = ""
    Else
        If chkB1.Value = 0 And chkB2.Value = 0 And
txtHas.Text = Empty Then
            MsgBox "Tidak ada yang bisa disimpan atau dikirim
!", vbCritical
                Exit Sub
            End If
            End If
            End If
            End If
    End If
End Sub

Private Sub cmdsv3_Click()
    If txtCpr.Text = "" Then
        MsgBox "Pesan Hash masih kosong !", vbCritical
        Exit Sub
    End If
        If chkHash.Value = 1 And txtHas.Text = Empty And
txtHas2.Text = Empty Then
        txtHas2.Text = txtCpr
        txtCpr.Text = ""
    End If
End Sub

Private Sub Form_Load()
    shpCon.FillColor = vbRed
    lblStatus.Caption = "Status : "
    CmdDis.Enabled = False
    txtMsg.Enabled = False
    CmdSend.Enabled = False
    chkB1.Enabled = False
    chkB2.Enabled = False
```

```
    txtB1.Enabled = False
    txtB2.Enabled = False
    cmdCmp.Enabled = False
    cmdHash.Enabled = False
    cmdSplit.Enabled = False
    cmdSv2.Enabled = False
    txtHas.Enabled = False
    CmdSend.Enabled = False
    cmdCek.Enabled = False
    txtHas2.Enabled = False
    cmdsv3.Enabled = False
    txtPln.Enabled = False
    cmdSv.Enabled = False
    cmdEn.Enabled = False
    chkHash.Enabled = False
    chkSplit.Enabled = False
End Sub

Private Sub sndKey1_ConnectionRequest(ByVal requestID
As Long)
If sndKey1.State <> sckClosed Then _
    sndKey1.Close

    sndKey1.Accept requestID
End Sub

Private Sub sndKey2_ConnectionRequest(ByVal requestID
As Long)
If sndKey2.State <> sckClosed Then _
    sndKey2.Close

    sndKey2.Accept requestID
End Sub

Private Sub tcpServer_ConnectionRequest(ByVal requestID
As Long)
If tcpServer.State <> sckClosed Then _
    tcpServer.Close

    Dim ask As String
    ask = MsgBox("Connected to Client ?", vbYesNo,
Me.Caption)
    If ask = vbYes Then
        tcpServer.Accept requestID
        lblStatus.Caption = "Status   " & txtLogin.Text & "
connected !"
        CmdDis.Enabled = True
        txtMsg.Enabled = True
        CmdSend.Enabled = True
        cmdStart.Enabled = False
        txtPln.Enabled = True
        cmdSv.Enabled = True
        cmdEn.Enabled = True
        chkHash.Enabled = True
        chkSplit.Enabled = True
        tcpServer.SendData "code accepted 1"
        Sleep 500
        sndKey1.SendData txtKac.Text
        Sleep 500
        sndKey2.SendData txtKan.Text
        shpCon.FillColor = vbGreen
    Else
        tcpServer.Accept requestID
        tcpServer.SendData "code rejected 0"
        lblStatus.Caption = "Status :  " & " Connection Rejected
!"
        CmdDis.Enabled = False
        txtMsg.Enabled = False
        CmdSend.Enabled = False
        cmdStart.Enabled = True
        txtPln.Enabled = False
        cmdSv.Enabled = False
        cmdEn.Enabled = False
        chkHash.Enabled = False
        chkSplit.Enabled = False
        shpCon.FillColor = vbRed
        Timer1.Interval = 2000
    End If
End Sub

Private Sub tcpServer_DataArrival(ByVal bytesTotal As
Long)
    Dim strData As String
    tcpServer.GetData strData

If strData = "code rejected 0" Then
    lblStatus.Caption = "Status : Rejected Remotely !"
    tcpServer.Close
    sndKey1.Close
    sndKey2.Close
    Winsock1.Close
    Winsock2.Close
    tcpServer.LocalPort = txtLocalPort.Text
    tcpServer.Listen
    sndKey1.Listen
    sndKey2.Listen
    Winsock1.Listen
    Winsock2.Listen
    shpCon.FillColor = vbRed
    CmdDis.Enabled = False
    txtMsg.Enabled = False
    CmdSend.Enabled = False
    CmdDis.Enabled = False
    txtPln.Enabled = False
    cmdSv.Enabled = False
    cmdEn.Enabled = False
    chkHash.Enabled = False
    chkSplit.Enabled = False
    Exit Sub
    Exit Sub
End If

If chkSplit.Value = 1 Or chkHash.Value = 1 Then
    txtCpr.Text = ""
    txtCpr.Text = txtCpr.Text & strData
Else
    txtCpr.Text = ""
    txtCpr.Text = txtCpr.Text & strData
    Call Dekripsi(strData)
    End If
End Sub

Private Sub Timer1_Timer()
    tcpServer.LocalPort = txtLocalPort.Text
    tcpServer.Listen
    sndKey1.Listen
    sndKey2.Listen
    Winsock1.Listen
    Winsock2.Listen
    Timer1.Interval = 0
End Sub

Private Sub txtCpr_Click()
txtCpr.SelStart = 0
txtCpr.SelLength = Len(txtCpr.Text)
End Sub

Private Sub txtLocalPort_Click()
txtLocalPort.SelStart = 0
txtLocalPort.SelLength = Len(txtLocalPort.Text)
End Sub

Private Sub txtLogin_Change()
If Len(txtLogin.Text) <= 0 Then
cmdStart.Enabled = False
Else
cmdStart.Enabled = True
End If
End Sub

Private Sub txtLogin_Click()
txtLogin.SelStart = 0
txtLogin.SelLength = Len(txtLogin.Text)
End Sub

Private Sub txtLogin_LostFocus()
    txtLogin.Text = UCase(txtLogin.Text)
End Sub
```

```
Private Sub txtPln_Click()
txtPln.SelStart = 0
txtPln.SelLength = Len(txtPln.Text)
End Sub

Private Sub txtPro_Change()
txtPro.SelStart = Len(txtPro.Text)
End Sub
'Prosedur Enkripsi
Private Sub Enkripsi(pcText As String)
   Dim Biner As String
   Dim Desimal1 As String
   Dim Desimal2 As String
   Dim N As Integer
   Dim Temp As String
   Dim nTemp As Double

   'Pesan
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Pesan = '" & pcText & "'"

   'Sleep 500

   'Ubah menjadi biner
   Biner = ""
   For N = 1 To Len(pcText)
      Biner = Biner &
FormatS(DecToBiner(Asc(Mid(pcText, N, 1))), "0", 8)
   Next N
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Ubah pesan menjadi biner:"
   'Sleep 500
   txtPro.Text = txtPro.Text & vbCrLf & Biner
   'Sleep 500

   'Tambah Nol
   If Len(Biner) Mod 3 <> 0 Then
      Biner = Biner & String(3 - (Len(Biner) Mod 3), "0")
   End If

   'Penggal setiap 3 bit biner dan ubah menjadi desimal
   Desimal1 = ""
   For N = 1 To Len(Biner) Step 3
      Temp = Mid(Biner, N, 3)
      If Len(Temp) <> 3 Then
         Temp = FormatS(Temp, "0", 3)
      End If
      Desimal1 = Desimal1 & Format(BinerToDec(Temp))
   Next N
   'Tulis
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Ubah setiap 3 bit biner menjadi bentuk
desimal:"
   'Sleep 500
   txtPro.Text = txtPro.Text & vbCrLf & Desimal1
   'Sleep 500

   'Ambil 4 digit desimal, masukkan ke fungsi
enkripsi/dekripsi
   'Tulis
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
   "Masukkan setiap 4 digit desimal (m) ke fungsi " & _
   "enkripsi: c = (m ^ e) mod n"
   Desimal2 = ""
   For N = 1 To Len(Desimal1) Step 4

      'Fungsi enkripsi
      Temp = Mid(Desimal1, N, 4)
      If Len(Temp) <> 4 Then Temp = Temp & String(4 -
Len(Temp), "0")
      nTemp = FastExp(Val(Temp), Val(txtKma),
Val(txtKmn))

      'Hasil enkripsi
      If Desimal2 <> "" Then Desimal2 = Desimal2 & " "
      Desimal2 = Desimal2 & Format(nTemp)

      'Tulis
      txtPro.Text = txtPro.Text & vbCrLf & _
         "c = (" & Temp & " ^ " & _
         txtKma & ") mod " & txtKan & " = " & nTemp

      'Sleep 500
   Next N

   'Tulis
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
   "Hasil enkripsi: " & vbCrLf & Desimal2 & vbCrLf

   'Hasil enkripsi
   HE = Desimal2
   txtMsg.Text = HE
   txtCpr.Text = HE
End Sub

'Prosedur Dekripsi
Private Sub Dekripsi(pcText As String)
   Dim Biner As String
   Dim Desimal1 As String
   Dim Desimal2() As String
   Dim N As Integer
   Dim Temp As String
   Dim nTemp As Double

   'Pesan
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Cipher Text = '" & pcText & "'"

   'Sleep 500

   'Masukkan ke fungsi dekripsi
   'Tulis
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
   "Masukkan cipher text (c) ke fungsi dekripsi: m = (c ^ d)
mod n"
   Desimal1 = ""
   Desimal2 = Split(pcText, " ")
   For N = 0 To UBound(Desimal2)
      If Trim(Desimal2(N)) <> "" Then
         'Fungsi dekripsi
         nTemp = FastExp(Val(Desimal2(N)), Val(txtKad),
Val(txtKan))

         'Hasil dekripsi
         Temp = Format(nTemp)
         If Len(Temp) < 4 Then Temp = String(4 -
Len(Temp), "0") & Temp
         Desimal1 = Desimal1 & Temp

         'Tulis
         txtPro.Text = txtPro.Text & vbCrLf & _
         "m = (" & Desimal2(N) & " ^ " & _
         txtKad & ") mod " & txtKan & " = " & Temp

         'Sleep 500
      End If
   Next N

   'Tulis
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
   "Hasil dekripsi: " & vbCrLf & Desimal1
   'Sleep 500

   'Ubah ke biner
   Biner = ""
   For N = 1 To Len(Desimal1)
      Biner = Biner &
FormatS(DecToBiner(Val(Mid(Desimal1, N, 1))), "0", 3)
   Next N
   txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
            "Ubah pesan menjadi biner:"
   'Sleep 500
   txtPro.Text = txtPro.Text & vbCrLf & Biner
   'Sleep 500

   'Ubah ke ascii
   Temp = ""
   For N = 1 To Len(Biner) Step 8
```

```
    Temp = Temp & Chr(BinerToDec(Mid(Biner, N, 8)))
  Next N
  txtPro.Text = txtPro.Text & vbCrLf & vbCrLf & _
        "Ubah setiap 8 bit biner ke bentuk ascii:"
  'Sleep 500
  txtPro.Text = txtPro.Text & vbCrLf & Temp & vbCrLf
  'Sleep 500

  'Hasil dekripsi
  HD = Temp
  txtPln.Text = HD
  txtPsn.Text = txtPsn.Text & vbCrLf & "CLIENT : " &
HD & vbCrLf
  txtCpr.Text = ""
End Sub

Private Sub txtRemotePort_Change()
txtLocalPort.Text = "1234"
txtLocalPort.Locked = True
End Sub

Private Sub sndKey1_DataArrival(ByVal bytesTotal As
Long)
Dim kme As String

  sndKey1.GetData kme

  txtKma.Text = ""
  txtKma.Text = txtKma.Text & kme
End Sub

Private Sub sndKey2_DataArrival(ByVal bytesTotal As
Long)
Dim kmn As String
  sndKey2.GetData kmn

  txtKmn.Text = ""
  txtKmn.Text = txtKmn.Text & kmn
End Sub

Private Sub Winsock1_ConnectionRequest(ByVal
requestID As Long)
If Winsock1.State <> sckClosed Then _
  Winsock1.Close

  Winsock1.Accept requestID
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As
Long)
  Dim dt As String
  Winsock1.GetData dt

If dt = "1" Then
  chkHash.Value = 1
Else
  chkHash.Value = 0
  End If
End Sub
Private Sub Winsock2_ConnectionRequest(ByVal
requestID As Long)
If Winsock2.State <> sckClosed Then _
  Winsock2.Close

  Winsock2.Accept requestID
End Sub

Private Sub Winsock2_DataArrival(ByVal bytesTotal As
Long)
  Dim dts As String
  Winsock2.GetData dts

If dts = "1" Then
  chkSplit.Value = 1
Else
  chkSplit.Value = 0
  End If
End Sub
```

➢ frmKey.frm

```
Option Explicit

Private A As Integer
Private Proses As Boolean

Private Sub cmdHitung_Click()
  Dim nA As Integer
  Dim pq As Double
  Dim d As Double
  Dim N As Double

  'Tes rabin miller untuk A = 8
  nA = 8

  'Periksa kunci Client
  If TestPrima(Val(txtpA.Text), nA) = False Then
    MsgBox "Nilai p Client harus bilangan prima !",
vbCritical
    Exit Sub
  ElseIf TestPrima(Val(txtqA.Text), nA) = False Then
    MsgBox "Nilai q Client harus bilangan prima !",
vbCritical
    Exit Sub
  ElseIf GCD(Val(txteA.Text), (Val(txtpA.Text) - 1) *
(Val(txtqA.Text) - 1)) <> 1 Then
    MsgBox "Nilai GCD(e,(p-1)(q-1)) untuk Client harus
sama dengan 1 !", vbCritical
    Exit Sub
  Else
    'Cek Fast Exp -> ok
    pq = (Val(txtpA.Text) - 1) * (Val(txtqA.Text) - 1)
    d = ExtendedEuclidean(Val(txteA.Text), pq)
    N = Val(txtpA.Text) * Val(txtqA.Text)
    If FastExp(FastExp(100, Val(txteA.Text), N), d, N) <>
100 Then
      MsgBox "Kunci Client tidak valid, karena tidak
dapat mengembalikan " & _
          "nilai setelah operasi Fast Exponentiation !",
vbCritical
      Exit Sub
    End If
  End If

  txtnA.Text = Val(txtpA.Text) * Val(txtqA.Text)

  pq = (Val(txtpA.Text) - 1) * (Val(txtqA.Text) - 1)
  d = ExtendedEuclidean(Val(txteA.Text), pq)
  txtdA.Text = d

End Sub

Private Sub cmdRndA_Click()
Dim nDigit As Integer
  Dim nTemp As Double
  Dim nE As Double
  Dim N As Double
  Dim d As Double

RandomUlang:

  'Banyak digit bilangan p
  Randomize Timer
  nDigit = 3 + Int(Rnd * 3)
  txtpA = GeneratePrimeNumber(nDigit)

  'Banyak digit bilangan q
  Randomize Timer
  nDigit = 3 + Int(Rnd * 3)
  txtqA = GeneratePrimeNumber(nDigit)
```

```
    '----------
    'Cari e
    '----------
    '(p-1)(q-1)
    nTemp = (Val(txtpA.Text) - 1) * (Val(txtqA.Text) - 1)

    Randomize Timer
    nE = (10 ^ (nDigit - 1)) + Int(Rnd * (Val(String(nDigit,
"9")) - (10 ^ (nDigit - 1))))
    While GCD(nE, nTemp) <> 1
        Randomize Timer
        nE = (10 ^ (nDigit - 1)) + Int(Rnd * (Val(String(nDigit,
"9")) - (10 ^ (nDigit - 1))))
    Wend

    txteA.Text = nE

    'N
    N = Val(txtpA.Text) * Val(txtqA.Text)

    'D
    d = ExtendedEuclidean(nE, nTemp)

    'Cek
    If FastExp(FastExp(100, nE, N), d, N) <> 100 Then
GoTo RandomUlang

    txtdA.Text = ""
    txtnA.Text = ""
End Sub

Private Sub cmdSaveExit_Click()
    Dim nA As Integer
    Dim pq As Double
    Dim d As Double
    Dim N As Double

    'Tes rabin miller untuk A = 8
    nA = 8

    'Periksa kunci Client
    If TestPrima(Val(txtpA.Text), nA) = False Then
        MsgBox "Nilai p Client harus bilangan prima !",
vbCritical
        Exit Sub
    ElseIf TestPrima(Val(txtqA.Text), nA) = False Then
        MsgBox "Nilai q Client harus bilangan prima !",
vbCritical
        Exit Sub
    ElseIf GCD(Val(txteA.Text), (Val(txtpA.Text) - 1) *
(Val(txtqA.Text) - 1)) <> 1 Then
        MsgBox "Nilai GCD(e,(p-1)(q-1)) untuk Client harus
sama dengan 1 !", vbCritical
        Exit Sub
    Else
        'Cek Fast Exp -> ok
        pq = (Val(txtpA.Text) - 1) * (Val(txtqA.Text) - 1)
        d = ExtendedEuclidean(Val(txteA.Text), pq)
        N = Val(txtpA.Text) * Val(txtqA.Text)
        If FastExp(FastExp(100, Val(txteA.Text), N), d, N) <>
100 Then
            MsgBox "Kunci Client tidak valid, karena tidak
dapat mengembalikan " & _
                "nilai setelah operasi Fast Exponentiation !",
vbCritical
            Exit Sub
        End If
    End If

    If txtnA.Text = "" And txtdA.Text = "" Then
        MsgBox "hitung kunci d dan kunci n !", vbCritical
    Else
    Akey.P = Val(txtpA.Text)
    Akey.Q = Val(txtqA.Text)
    Akey.e = Val(txteA.Text)
    Akey.N = Val(txtnA.Text)
    Akey.d = Val(txtdA.Text)

    Unload Me
    End If
End Sub
```

➢ Modules1.bas

```
Option Explicit

Public Ubah As Boolean

'Tipe Kunci
Public Type Kunci
    P As Double
    Q As Double
End Type

'Key's
Public Akey As Kunci

'Message
Public Msg1 As String
Public Msg2 As String
Public MsgCancel As Boolean

'Public Declare Sub Sleep Lib "kernel32" (ByVal
dwMilliseconds As Long)

'Delay
Public Sub Sleep(ByVal pnDelay As Long)
    Dim nD

    DoEvents
    nD = Timer

    While (Timer - nD) <= pnDelay / 1000
        DoEvents
    Wend
End Sub

'GCD Function
Public Function GCD(ByVal A As Double, ByVal B As
Double) As Long
    Dim P As Double
    Dim Q As Double
    Dim R As Double

    P = A
    Q = B

    While Q <> 0
        R = FModulus(P, Q)
        P = Q
        Q = R
    Wend

    GCD = P
End Function

'Pembangkitan bilangan prima dengan metode Rabin Miller
sebesar nDigit
Public Function GeneratePrimeNumber(ByVal nDigit As
Long)
    Dim bPrima As Boolean
    Dim nRandom As Long
    Dim nTotal As Integer
    Dim nA As Long
    Dim BTest80 As Boolean

    bPrima = False
    nTotal = 0

    While bPrima = False
```

```
        nTotal = nTotal + 1

        'Tentukan jumlah digit
        Randomize Timer
        nRandom = (10 ^ (nDigit - 1)) + Int(Rnd *
(Val(String(nDigit, "9")) - (10 ^ (nDigit - 1))))

        'Angka Genap -> ubah menjadi ganjil
        If nRandom Mod 2 = 0 Then nRandom = nRandom + 1

        'Test 80 persen
        BTest80 = Test80P(nRandom)
        If BTest80 Then
          '***--- VALIDASI ---***
          If nRandom <= 1 Then
            bPrima = False
          ElseIf nRandom = 2 Then
            'Bilangan Prima
            bPrima = True
          Else
            '***--- BANGKITKAN BILANGAN ACAK A ---***
            If nRandom > 16 Then
              While ((nA = 0) Or (nA = 1))
                Randomize
                nA = Int(Rnd * 15)
              Wend
            Else
              While ((nA = 0) Or (nA = 1))
                Randomize
                nA = Int(Rnd * nRandom)
              Wend
            End If

            '***--- TEST PRIMA RABIN MILLER ---***
            bPrima = TestPrima(nRandom, nA)
          End If
        End If

        'Confirm
        If nTotal = 3000 Then
          GeneratePrimeNumber = 0
          Exit Function
        End If
      Wend

      GeneratePrimeNumber = nRandom
End Function

'Algoritma untuk tes bilangan prima (persentase 80%)
Public Function Test80P(pnP As Long) As Boolean
    Dim nI As Long
    Dim bOK As Boolean

    bOK = True
    For nI = 2 To 255
      If nI = 2 Or TestPrima(CDbl(nI), 2) Then
        If pnP <> nI And (pnP Mod CDbl(nI)) = 0 Then
          bOK = False
          Exit For
        End If
      End If
    Next nI

    Test80P = bOK
End Function

'Algoritma untuk tes apakah suatu bilangan merp. prima atau tidak
Public Function TestPrima(ByVal pnP As Long, ByVal pnA As Long) As Boolean

    'Angka Genap
    If pnP Mod 2 = 0 Then
      TestPrima = False
      Exit Function
    End If

    'Test Prima
    TestPrima = IsRabinMiller(pnP, pnA)

End Function

'Algoritma untuk test Rabin-Miller
Public Function IsRabinMiller(pnP As Long, pnA As Long) As Boolean
    Dim nLoop As Long
    Dim nTemp As Long

    Dim pnC As Long
    Dim pnB As Long
    Dim pnM As Long

    Dim pnJ As Long
    Dim pnZ As Long

    pnC = pnP - 1

    '------ CARI b
    nTemp = 0
    While (pnC Mod (2 ^ nTemp)) = 0 And ((2 ^ nTemp) < pnP)
      nTemp = nTemp + 1
    Wend
    pnB = nTemp - 1

    '------ CARI m
    pnM = (pnC / (2 ^ pnB))

    pnJ = 0
    pnZ = FastExp(pnA, pnM, pnP)

    If (pnZ = 1) Or (pnZ = (pnP - 1)) Then
      IsRabinMiller = True
      Exit Function
    End If

lAgain:

    If pnJ > 0 And pnZ = 1 Then
      IsRabinMiller = False
      Exit Function
    End If

    pnJ = pnJ + 1
    If (pnJ < pnB) And (pnZ <> pnP - 1) Then
      pnZ = FastExp(pnZ, 2, pnP)
      GoTo lAgain
    End If

    If pnZ = (pnP - 1) Then
      IsRabinMiller = True
      Exit Function
    End If

    If pnJ = pnB And pnZ <> (pnP - 1) Then
      IsRabinMiller = False
      Exit Function
    End If
End Function

'Algoritma Fast Exponentiation - (A^B) mod C
Public Function FastExp(ByVal A As Double, ByVal B As Double, ByVal C As Double) As Double
    Dim Product As Double
    Dim A1 As Double
    Dim B1 As Double

    A1 = A
    B1 = B
    Product = 1
    While (B1 <> 0)
      While FModulus(B1, 2) = 0
        B1 = FDiv(B1, 2)
        A1 = FModulus(A1 * A1, C)
      Wend
      B1 = B1 - 1
```

```
        Product = FModulus(Product * A1, C)
    Wend
    FastExp = Product

End Function

Public Function FModulus(ByVal pnA1 As Double, ByVal
pnA2 As Double) As Double
    Dim pnA As Double
    Dim nMod As Double

    pnA = Abs(pnA1)
    nMod = pnA / pnA2
    FModulus = (pnA - (pnA2 * Int(nMod))) * IIf(pnA1 < 0,
-1, 1)
End Function

'Algoritma Extended Euclidean
Public Function ExtendedEuclidean(NilaiX As Double,
pnValueA11 As Double) As Double
    Dim A(3, 2) As Double
    Dim nM As Double
    Dim nX As Double
    Dim nI, nJ As Integer
    Dim bSelesai As Boolean

    bSelesai = False

    'Bentuk Array
    A(1, 1) = pnValueA11
    A(1, 2) = NilaiX

    'Matriks Identitas
    A(2, 1) = 1
    A(2, 2) = 0
    A(3, 1) = 0
    A(3, 2) = 1

    While Not bSelesai

        'Hitung nilai m
        nM = FDiv(A(1, 1), A(1, 2))

        For nI = 1 To 3

            'Hitung nilai x
            nX = A(nI, 1) - nM * A(nI, 2)

            'Ubah nilai
            A(nI, 1) = A(nI, 2)
            A(nI, 2) = nX

            If nI = 1 And nX = 0 Then

                bSelesai = True
            End If
        Next nI
    Wend
    If A(3, 1) >= 0 Then
        ExtendedEuclidean = A(3, 1)
    Else
        ExtendedEuclidean = A(3, 1) + pnValueA11
    End If

End Function

'Operasi div untuk bilangan besar
Public Function FDiv(pnA1 As Double, pnA2 As Double)
As Double
    Dim nDiv As Double

    nDiv = pnA1 / pnA2
    FDiv = Int(nDiv)

End Function

'Untuk memformat string pcText sepanjang pnLength,
kosong diganti dengan pcZeroText
Public Function FormatS(ByVal pcText As String,
pcZeroText As String, pnLength As Long) As String
    If Len(pcText) > pnLength Then
        'Jika lebih besar, maka cut
        pcText = Left(pcText, pnLength)
    ElseIf Len(pcText) < pnLength Then
        'Jika lebih kecil, maka tambah
        FormatS = String(pnLength - Len(pcText),
pcZeroText) & pcText
    Else
        FormatS = pcText
    End If
End Function

'DESIMAL KE BINER
Public Function DecToBiner(ByVal pnAngka As Double)
As String
    Dim nLoop As Double
    Dim nHasilBagi As Double
    Dim nSisaBagi As Double
    Dim cBiner1 As String
    Dim cBiner2 As String

    nHasilBagi = pnAngka
    While nHasilBagi <> 0
        nSisaBagi = FModulus(nHasilBagi, 2)
        cBiner1 = cBiner1 & Format(nSisaBagi)
        nHasilBagi = FDiv(nHasilBagi, 2)
    Wend
    If cBiner1 = "" Then cBiner1 = "0"

    'Ambil Terbalik
    For nLoop = Len(cBiner1) To 1 Step -1
        cBiner2 = cBiner2 & Mid(cBiner1, nLoop, 1)
    Next nLoop

    'Angka Biner
    DecToBiner = cBiner2
End Function

'BINER KE DESIMAL
Public Function BinerToDec(pcText As String) As Double
    Dim nLoop As Double
    Dim nHasil As Double

    nHasil = 0
    'Konversi dari belakang
    For nLoop = Len(pcText) To 1 Step -1
        If Mid(pcText, nLoop, 1) = "1" Then
            nHasil = nHasil + (2 ^ (Len(pcText) - nLoop))
        End If
    Next nLoop

    'Angka Desimal
    BinerToDec = nHasil
End Function
```

➢ SHA-1.bas

```
Option Explicit

Private Declare Function GetInputState Lib "user32" () As
Long

Public Function Sha1(hashthis As String) As String
Dim buf(0 To 4) As String
Dim Xin(0 To 79) As String
Dim tempnum As Integer, tempnum2 As Integer
Dim loopit As Integer, loopouter As Integer
Dim loopinner As Integer, A As String
Dim B As String, C As String
Dim d As String, e As String, tempstr As String
Dim Outp As String

    ' Add padding
```

```
    tempnum = 8 * Len(hashthis)
    hashthis = hashthis + Chr$(128) 'Add binary 10000000
    tempnum2 = 56 - Len(hashthis) Mod 64

    If tempnum2 < 0 Then
       tempnum2 = 64 + tempnum2
    End If

    hashthis = hashthis + String$(tempnum2, Chr$(0))

    For loopit = 1 To 8
       tempstr = Chr$(tempnum Mod 256) + tempstr
       tempnum = tempnum - tempnum Mod 256
       tempnum = tempnum / 256
    Next loopit

    hashthis = hashthis + tempstr

    ' For each 512 bit section
    For loopouter = 0 To Len(hashthis) / 64 - 1
       A = buf(0)
       B = buf(1)
       C = buf(2)
       d = buf(3)
       e = buf(4)

       ' Get the 512 bits
       For loopit = 0 To 15
          Xin(loopit) = ""
          For loopinner = 4 To 1 Step -1
             Xin(loopit) = Hex(Asc(Mid$(hashthis, 64 *
loopouter + 4 * loopit + loopinner, 1))) + Xin(loopit)
          If Len(Xin(loopit)) Mod 2 Then Xin(loopit) = "0"
+ Xin(loopit)
          Next loopinner

          If GetInputState() <> 0 Then
             DoEvents
          End If

       Next loopit

       For loopit = 16 To 79
          Xin(loopit) = RotLeft(BXor(BXor(BXor(Xin(loopit
- 3), Xin(loopit - 8)), Xin(loopit - 14)), Xin(loopit - 16)), 1)

          If GetInputState() <> 0 Then
             DoEvents
          End If

       Next loopit

       For loopit = 0 To 19
          tempstr = Bor(BAnd(B, C), BAnd(BNot(B), d))
          tempstr = BMod32Add(RotLeft(A, 5),
BMod32Add(tempstr, BMod32Add(e,
BMod32Add(Xin(loopit), "5A827999"))))
          e = d
          d = C
          C = RotLeft(B, 30)
          B = A
          A = tempstr
       Next loopit

       For loopit = 20 To 39
          tempstr = BXor(BXor(B, C), d)
          tempstr = BMod32Add(RotLeft(A, 5),
BMod32Add(tempstr, BMod32Add(e,
BMod32Add(Xin(loopit), "6ED9EBA1"))))
          e = d
          d = C
          C = RotLeft(B, 30)
          B = A
          A = tempstr


          If GetInputState() <> 0 Then
             DoEvents
          End If

       Next loopit

       For loopit = 40 To 59
          tempstr = Bor(Bor(BAnd(B, C), BAnd(B, d)),
BAnd(C, d))
          tempstr = BMod32Add(RotLeft(A, 5),
BMod32Add(tempstr, BMod32Add(e,
BMod32Add(Xin(loopit), "8F1BBCDC"))))
          e = d
          d = C
          C = RotLeft(B, 30)
          B = A
          A = tempstr


          If GetInputState() <> 0 Then
             DoEvents
          End If


       Next loopit

       For loopit = 60 To 79
          tempstr = BXor(BXor(B, C), d)
          tempstr = BMod32Add(RotLeft(A, 5),
BMod32Add(tempstr, BMod32Add(e,
BMod32Add(Xin(loopit), "CA62C1D6"))))
          e = d
          d = C
          C = RotLeft(B, 30)
          B = A
          A = tempstr


          If GetInputState() <> 0 Then
             DoEvents
          End If

       Next loopit

       buf(0) = BMod32Add(buf(0), A)
       buf(1) = BMod32Add(buf(1), B)
       buf(2) = BMod32Add(buf(2), C)
       buf(3) = BMod32Add(buf(3), d)
       buf(4) = BMod32Add(buf(4), e)


       If GetInputState() <> 0 Then
          DoEvents
       End If


    Next loopouter

    ' Extract Hash
    hashthis = ""
    For loopit = 0 To 4
       For loopinner = 0 To 3
          hashthis = hashthis + Hex(Val("&H" +
Mid$(buf(loopit), 1 + 2 * loopinner, 2)))
       Next loopinner

       If GetInputState() <> 0 Then
          DoEvents
       End If

    Next loopit

    ' And return it
    Sha1 = hashthis

End Function

Private Function RotLeft(ByVal value1 As String, ByVal
rots As Integer) As String
Dim tempstr As String
Dim loopit As Integer, loopinner As Integer
Dim tempnum As Integer
```

```
    rots = rots Mod 32

    If rots = 0 Then
        RotLeft = value1
        Exit Function
    End If

    value1 = Right$(value1, 8)
    tempstr = String$(8 - Len(value1), "0") + value1
    value1 = ""

    ' Convert to binary
    For loopit = 1 To 8
        tempnum = Val("&H" + Mid$(tempstr, loopit, 1))
        For loopinner = 3 To 0 Step -1
            If tempnum And 2 ^ loopinner Then
                value1 = value1 + "1"
            Else
                value1 = value1 + "0"
            End If
        Next loopinner
    Next loopit
    tempstr = Mid$(value1, rots + 1) + Left$(value1, rots)

    ' And convert back to hex
    value1 = ""
    For loopit = 0 To 7
        tempnum = 0
        For loopinner = 0 To 3
            If Val(Mid$(tempstr, 4 * loopit + loopinner + 1, 1))
Then
                tempnum = tempnum + 2 ^ (3 - loopinner)
            End If
        Next loopinner
        value1 = value1 + Hex(tempnum)
    Next loopit

    RotLeft = value1
End Function

Private Function BXor(ByVal value1 As String, ByVal
value2 As String) As String
Dim valueans As String
Dim loopit As Integer, tempnum As Integer

    tempnum = Len(value1) - Len(value2)
    If tempnum < 0 Then
        valueans = Left$(value2, Abs(tempnum))
        value2 = Mid$(value2, Abs(tempnum) + 1)
    ElseIf tempnum > 0 Then
        valueans = Left$(value1, Abs(tempnum))
        value1 = Mid$(value1, tempnum + 1)
    End If

    For loopit = 1 To Len(value1)
        valueans = valueans + Hex(Val("&H" + Mid$(value1,
loopit, 1)) Xor Val("&H" + Mid$(value2, loopit, 1)))
    Next loopit

    BXor = valueans
End Function

Private Function Bor(ByVal value1 As String, ByVal
value2 As String) As String
Dim valueans As String
Dim loopit As Integer, tempnum As Integer

    tempnum = Len(value1) - Len(value2)
    If tempnum < 0 Then
        valueans = Left$(value2, Abs(tempnum))
        value2 = Mid$(value2, Abs(tempnum) + 1)
    ElseIf tempnum > 0 Then
        valueans = Left$(value1, Abs(tempnum))
        value1 = Mid$(value1, tempnum + 1)
    End If

    For loopit = 1 To Len(value1)
        valueans = valueans + Hex(Val("&H" + Mid$(value1,
loopit, 1)) Or Val("&H" + Mid$(value2, loopit, 1)))
    Next loopit

    Bor = valueans
End Function

Private Function BAnd(ByVal value1 As String, ByVal
value2 As String) As String
Dim valueans As String
Dim loopit As Integer, tempnum As Integer

    tempnum = Len(value1) - Len(value2)
    If tempnum < 0 Then
        value2 = Mid$(value2, Abs(tempnum) + 1)
    ElseIf tempnum > 0 Then
        value1 = Mid$(value1, tempnum + 1)
    End If

    For loopit = 1 To Len(value1)
        valueans = valueans + Hex(Val("&H" + Mid$(value1,
loopit, 1)) And Val("&H" + Mid$(value2, loopit, 1)))
    Next loopit

    BAnd = valueans
End Function

Private Function BNot(ByVal value1 As String) As String
Dim valueans As String
Dim loopit As Integer

    value1 = Right$(value1, 8)
    value1 = String$(8 - Len(value1), "0") + value1
    For loopit = 1 To 8
        valueans = valueans + Hex(15 Xor Val("&H" +
Mid$(value1, loopit, 1)))
    Next loopit

    BNot = valueans
End Function

Private Function BMod32Add(ByVal value1 As String,
ByVal value2 As String) As String
    BMod32Add = Right$(BAdd(value1, value2), 8)
End Function

Private Function BAdd(ByVal value1 As String, ByVal
value2 As String) As String
Dim valueans As String
Dim loopit As Integer, tempnum As Integer

    tempnum = Len(value1) - Len(value2)
    If tempnum < 0 Then
        value1 = Space$(Abs(tempnum)) + value1
    ElseIf tempnum > 0 Then
        value2 = Space$(Abs(tempnum)) + value2
    End If

    tempnum = 0
    For loopit = Len(value1) To 1 Step -1
        tempnum = tempnum + Val("&H" + Mid$(value1,
loopit, 1)) + Val("&H" + Mid$(value2, loopit, 1))
        valueans = Hex(tempnum Mod 16) + valueans
        tempnum = Int(tempnum / 16)
    Next loopit

    If tempnum <> 0 Then
        valueans = Hex(tempnum) + valueans
    End If

    BAdd = valueans
End Function
```