KRIPTOGRAFI PADA *FILE* DOKUMEN *MICROSOFT OFFICE* MENGGUNAKAN METODE RSA

SKRIPSI

LABORATURIUM KOMPUTASI CERDAS

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

Aditya Permana

NIM : 0810963028

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNIK INFORMATIKA DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

KRIPTOGRAFI PADA *FILE* DOKUMEN *MICROSOFT OFFICE* MENGGUNAKAN METODE RSA

SKRIPSI

LABORATURIUM KOMPUTASI CERDAS

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

Aditya Permana

NIM : 0810963028

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMATIKA DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PENGESAHAN

KRIPTOGRAFI PADA *FILE* DOKUMEN *MICROSOFT OFFICE* MENGGUNAKAN METODE RSA

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :

ADITYA PERMANA

NIM : 0810963028

Skripsi ini telah disetujui

Juni 2013

Pembimbing I,	Pembimbing II,
Edy Santoso,S.Si., M.Kom	**Dian Eka Ratnawati, S.Si, M.Kom**
NIP. 197404142003121004	**NIP.197306192002122001**

LEMBAR PENGESAHAN

KRIPTOGRAFI PADA *FILE* DOKUMEN *MICROSOFT OFFICE* MENGGUNAKAN METODE RSA

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar Sarjana
dalam bidang Ilmu Komputer

Disusun Oleh:

ADITYA PERMANA

NIM.0810963028

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 2 Juli 2013

Penguji I	Penguji II
Candra Dewi, S.Kom., M.Sc	**Ahmad Afif Supianto, S.Si., M.Kom**
NIP. 19771114 200312 2 001	**NIP. 820623 16 1 1 0425**

Penguji III

Imam Cholissodin, S.Si., M.Kom

NIP. 850719 16 1 1 0422

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji., M.T.

NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya yang bertanda tangan di bawah ini :

Nama : Aditya Permana

NIM : 0810963028

Program Studi : Informatika / Ilmu Komputer

Penulis skripsi berjudul : Kriptografi Pada *File* Dokumen *Microsoft Office* Menggunakan Metode RSA

Dengan ini menyatakan bahwa :

1. Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termasuk di isi dan tertulis di daftar pustaka dalam Skripsi ini.
2. Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran

Malang, 10 Juni 2013

Yang menyatakan,

Aditya Permana

NIM. 0810963028

KRIPTOGRAFI PADA *FILE* DOKUMEN *MICROSOFT OFFICE* MENGGUNAKAN METODE RSA

ABSTRAK

Dokumen merupakan suatu file yang digunakan untuk menyimpan atau mendokumentasikan hal yang sangat penting. Setiap perusahaan tidak dapat terlepas dari dokumen file dalam menjalankan dan mengelola data-data penting, oleh karena itu, keamanan suatu dokumen file sangat diutamakan. Diperlukan suatu aplikasi khusus untuk mengamankan dokumen file dengan tingkat keamanan yang tinggi. Aplikasi kriptografi merupakan salah satu alternatif yang dapat digunakan untuk memenuhi kebutuhan keamanan data yang sangat tinggi. Secara pengertian kriptografi merupakan ilmu yang mempelajari tentang penyandian data. Dalam implementasinya, kriptografi melakukan proses enkripsi dan dekripsi. Proses enkripsi adalah proses mengubah plaintext menjadi chipertext. Sedangkan proses dekripsi adalah proses mengubah chipertext menjadi plaintext. Diperlukan metode yang dapat melakukan proses perubahan plaintext menjadi chipertext dan sebaliknya. Maka pada penelitian ini dipilihlah metode RSA untuk melakukan penyandian terhadap dokumen file tersebut. Metode ini menggunakan dua kunci yang berbeda, yaitu kunci privat dan kunci publik. Kedua kunci ini merupakan bilangan prima. Kelebihan dari metode RSA adalah dalam hal pembangkitan kunci. Metode RSA menggunakan bilangan prima besar dalam pembuatan kunci, sehingga sulit untuk ditebak. Kunci publik digunakan untuk proses enkripsi dengan cara mengubah plaintext menjadi chipertext. Plaintext yang digunakan dalam penelitian ini adalah kumpulan byte yang terdapat dalam file dokumen. Setelah file byte dirubah ke bentuk chipertext selanjutnya menuliskan chipertext tersebut ke dalam file dokumen. Hasil dari proses enkripsi adalah file dokumen yang dienkripsi tidak dapat dibuka. Untuk dapat membuka file dokumen dilakukan proses dekripsi yang mengubah kumpulan byte file tersebut ke bentuk semula dengan menggunakan kunci privat. Semakin besar nilai dari sebuah kunci privat, maka semakin sulit pula untuk dibongkar atau ditebak.

Kata Kunci : Kriptografi, RSA, Enkripsi, Dekripsi, *File Document MS.Office*

CRYPTOGRAPHY ON MICROSOFT OFFICE FILE DOCUMENTS USING RSA

ABSTRACT

Document is a file that used to storing or documenting something that very important. Every company can not be separated from the document file when running and managing important data, therefore, the security of a document file is priorited. It is required a special application for secure file documents with high levels of security. Cryptographic application is one of alternative application that can be used to fill the necessity of high data security. In terms,cryptography is the study of data encryption. In implementation,this cryptographic conducts decryption and encryption process. Encryption is the process of changing plaintext into ciphertext. While the decryption process is the process of changing ciphertext into plaintext. Therefore we need a method that is used to make the process of changing plaintext into ciphertext and conversely. So in this research was chosen RSA method to encoding of the document file. This method uses two different keys, there are private key and public key. Both of keys are large prime number. The advantage of RSA method is in the case of evocation of key. RSA method uses large prime numbers in the key generation, so it is difficult to predict. Public key is used to encryption process by altering original data or plaintext become chipertext. Plaintext which used in this research is a set of bytes which there are in the document file. After file of byte altered into chipertext, the next step is to write down again the chipertext into document file. The results of this encryption process is encrypted document file can not be opened. To be able to open the document file is by doing the process of decryption that change the file back to its original shape by using a private key. The greater the value of a private key, the more difficult it is to be demolished or guessed.

Keyword : Cryptography, RSA, Encryption, Decryption, File Document MS. Office

KATA PENGANTAR

Segala puji syukur ke hadirat Tuhan yang Maha Esa yang telah melimpahkan rahmat serta berkatnya-Nya, sehingga penyusun dapat menyelesaikan laporan skripsi dengan judul **"KRIPTOGRAFI PADA *FILE* DOKUMEN *MICROSOFT OFFICE* MENGGUNAKAN METODE RSA"** yang disusun guna memenuhi salah satu syarat menyelesaikan studi pada Ilmu Komputer Universitas Brawijaya dengan lancar.

Terselesaikannya laporan penelitian skripsi ini tentu tidak lepas dari bantuan beberapa pihak yang selalu mendukung baik secara materi maupun moral, oleh karena itu penyusun ingin menyampaikan ucapan terima kasih kepada :

1. Edy Santoso, S.Si., M.Kom., selaku Dosen pembimbing I yang telah bersedia meluangkan waktu, pengarahan, dan dukungannya dalam penulisan skripsi ini.
2. Dian Eka Ratnawati, S.Si., M.Kom., selaku Dosen pembimbing II yang banyak memberikan bimbingan, nasihat dan arahannya dalam penulisan skripsi ini.
3. Drs. Mardji, M.T., selaku Ketua Prodi Teknik Informatika Universitas Brawijaya
4. Kedua orang tua saya yang senantiasa memberikan dukungan serta doa untuk menyelesaikan penulisan skripsi ini.
5. Rekan-rekan di Program Studi Ilmu Komputer Universitas Brawijaya terutama rekan satu angkatan yang banyak memberikan bantuan dan tenaganya demi kelancaran skripsi ini.
6. Yeeryzkhe Githasari Lieztyanto yang selalu mendukung secara moral serta perhatiannya dan membantu dalam pemahaman rumus matematika dalam skripsi ini.
7. Semua pihak yang membantu dalam penyusunan laporan skripsi ini.

Penyusun berharap agar penelitian yang dilakukan dapat berguna bagi semua pihak. Skripsi ini masih banyak kekurangannya, sehingga dengan segala kerendahan hati penulis mengharapkan kritik serta saran pembaca.

Malang, 10 Juni 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCE CODE

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini penggunaan suatu aplikasi office sangat dibutuhkan, terutama di dalam dunia kerja. Aplikasi *Microsoft Office* merupakan suatu aplikasi yang masih digemari oleh beberapa perusahaan atau instansi tertentu. Ariwibowo mengatakan bahwa pada saat ini sebagian orang terbiasa menggunakan mesin pengolah kata dari *Microsoft Office* yaitu *Microsoft Office Word* [ARI-08]. Dengan semakin berkembangnya dunia teknologi dan informasi ini maka semakin meningkat pula ancaman terhadap file dokumen tersebut. Seiring dengan meningkatnya pengguna aplikasi *Microsoft Office* ini maka dibutuhkan pula cara untuk meningkatkan keamanan pada data terhadap dokumen tersebut. Hal ini dikarenakan setiap perusahaan atau instansi memiliki suatu data yang tidak semua orang boleh mengetahuinya. Oleh sebab itu, keamanan terhadap file dokumen ini sangat dibutuhkan di dunia kerja.

Banyak cara yang dilakukan untuk mengamankan data dari ancaman pihak luar yang tidak memiliki hak untuk mengolah data dokumen tersebut. Teknik penyandian merupakan salah satu solusi yang dapat digunakan untuk mengamankan data dokumen yang diinginkan. Dengan menggunakan cara ini, maka setiap data akan dirubah sedemikian hingga menjadi data yang tidak bisa dibaca. Proses penyandian suatu data yang sering juga disebut dengan Kriptografi merupakan ilmu yang mempelajari hal tersebut. Kriptografi merupakan suatu seni penyandian yang digunakan untuk merubah tulisan atau teks asli (*plaintext*) ke dalam bentuk file yang telah tersandi atau yang disebut pula *chipertext*.

Dalam prosesnya, kriptografi mempunyai dua proses penting, yaitu proses enkripsi dan proses dekripsi [ARI-08]. Proses enkripsi merupakan proses perubahan teks asli atau plaintext menjadi bentuk *chipertext*. Sedangkan proses dekripsi merupakan proses kebalikan dari proses enkripsi, yaitu proses pembentukan kembali suatu *chipertext* menjadi sebuah *plaintext* kembali.

Dibutuhkan suatu kunci atau *key* untuk melakukan suatu proses enkripsi dan dekripsi. Di dalam proses enkripsi dekripsi itu sendiri terdapat dua model, yaitu model simetris dan model asimetris. Perbedaan dari kedua model terdapat pada kunci atau *key* yang digunakan untuk proses enkripsi dan dekripsi. Model simetris merupakan model kriptografi yang menggunakan kunci yang sama pada saat enkripsi dan dekripsi, sedangkan model asimetris, merupakan model kriptografi yang menggunakan dua kunci yang berbeda pada saat enkripsi dan pada saat proses dekripsinya. Dalam proses pengiriman kunci privat dan publik, dalam penelitian ini pengiriman kunci melalui proses pengiriman *Email* kepada *user* yang berhak untuk melakukan proses dekripsi. Proses ini dipilih karena proses pengiriman melalui *email* dinilai merupakan suatu proses yang aman. Kunci privat dan kunci publik dapat sampai ke pengguna secara tepat dan cepat.

Menurut Zainal Arifin [ARI-09], algoritma satu kunci atau yang disebut juga dengan algoritma simetris tidak cukup aman apabila diterapkan pada era masa kini dimana tingkat komputasi suatu prosesor sudah meningkat secara signifikan dibandingkan dengan lima sampai sepuluh tahun yang lalu. Sehingga pemilihan kunci asimetris pada era masa kini menjadi pilihan sebagai algoritma yang digunakan dalam proses penyandian suatu data. Salah satu contoh algoritma simetris adalah DES. Dalam algortma DES, panjang kunci adalah 56 bit, mungkin pada saat itu kunci dengan panjang 56 bit sangatlah sulit untuk ditebak atau dibongkar. Namun dengan era prosesor *quad core* pada saat ini, melakukan komputasi untuk menemukan kunci dengan panjang 56 bit tidaklah susah. Saat ini satu-satunya cara yang dilakukan untuk memecahkan sandi DES dan RSA adalah dengan melakukan *brute force* terhadap kunci yang digunakan untuk melakukan penyandian, sehingga dua model algoritma ini sangat bergantung pada panjang kunci yang digunakan untuk melakukan enkripsi dan dekripsi. Berdasarkan kajian penelitian yang telah dilakukan sebelumnya tentang kriptografi, dalam penelitian ini algoritma yang digunakan dalam proses enkripsi dan dekripsi adalah algoritma RSA dimana algoritma ini termasuk algoritma asimetris atau penggunaan dua kunci dalam proses dekripsi dan enkripsinya. Algoritma ini dibuat oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*)pada tahun 1976 yaitu, Ron Riverst, Adi Shamir dan Leonard Adleman. Keunggulan dari algortima ini

adalah tingkat kesulitanya adalah dalam memfaktorkan bilangan non prima menjadi faktor primanya. Di dalam implementasinya, algoritma RSA membangkitkan dua kunci. Yang pertama adalah kunci umum atau *public key*, kunci ini digunakan untuk melakukan enkripsi. Sedangkan kunci yang kedua adalah kunci privat atau *private key*. Kunci ini digunakan pada saat melakukan dekripsi *chipertext* menjadi *plaintext* yang asli. Untuk melakukan pembangkitan dua kunci ini, diperlukan dua buah bilangan prima yang besar. Dalam perhitungan untuk menemukan kunci tersebut, dibutuhkan beberapa teori matematika yang digunakan untuk menunjang perhitungan dalam mencari dua bilangan prima besar. Dalam penelitian Zainal Arifin [ARI-09] standart saat ini merekomendasikan ukuran 512 bit walaupun sebenarnya membongkar 256 bit saja sudah sangat sulit menggunakan komputer biasa.

Dengan tingkat keamanan yang cukup baik seperti dalam penelitian yang dilakukan Zainal Arifin [ARI-09] yang menyebutkan bahwa sejauh ini belum seorang pun yang berhasil menemukan lubang sekuriti pada RSA dan telah digunakan secara luas di bidang perbankan dan pemerintah, metode ini sangat cocok untuk kebutuhan perusahaan atau institusi yang dituntut untuk selalu menjaga kerahasiaan data – datanya. Oleh karena itu, penelitian ini mengangkat tema "Kriptografi menggunakan metode RSA pada file dokumen *Microsoft Office*".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan maka permasalahan yang akan dibahas dalam penelitian ini adalah :

1. Bagaimana menerapkan algoritma RSA ke dalam proses enkripsi dan dekripsi file dokumen *Microsoft Office.*
2. Bagaimana hasil pengujian terhadap kunci algoritma RSA yang digunakan dalam penelitian ini.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. File dokumen yang digunakan adalah file dokumen dari Microsoft Office yaitu file yang berekstensi *.doc, .ppt* dan*.xls*
2. Dalam penelitian ini pengiriman kunci menggunakan email.
3. *Email* yang digunakan dalam penelitian ini adalah *email* dari *account* google.com.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Menerapkan algoritma RSA ke dalam proses enkripsi dan dekripsi file dokumen *Microsoft Office.*
2. Menguji algoritma RSA untuk mengetahui tingkat ketahanan kunci yang digunakan untuk proses enkripsi dan dekripsi terhadap serangan dari pihak lain.

1.5 Manfaat Penelitian

Adapun manfaat yang ingin dicapai dalam penelitian ini adalah :

1. Bagi peneliti memberikan wawasan keilmuan yang berkaitan dengan Kriptografi menggunakan algoritma RSA serta implementasinya menggunakan file dokumen
2. Bagi pihak lain dan khususnya instansi atau perusahaan dapat mempunyai alternatif keamanan data menggunakan kriptografi dengan algoritma RSA yang dapat diterapkan pada file dokumen Microsoft Office.

BAB II
TINJAUAN PUSTAKA

2.1 Sejarah Kriptografi

Kriptografi merupakan suatu ilmu penyandian yang cukup tua. Ilmu ini telah digunakan oleh bangsa Yunani kuno pada tahun 400SM. Dari catatan bahwa "Penyandian Transposisi" merupakan sistem kriptografi pertama yang dimanfaatkan atau digunakan. Semenjak digunakan, ilmu ini telah menghasilkan beberapa sistem kriptografi kuno yaitu *Caesar Chiper* , *Playfair Chiper* dan *ADFVGX Chiper* yang digunakan pada perang dunia I, hingga algoritma-algoritma kriptografi rotor yang populer pada Perang Dunia II , seperti Sigaba / M-134 (Amerika Serikat), Typex (Inggris), Purple (Jepang), dan mesin kriptografi legendaris Enigma (Jermn) [HID-03].

Pada tahun 1980, kriptografi awalnya digunakan untuk dunia militer dan komunikasi publik. Namun dengan semakin berkembang pesatnya suatu komunikasi data dan teknologi internet, maka ilmu ini semakin populer dikalangan umum tidak hanya di dunia militer saja. Awal mula ilmu ini terjadi pada tahun 1977 dimana algoritma DES atau *Data Encryption Standard* dibuat menjadi suatu standar sebagai algoritma penyandian. Kekuatan dari algoritma DES terdapat pada panjang kuncinya yaitu 56-bit. Namun menurut Taufik Hidayat (2008) dengan semakin berkembangnya dunia teknologi baik dalam segi *hardware* maupun *software*, membuat algotritma ini menjadi tidak aman dan mudah dibajak hanya dalam beberapa hari. Maka dengan seiring perkambangan ilmu penyandian, ditemukanlah sebuah algoritma yang lebih kuat lagi dalam mengenkripsi dengan menggunakan dua buah kunci untuk menyandikan suatu data, algoritma tersebut adalah algoritma RSA [ARI-09].

Aloritma RSA ditemukan oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*) yaitu Ron Rivest, Adi Shamir, dan Len Adleman. Algoritma ini menggunakan dua kunci dalam melakukan enkripsi dan dekripsi.

2.2 Kriptografi secara umum

Kriptografi berasal dari kata Yunani yaitu *Crypto* yang artinya rahasia dan *Graphia* yang mempunyai arti tulisan. Sehingga kriptografi disebut juga tulisan yang rahasia atau di dalam ilmu komputer, kriptografi adalah suatu seni sekaligus ilmu yang digunakan untuk menjaga suatu keamanan pesan. Seseorang yang melakukan suatu kriptografi disebut kriptografer. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data [MOR-11].

Di dalam proses kriptografi, terdapat beberapa proses yaitu enkripsi dan dekripsi. Enkripsi merupakan suatu proses pembuatan *chipertext*. *Chipertext* adalah data atau tulisan yang telah dienkripsi dan tidak dapat dibaca oleh manusia biasa karena karakternya berupa karakter acak. Proses enkripsi dimulai dengan mengubah *plaintext* atau data asli ke data *chipertext* dengan menggunakan beberapa algoritma enkripsi. Sedangkan proses dekripsi merupakan proses dimana *chipertext* akan dikembalikan kedalam bentuk *plaintext* kembali dengan cara menggunakan algoritma yang sama seperti proses enkripsi. Sedangkan di dalam proses enkripsi dan dekripsi tersebut, terdapat suatu proses yang sangat penting untuk proses enkripsi dan dekripsi, yaitu pembentukan suatu kunci yang akan digunakan untuk melakukan proses enkripsi dan digunakan juga pada saat proses dekripsi. Di dalam dunia kriptografi, proses dekripsi juga dapat disebut sebagai *cryptoanalysis*, yaitu suatu ilmu yang digunakan untuk memecahkan suatu sandi kriptografi menjadi sebuah *plaintext*.

Menurut Menezes [MEN-96] pada tahun 1996, di dalam kriptografi terdapat beberapa tujuan utama yang akan dicapai dalam hal keamanan data atau komunikasi, yaitu :

1. Kerahasiaan merupakan suatu tujuan yang paling utama dan paling penting di dalam ilmu kriptografi. Keamanan suatu data sangat diutamakan dalam kriptografi. Hanya orang atau pihak-pihak tertentu saja yang dapat membaca atau memecahkan suatu sandi dari sebuah kriptografi.

2. Autentukasi atau identifikasi pengguna merupakan tujuan kedua yang juga penting dalam kriptografi. Dua pengguna yang saling bertukar data harus dapat mengenali satu sama lain. Data yang dikirim harus diidentifikasi keaslianya, isi data dan yang lainnya.
3. Integritas Data adalah suatu hal yang mutlak di dalam sebuah kriptografi. Data yang disandikan harus memiliki intregitas data yang baik. Ini dikarenakan dalam setiap penyadian data dan pengembalian data ke dalam bentuk aslinya, harus tidak terjadi penyisipan, perubahan atau pensubtitusaian atau penukaran isi data. Data asli yang akan dienkripsi harus sama dengan data yang hasil dekripsi, supaya tidak terjadi kesalahan komunikasi antar kedua pengguna.
4. *Non-Repudiation* adalah pencegahan penolakan atau penyangkalan pesan terhadap suatu informasi atau data oleh yang membuat atau mengirimkan data atau informasi tersebut.

Dalam prosesnya, ilmu kriptografi menggunakan dua macam algoritma dalam enkripsi dan dekripsi suatu data, yaitu algoritma asimetris *(asymmetric algorithms)* dan algoritma simetris *(symetric algorithms)*.

2.3.1 Algoritma Simetris (Symmetric Algorithms)

Algoritma simetris merupakan algoritma kriptografi yang paling sederhana. Dalam penggunaannya algoritma ini hanya membutuhkan satu kunci untuk melakukan proses enkripsi dan dekripsi atau dengan kata lain pengguna dari algoritma kriptografi ini harus memiliki kunci yang sama pada saat digunakan dalam proses enkripsi dan pada saat proses enkripsi. Oleh karena itu algoritma ini juga disebut sebagai algoritma yang sederhana namun kurang aman pada proses implementasinya karena siapapun yang mempunyai kunci enkripsinya maka dapat dipastikan pihak tersebut juga dapat membongkar *chipertext* yang telah dienkripsi. Berikut adalah proses algoritma simetris :

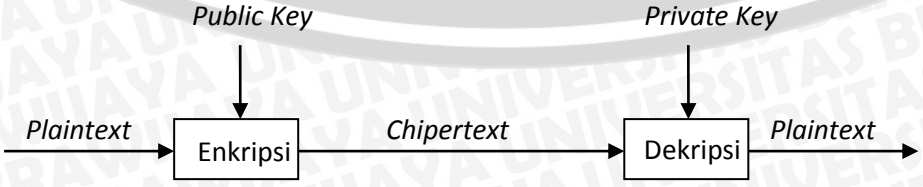
Gambar 2.1 Algoritma Simetris

Dalam gambar 2.1 dijelaskan bahwa data asli atau *plaintext* akan di enkripsi menggunakan sebuah kunci atau *key*. Lalu data tersebut akan menjadi sebuah data acak atau yang disebut juga dengan *chipertext* . Selanjutnya pada saat proses pengembalian ke data asli atau dekripsi, kunci yang digunakan untuk merubah data acak tersebut identik atau sama dengan kunci yang digunakan untuk melakukan proses enkripsi. Dalam algoritma simetris terdapat dua kategori, yaitu kategori algoritma aliran dan algoritma blok. Pada algoritma aliran *(Stream Chipers)* proses penyandianya berorientasi pada satu bit atau satu byte data. Sedangkan pada algoritma blok *(Block Chipers)* proses penyandianya berorientasi pada sekumpilan bit atau byte data [RIY-08].

2.3.2 Algoritma Asimetris (Asymmetric Algorithms)

Algoritma Asimetris atau yang disebut juga sebagai *public key chryptography* merupakan algoritma yang menggunakan dua buah kunci. Algoritma ini pertama kali dipublikasikan oleh Diffie dan Hellman pada tahun 1976. Berbeda dengan algoritma simetris, algoritma ini memiliki dua kunci yang berbeda dalam penggunaanya. Algoritma ini membangkitkan dua buah kunci yaitu kunci publik *(public key)* dan kunci privat *(private key).* Dalam penggunaanya, kunci publik digunakan pada saat proses enkripsi berlangsung. Kunci ini bersifat umum, semua pihak dapat mengetahui dan menggunakanya untuk proses enkripsi atau pembentukan *chipertext*. Namun pada saat proses dekripsi atau pengembalian bentuk data dari *chipertext* ke *plaintext* harus menggunakan kunci privat. Sifat dari kunci privat adalah rahasia, jadi tidak semua pihak dapat mengetahui atau bahkan menggunakannya untuk melakukan pembongkaran suatu *chipertext* atau proses dekripsi [RIY-08].

Berikut adalah gambar proses algoritma asimetris :



Gambar 2.2 Algoritma asimetris

2.3 RSA

2.4.1 Sejarah RSA

RSA merupakan suatu algoritma yang ditemukan oleh tiga orang peneliti dari Massachusetts Institute of Technology, mereka adalah Ron Rivest, Adi Shamir dan Len Adleman. Nama dari algoritma ini diambil dari inisial nama para penemunya. Clifford Cocks, seorang matematikawan Inggris yang bekerja untuk GCHQ, menjabarkan tentang sistem equivalen pada dokumen internal di tahun 1973. Penemuan Clifford Cocks tidak terungkap hingga tahun 1997 disebakan oleh *top-secret classification*. Algoritma tersebut dipatenkan oleh Massachusetts Institute of Technology pada tahun 1983 di Amerika Serikat sebagai U. S. Patent 4405829 [ARI-09].

Pada algoritma RSA terdapat tiga proses yaitu, pembangkitan kunci, proses enkripsi dan proses dekripsi. Letak kesulitan algoritma ini adalah bagaimana menemukan dua faktor bilangan prima yang besar yang akan digunakan sebagai kunci publik dan kunci privat. Dua bilangan prima besar tersebut p dan q dimana $p \neq q$.

2.4.2 Perumusan Algoritma RSA

Pada Algoritma RSA terdapat beberapa perumusan yang digunakan dalam proses pembentukan kunci dari dua buah bilangan prima yang besar (Munir, 2004).

1. Teori Little Fermat

   Teorema yang pertama adalah teorema *Little Fermat*. Teorema ini digunakan untuk memastikan bilangan p dan q (dua bilangan prima besar) merupakan bilangan prima. Berikut adalah perumusan teorema *Little Fermat.*

   a. Misalkan p prima dan p tidak sedemikian hingga a maka,

   $$a^{p-1} = 1(mod p) \quad (2.1)$$

   b. Bila p prima maka

   $$a^p = a(mod p) \quad (2.2)$$

   untuk sembarang bilangan bulat a.

c. Ada dua kemungkinan apabila $p|a$ maka pernyataan nomer dua otomatis berlaku namun jika p tidak sedemikian hingga a maka pernyataan yang pertama yang berlaku.

Bukti terhadap teorema *little Fermat* :

Pembuktian untuk $5^{38} \equiv 4\ (mod\ 11)$. Ambil p = 11, a = 5 dimana p tidak sedemikian hingga a maka $5^{10} \equiv 1\ (mod\ 11)$. Dengan fakta $5^2 \equiv 3(mod\ 11)$ maka diperoleh :

$$5^{38} = 5^{10\cdot 3+8} = (5^{10})^3(5^2)^4$$

$$\equiv 1 \cdot 3^4 (mod 11)$$

$$\equiv 4\ (mod\ 11)$$

2. Teori Phi Euler

Selain menggunakan teorema *little Fermat*, dalam algoritma RSA juga menggunakan fungsi phi Euler ϕ(n). Fungsi ini digunakan untuk menyatakan banyaknya bilangan bulat positif yang lebih kecil atau sama dengan *n*, dan relatif prima terhadap *n*. Bila *n* merupakan bilangan prima maka,

$$\phi(n) = n - 1 \qquad (2.3)$$

Fungsi □ merupakan fungsi multiplikatif. Teorema ini menunjukan bahwa $\phi(mn)=\phi(m)\phi(n)$ untuk semua bilangan – bilangan bulat $m \geq 1$ dan $n \geq 1$.

Contoh :

Sebagai contoh, ambil $m = 5, n = 6$,dan $\phi(mn) = \phi(30) = 8$. Dari seluruh bilangan bulat yang tidak lebih dari 30 hanya terdapat 8 bilangan yang merupakan relatif prima terhadap 30, yaitu 1, 7, 11, 13, 17, 19, 23, 29. Sedangkan 30 = 5 · 6. Maka didapatkan pula $\phi(5) = 4$ yaitu 1, 2, 3, 4 dan $\phi(6) = 2$ yaitu 1 dan 5, sehingga

$$\phi(30) = \phi(5 \cdot 6) = \phi(5)\phi(6) = 4 \cdot 2 = 8$$

3. Teori Euclid

Algoritma Euclide adalah algoritma untuk mencari pembagi persekutuan terbesar dari dua bilangan bulat. Algoritma Euclide dirumuskan sebagai berikut , misalkan akan dicari pembagi persekutuan terbesar (ppt) dari

bilangan bulat a dan b. Karena ppt ($|a|$, $|b|$) = ppt (a,b) dan misalkan a ≥ b > 0. Langkah pertama menerapkan algoritma pembagian terhadap a dan b diperoleh :

$$a = q_1 b + r_1 \quad \text{dengan} \quad 0 \le r_1 < b \qquad (2.4)$$

Jika terjadi $r_1 = 0$ maka $b|a$ dan ppt (a,b) = b. Jika $r_1 \neq 0$, bagilah b oleh r_1 dan diperoleh q_2 dan r_2 yang memenuhi :

$$b = q_2 r_1 + r_2 \quad \text{dengan} \quad 0 \le r_2 < r_1 \qquad (2.5)$$

Jika $r_2 = 0$, maka algoritma berhenti ,sebaliknya jika $r_2 \neq 0$ dengan cara yang sama diperoleh :

$$r_1 = q_3 r_2 + r_3 \text{ dengan } 0 \le r_3 < r_2 \qquad (2.6)$$

Proses pembagian ini dilanjutkan sampai sisa pembagian mencapai nilai nol. Misalkan pada langkah ke (n+1) yang mana r_{n-1} dibagi r_n degan $b > r_1 > r_2 > \ldots \ge 0$. Sehingga jika diringkas, proses di atas dapat menghasilkan sistem persamaan sebagai berikut :

$$\begin{aligned}
a &= q_1 b + r_1 && 0 \le r_1 < b \\
b &= q_2 r_1 + r_2 && 0 \le r_2 < r_1 \\
r_1 &= q_3 r_2 + r_3 && 0 \le r_3 < r_2 \\
&\vdots \\
r_{n-2} &= q_n r_{n-1} + r_3 && 0 \le r_n < r_{n-1} \\
r_{n-1} &= q_{n+1} r_n + 0
\end{aligned}$$

Sisa pembagian terakhir yang bukan nol r_n = ppt (a,b)

Contoh Penggunaan Algoritma Euclide

Akan dihitung PPT (80,12).

Jawab: 80 = 6.12 + 8

12 = 1.8 + 4

8 = 2.4 + 0

Sisa pembagian terakhir sebelum 0 adalah 4 , sehingga PPT (80,12) = 4

4. Metode *Fast Exponentiation*

Metode ini digunakan untuk menghitung operasi pemangkatan besar bilangan bulat modulo dengan cepat. Metode *fast exponentiation*

memanfaatkan ekspansi biner dari eksponennya [BUC-02].

Diberikan sembarang grup $G, g \in G$ dan bilangan bulat positif z. Untuk menghitung g^z dengan metode *fast exponentiation* , yaitu

- Hitung nilai g^{2^i} , $0 \leq i<k$
- Nilai g^z merupakan hasil dari perkalian nilai-nilai g^{2^i} , dengan a_1 = 1. Diperoleh bahwa

$$g^{2^{i+1}} = \left(g^{2^i}\right)^2 \qquad (2.7)$$

Contoh Penggunaan Metode Fast Exponentiation

Akan dihitung 6^{97} mod100 .

Jawab : $97 = 1.2^6 + 1.2^5 + 1.2^0$ atau $97 = (1100001)_2$ (semua perhitungan dilakukan dalam mod 100)

$6^{2^0} = 6$; $6^{2^1} = 36$; $6^{2^2} = 36^2 = 96\ (mod\ 100)$; $6^{2^3} = 16\ (mod\ 100)$;

$6^{2^4} = 16^2 = 56\ (mod\ 100)$; $6^{2^5} = 56^2 = 36\ (mod\ 100)$; $6^{2^6} = 96(mod\ 100)$

Sehingga $6^{97} = 6.6^{2^5}.6^{2^6}\ (mod\ 100)$ = 6. 36.96 (mod 100) = 36

$6^{97} mod\ 100 = 36$

2.4.3 Pembangkitan Kunci

Di bawah ini adalah langkah-langkah pembentukan kunci dalam algoritma RSA [MUN-04].

1. Pilih dua bilangan prima yang besar dimana disimbolkan dengan *p* dan *q*. Dan *p* dan *q* tidak boleh sama.
2. Lakukan operasi perkalian pada dua bilangan prima tersebut. $n = pq$. Simbol *n* ini sebagai modulus yang akan digunakan pada saat pembentukan kunci publik dan kunci privat.
3. Hitung :

$$\phi n=(p-1)(q-1) \qquad (2.8)$$

4. Pilih kunci publik yang relatif prima terhadap $\phi(n)$.

5. Bangkitkan kunci privat dengan menggunakan rumus

$$e \cdot d = k\,\phi(n)+1 \qquad (2.9)$$

Setelah selesai proses pembangkitan kunci privat dan kunci publik, selanjutnya kunci tersebut akan digunakan dalam perhitungan proses enkripsi dan dekripsi. Pada saat proses enkripsi, kunci yang digunakan adalah kunci publik. Kunci ini boleh diketahui oleh semua pihak, namun pada saat proses dekripsi, kunci yang digunakan merupakan kunci yang rahasia atau disebut juga private key. Sehingga dua macam kunci ini memiliki tujuan penggunaan yang berbeda dan mempunyai sifat yang berbeda pula. Berikut adalah sifat-sifat dari besaran-besaran yang ada di dalam RSA [MUN-04].

1. *p* dan *q* yang meruapakan bilangan prima (rahasia)
2. $n = p \cdot q$ (publik)
3. $\phi n=(p-1)(q-1)$ (rahasia)
4. *e* (kunci enkripsi) (publik)
5. *d* (kunci dekripsi) (rahasia)
6. *m* (*plaintext*) (rahasia)
7. *c* (*chipertext*) (publik)

2.4.4 Algoritma Sieve Of Eratosthenes

Algoritma *Sieve Of Eratosthenes* merupakan suatu algoritma yang digunakan untuk melakukan pengumpulan atau pencarian bilangan prima. Menurut Al-Ghazali [ALG-10], algoritma ini ditemukan oleh seorang penjaga perpustakaan dari perpustakaan terkenal Alexandria pada 276-194 S.M. yang bernama Eratosthenes. Dia juga seorang sarjana yang hebat. Eratosthenes dikenang dengan pengukurannya terhadai keliling bumi, memperkirakan jarak antara bumi dengan matahari dan bulan. Di dalam dunia matematika, Eratosthenes juga dikenal sebagai penemu algoritma yang digunakan untuk mencari suatu bilangan yang bernilai prima. Algoritma tersebut juga sangat klasik, yaitu menemukan seluruh bilangan prima sampai dengan batasan bilangan *N* [HAR-09].

Nilai *N* merupakan nilai yang sudah ditentukan terlebih dahulu. Proses pencariannya adalah dengan menuliskan semua bilangan integer dari angka 2 sampai dengan angka *N* yang sudah ditentukan sebelumnya. Dari setiap bilangan

tersebut, ambil bilangan pertama yaitu bilangan 2 lalu hapus setiap bilangan yang kelipatan dua. Langkah selanjutnya adalah ambil bilangan setelah bilangan pertama tadi, yaitu bilangan 3. Sama dengan langkah sebelumnya, hapus semua bilangan dari 3 sampai N yang mempunyai kelipatan 3. Langkah ini dilakukan berulang-ulang sampai semua bilangan yang ditulis diawal tadi habis atau hingga bilangan yang digunakan untuk mencari kelipatan tersebut habis dan akan tersisa bilangan prima. Proses ini secara otomatis akan mencari bilangan prima dari kumpulan bilangan integer. Dalam penerapan algoritma ini terdapat cara yang mangkus untuk mengurangi waktu proses atau kompleksitasnya, yaitu dengan cara membatasi bilangan yang akan dilipatkan. Jadi bilangan yang akan dilipatkan tadi dibatasi sampai $\sqrt{N}$ (pembulatan).

Contoh :

Menentukan bilangan prima antara 2 hingga 100, dimana N = 100. Langkah-langkahnya adalah sebagai berikut [ALG-10] :

1. Tampilkan semua bilangan integer antara 2 sampai 100 dan masukan dalam array.
2. Selanjutnya coret bilangan di dalam array yang kelipatan 2 sampai bilangan 100.
3. Selanjutnya ambil bilangan yang tersisa dalam array setelah bilangan 2. Dalam contoh ini adalah bilangan 3. Coret bilangan di dalam array yang kelipatan tiga.
4. Selanjutnya ambil bilangan setelah bilangan 3 yang masih tersisa di dalam array.
5. Coret semua bilangan yang kelipatan 5 yang ada di dalam array. Dan ambil bilangan yang tersisa setelah bilangan 5.
6. Selanjutnya coret semua bilangan yang kelipatan 7 yang ada di dalam array. Selanjutnya ambil bilangan yang tersisa di dalam array setelah bilangan 7.
7. Bilangan tersebut adalah 11, namun karena $11 \geq \sqrt{100}$ maka perulangan dihentikan dan bilangan yang masih tersisa di dalam array merupakan bilangan prima.

Jadi kesimpulanya adalah bilangan yang 2, 3, 5, 7, 9 dan bilangan yang tersisa di dalam array merupakan bilangan prima. Ini dapat menyederhanakan proses perulangan yang dapat menjadi kelemahan algoritma ini dalam hal kompleksitas.

2.4.5 Enkripsi RSA

Enkripsi RSA merupakan suatu proses dimana terjadi perubahan atau transformasi sebuah *plaintext.* Asimetris enkripsi disebutjuga dengan sebutan *public key cryptography*. Ini dikarenakan hanya pada enkripsi asimetris inilah penggunaan dua kunci yaitu kunci publik dan kunci privat diterapkan dalam proses enkripsi dan dekripsi. Sesuai namanya, kunci publik merupakan suatu kunci yang bersifat terbuka atau tidak rahasia dan kunci privat merupakan suatu kunci yang bersifat rahasia dan hanya orang yang mempunyai hak saja yang dapat mengetahui kunci privat tersebut [STA-07].

Dalam proses ini, RSA menggunakan kunci publik untuk perhitungan enkripsinya. Diilustrasikan A akan mengirimkan *m* (pesan) kepada B. Pihak A harus membuat chipertext *c* dengan menggunakan rumus,

$$c = m^e \bmod n \tag{2.10}$$

Di dalam rumus 2.8 *e* dan *n* adalah kunci publik dari B. Lalu A mengirimkan pesan ke B. Untuk mendekripsikan, B juga harus melakukan eksponensiasi yaitu dengan rumus,

$$m = c^d \bmod n \tag{2.11}$$

Hubungan antara variabel *e* dan *d* adalah modal pihak B untuk membaca data *chipertext*. Selama hanya pihak B yang mengetahui nilai privatnya maka hanya pihak B saja yang dapat melakukan proses dekripsi.

2.4.6 Dekripsi RSA

Merupakan suatu proses dimana sebuah *chipertext* yang terbentuk dikembalikan kembali ke dalam bentuk *plaintext* menggunakan sebuah kunci privat. Kunci privat ini hanya boleh dimiliki oleh orang yang berhak membuka enkripsi tersebut. Dalam prosesnya, *c (chipertext)* didekripsi dengan cara hitung kunci privat penerima untuk dekripsi *d* ke dalam rumus

$$m_i = C_i{}^d mod n. \tag{2.12}$$

Relasi antara *e* dan *d* memastikan dalam proses dekripsi suatu *chipertext* [LAB-00]. Dalam proses dekripsi ini, algoritma RSA menggunakan kunci private yang terbentuk pada proses enkripsi dan dikirimkan kepada pihak yang akan melakukan proses dekripsi.

2.4.7 Brute Force *Attack* pada RSA

Algoritma *brute force* adalah algoritma yang memecahkan masalah dengan sangat sederhana, langsung, dan dengan cara yang jelas/lempang. Penyelesaian permasalahan password cracking dengan menggunakan algoritma *brute force* akan menempatkan dan mencari semua kemungkinan password dengan masukan karakter dan panjang password tertentu tentunya dengan banyak sekali kombinasi password [PRA-10].

Brute force attack pada metode RSA merupakan suatu serangan dengan melakukan percobaan kombinasi pada kunci privat RSA. Pihak yang melakukan penyerangan dengan menggunakan metode brute force akan mencoba mengkombinasikan setiap angka untuk menebak kunci publik dan kunci privat. Suatu kunci yang pendek akan mudah dipecahkan menggunakan metode brute force ini. Oleh karena itu, disarankan bagi pengguna metode enkripsi RSA untuk menggunakan bilangan prima besar seagai kunci publik dan kunci privat. Tingkat keamanan suatu kunci pubik dan privat tergantung pada panjang kunci yang digunakan, semakin panjang kunci yang digunakan maka semakin susah juga bagi para *cracker* atau *hacker* untuk memecahkan kunci yang digunakan dalam enkripsi tersebut. Setiap sistem penyerangan yang menggunakan sistem *brute force* pada umumnya berhasil melakukan pemecahan kunci enkripsinya, namun yang membedakan serangan terhadap suatu kunci enkripsi dengan kunci enkripsi yang lain adalah waktu yang dibutuhkan untuk melakukan pemecahan kunci enkripsinya.

Langkah dari Brute Force Attack adalah 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

$62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 \times 62 = 628$ buah.

Secara teori, serangan secara exhaustive ini dipastikan berhasil mengungkap plainteks tetapi dalam waktu yang sangat lama [SUL-12]. *Brute Force Attack* melakukan pemfaktorkan kunci publik pertama menjadi bilangan prima p dan q dengan persamaan 2.13 :

$$q = n / p \qquad (2.13)$$

dimana n yang diuji coba. Sedangkan *p* dan *q* merupakan bilangan prima,dimana nilai dari *p* akan didapatkan dari hasil iterasi menggunakan cara *brute force*. Setelah *p* dan *q* bisa difaktorkan, maka akan dengan mudah untuk mendapatkan kunci privatnya karena sifat dari kunci publik tersebut tidak rahasia. Dengan kunci publik dan rumus 2.8 yaitu $\phi n=(p-1)(q-1)$ maka kunci privat akan dapat diketahui.

2.4 *Electronic Mail*

Email pertama kali dikemabangkan oleh para peneliti yang ada di MIT. Para peneliti tersebut mengembangkan komunikasi *email* pada tahun 1966, dan pada tahun 1969 ARPANET juga memberikan kontribusi untuk melakukan pengembangan teknologi *email.* Lalu pada tahun 1971 Ray Tomlinson memperkenalkan penggunaan tanda "@" untuk memisahkan *user* dengan nama *domain.*

Email merupakan sebuah layanan dalam dunia internet yang pada saat ini semakin populer dikalangan pengguna internet. Banyak perusahaan yang memanfaatkan ini untuk komunikasi antar *user* atau pegawainya. *Email* meruapakan suatu pesan yang dikirimkan melalui koneksi jaringan internet. Sesuai artinya, *email* merupakan suatu surat elektronik yang dapat dikirmkan kepada seseorang yang juga memiliki alamat *email.* Pada saat ini semua orang dapat secara cuma-cuma mendapatkan alamat *email* tersebut. Banyak *search engine* yang akhir-akhir ini melengkapi fitur mereka dengan menambahkan

layanan email secara gratis. Salah satu contohnya adalah *Google Mail*. Salah satu layanan dari *gmail* atau *google mail* ini adalah layanan email. Sebagai contoh, di dalam fiturnya, google menyediakan space kepada pengguna untuk dapat dimanfaatkan sebagai *hardisk* bayangan yang dapat digunakan untuk menyimpan semua data email *user*.

2.5 Visual Basic

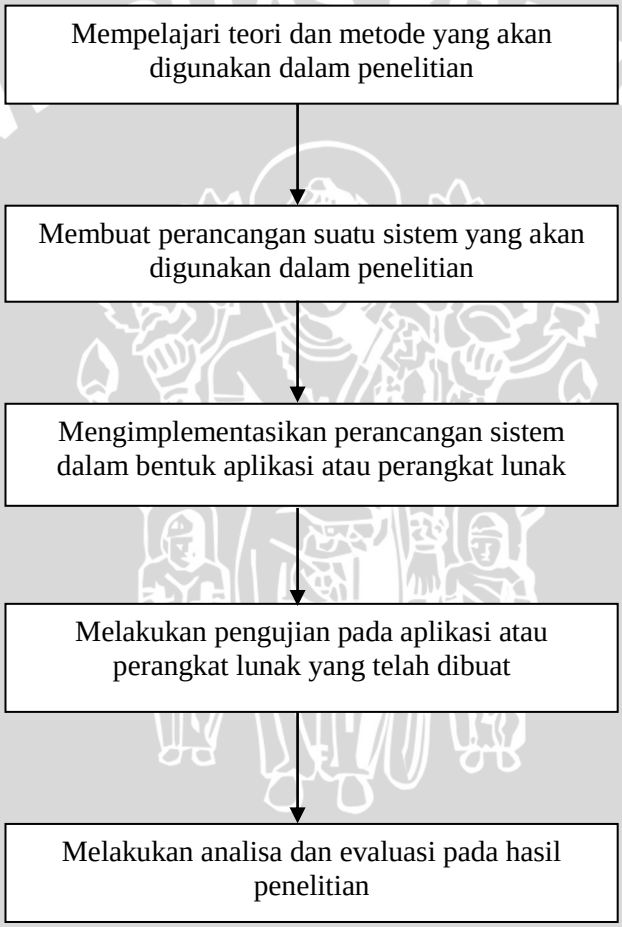
Visual Basic merupakan bahasa pemrograman yang sangat populer dikalangan programer. Bahasa ini dijadikan sebagai bahasa pemrograman utama oleh *Microsoft* selama bertahun – tahun. Basic sendiri meruapakan sebuah akronim dari kata *Beginner's All purpose Symbolic Instruction Code*. Jadi dari arti akronimnya, bahasa pemrograman Basic ini memang dirancang untuk para pemula. Selain mudah digunakan oleh para pemula, bahasa ini juga andal untuk semua tujuan. Kelebihan bahasa ini adalah dapat menggunakan berbagai fasilitas yang ada untuk mengembangkan dan mengetahui lebih jauh semua hal yang ada di dalam sistem operasi *Windows*.

2.5.1 VB.Net

VB.Net merupakan salah suatu bahasa pemrograman VB yang menggunakan framework .Net. Di dalam VB.Net juga mendukung penggunaan sebuah DLL dalam pemrogramanya. VB.Net merupakan pengembangan dari VB6, diluncurkan oleh *Microsoft* pada bulan februari 2002. Penelitian ini juga memanfaatkan DLL dari *Microsoft* yang digunakan untuk memanipulasi file dokumen yang tersedia pada sistem operasi milik *Microsoft*. Dengan DLL yang tersedia, peneliti dapat mengakses isi data dari file dokumen yang dimaksud, sehingga dapat memudahkan dalam proses enkripsi data. Jadi setiap file dokumen yang akan dikriptografikan, akan dibaca terlebih dahulu menggunakan DLL yang tersedia untuk mengambil setiap karakter yang ada. VB.Net memiliki keunggulan dalam integrasi dengan berbagai aplikasi yang dibuat oleh Windows, salah satunya adalah aplikasi *Office* yaitu *Microsoft Office*.

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab ini menjelaskan tentang langkah-langkah dan perancangan serta metodologi yang akan digunakan dalam penelitian. Langkah-langkah yang akan dikerjakan selama penelitian ditunjukan pada gambar 3.1.



Gambar 3.1 Langkah – langkah penelitian

Berdasarkan gambar 3.1, langkah-langkah yang dilakukan dalam penelitian yaitu sebagai berikut:

1. Melakukan studi literatur tentang kriptografi, algoritma RSA dan tentang *file* dokumen dari *Microsoft Office.* Ini bertujuan untuk melakukan pembacaan suatu karakter yang terdapat dalam file dokumen tersebut.
2. Melakukan perancangan sistem yang akan digunakan dalam penelitian.
3. Menerapakan rancangan sistem yang telah dibuat sesuai dengan studi literatur yang telah dipelajari maupun dari penelitian sebelumnya.
4. Melakukan pengujian terhadap aplikasi atau perangkat lunak yang telah dibuat dalam penelitian dengan cara mengujinya menggunakan beberapa metode pengujian.
5. Melakukan analisa terhadap hasil pengujian dan melakukan evaluasi terhadap perangkat lunak yang telah diuji.

3.1 Proses Kriptografi

Dalam proses kriptografi terdapat beberapa langkah dalam implementasinya. Diawali dengan tahapan mengambil *plaintext* yang akan dienkripsi selanjutnya dilakukan pembentukan kunci untuk proses enkripsi, di dalam proses ini kunci yang digunakan adalah kunci publik. Selanjutnya setelah mendapatkan kunci privat, maka dilakukan proses dekripsi chipertext menjadi sebuah *plaintext* yang dapat dibaca. Proses kriptografi ditunjukan pada gambar 3.2 .

Gambar 3.2 Proses Kriptografi

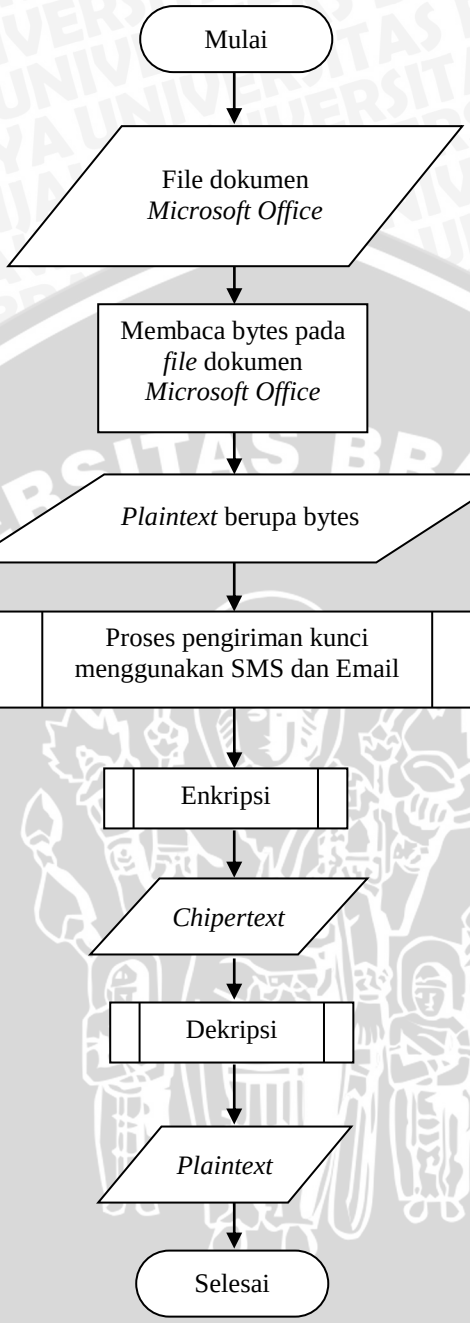
3.2 Analisa dan Perancangan Sistem

3.2.1 Deskripsi Umum Sistem

Dalam penelitian ini akan dibuat sebuah aplikasi atau perangkat lunak yang dapat melakukan kriptografi terhadap suatu *file* dokumen *Microsoft Office*. Dalam proses kriptografi terdapat tiga bagian proses yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Di dalam proses pembentukan kunci, aplikasi ini menggunakan algoritma *Sieve Of Eratosthenes* untuk pencarian bilangan prima. Setelah menemukan kumpulan bilangan prima, proses selanjutnya adalah memilih dua bilangan prima yang terbesar. Setelah memilih dua bilangan prima yang terbesar, maka proses berikutnya adalah membaca byte-byte yang ada di dalam file tersebut dan diambil sepanjang 512 byte untuk dilakukan proses enkripsi. Selanjutnya dilakukan proses enkripsi menggunakan dua buah kunci yang telah dibangkitkan yaitu kunci publik dan kunci privat dengan cara mengubah *plaintext* dari file dokumen menjadi sebuah *chipertext.* Proses yang terakhir adalah proses dekripsi. Dalam proses ini dilakukan pengubahan kembali sebuah *chipertext* menjadi sebuah *plaintext* yang dapat dibaca kembali dengan menggunakan kunci privat yang tercipta pada saat proses enkripsi sebelumnya.

3.2.2 Perancangan Sistem

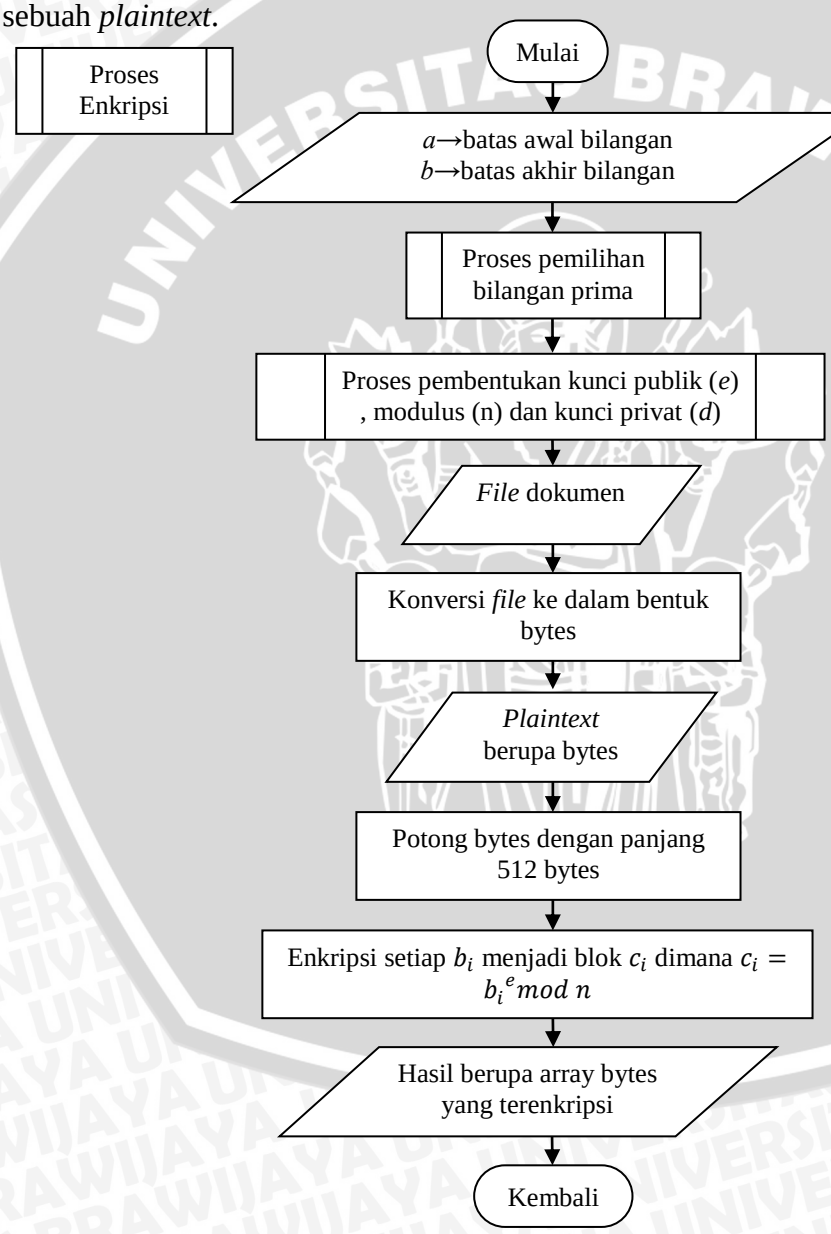
Perancangan sistem yang akan dikerjakan dalam penelitian ini merupakan penggabungan dari beberapa proses yang akan menghasilkan suatu *plaintext* dan *chipertext.* Aplikasi ini juga dapat mengubah kembali *chipertext* menjadi sebuah *plaintext.* Dalam perancangan ini peneliti menggunakan objek penelitian dokumen dari *Microsoft Office.*Alur sistem atau *flowchart* sistem secara umum di dalam peneitian ini ditunjukan pada gambar 3.3.



Gambar 3.3*Flowchart* sistem secara umum

3.2.2.1 Proses Enkripsi

Setelah mendapatkan kunci publik dan kunci privat, langkah selanjutnya adalah proses enkripsi. Penelitian ini menggunakan objek yang akan dikriptografikan atau disandikan dari *file* dokumen *Microsoft Office*. *File* dokumen yang akan dibaca adalah *file .doc* dan *.xls*. Dalam prosesnya, sistem akan membaca bytes dari *file* dokumen tersebut. Selanjutnya sistem akan mengambil 512 bytes untuk di enkripsi. Gambar 3.4 menunjukan proses enkripsi sebuah *plaintext*.



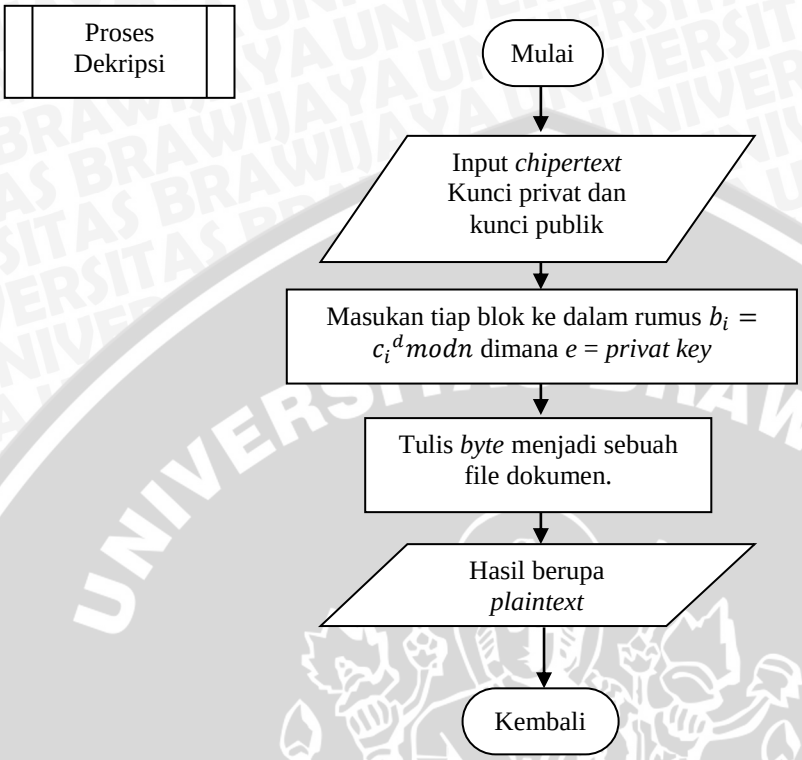
Gambar 3.4 Proses Enkripsi

Berikut adalah penjelasan dari *flowchart* proses enkripsi :

1. Langkah yang pertama adalah melakukan proses pencarian bilangan prima. Langkah pencarian bilangan prima ini ditunjukan pada gambar 3.8
2. Selanjutnya adalah membentuk kunci yang akan digunakan dalam proses enkripsi dan dekripsi. Hasil dari proses pembentukan kunci ini adalah pasangan kunci privat dan publik. Proses ini ditunjukan pada gambar 3.6
3. Langkah yang ketiga adalah mengambil data inputan dari *file* dokumen yang akan di enkripsi. *File* ini merupakan *file* yang berkstensi *.doc, .xls*.dan *.ppt.* Metode pengambilan data inputnya adalah dengan mengambil bytes-bytes yang ada di dalam *file* yang berupa array bytes.
4. Setelah mendapatkan data inputan berupa *array* bytes, maka selanjutnya inputan tersebut akan dienkripsi menggunakan rumus $c_i = b_i^{\,e} mod n$ dimana $b_i$merupakan blok-blok yang berupa index *array*. Lalu setelah mendapatkan nilai, maka bytes yang telah dienkripsi tersebut dimasukan kembali kedalam *file* yang dienkripsi dan digabungkan dengan bytes sisa yang tidak dienkripsi.

3.2.2.2 Proses Dekripsi

Setelah melakukan proses enkripsi dan mendapatkan sebuah *chipertext*, langkah selanjutnya adalah melakukan proses dekripsi atau proses merubah *chipertext* menjadi sebuah *plaintext*. Dalam proses dekripsi ini dibutuhkan sebuah *private key* yang dihasilkan dalam proses pembentukan kunci seperti yang ditunjukan pada gambar 3.5.



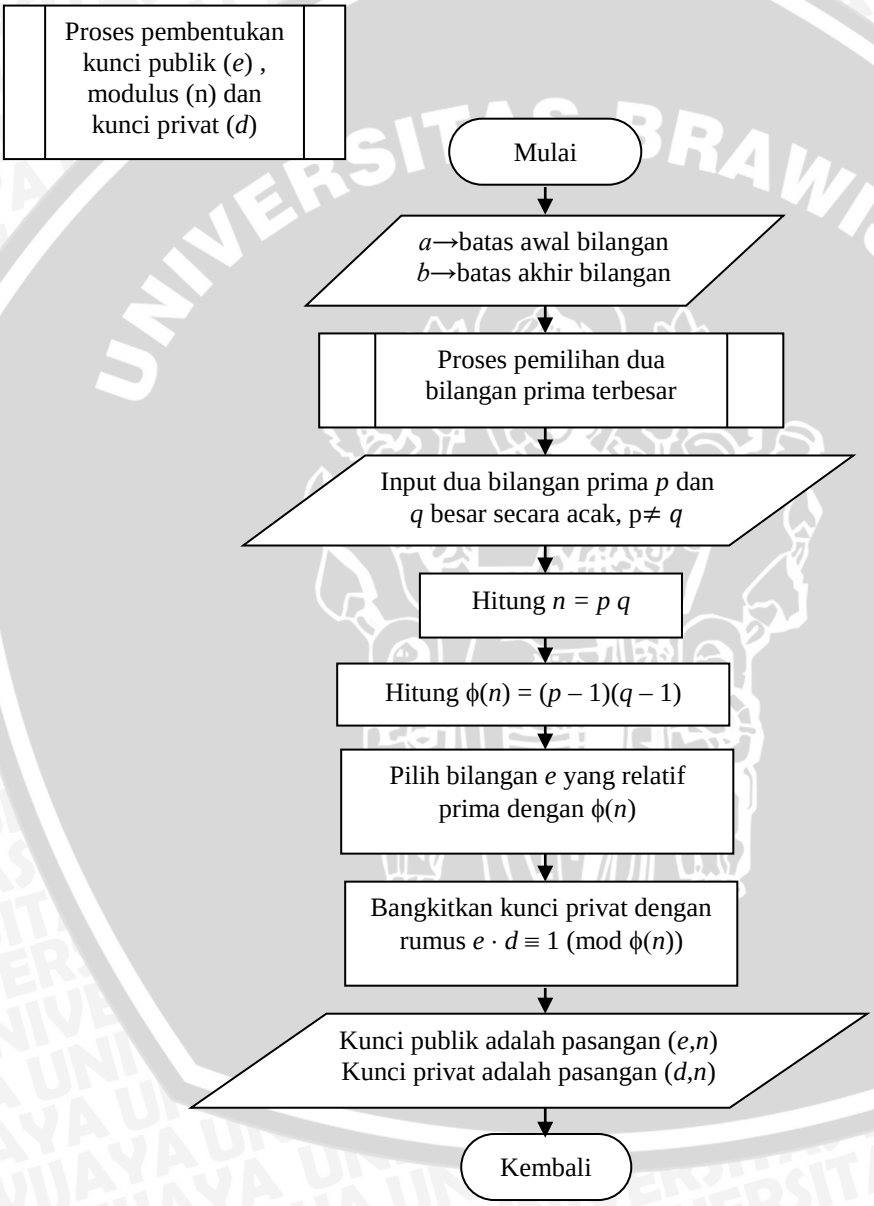
Gambar 3.5 Proses Dekripsi

Berikut adalah penjelasan dari *flowchart* proses dekripsi :

1. Terdapat inputan berupa *plaintext*, kunci publik dan kunci privat.
2. Hitung tiap-tiap blok ke dalam rumus $b_i = c_i{}^d modn$ dimana *e* merupakan kunci privat dan blok-blok tersebut berupa *array* yang memiliki nilai bytes.
3. Setelah memasukan tiap-tiap blok bilangan tersebut kedalam rumus dekripsi, maka akan dihasilkan sebuah bilangan desimal yang nantinya akan menjadi *plaintext.*
4. Setelah mendapatkan hasil dari proses perhitungan dekripsi, maka bytes yang telah di dekripsi dimasukan ke dalam kumpulan bytes *file* dan digabungkan dengan byte yang tidak dienkripsi.

3.2.2.3 Pembentukan Kunci

Dalam proses kriptografi, tahap pertama yang harus dilakukan adalah membentuk dua buah kunci untuk proses enkripsi dan dekripsi. Kunci publik merupakan kunci yang digunakan untuk melakukan proses enkripsi, sedangakan kunci privat merupakan kunci yang digunakan untuk melakukan proses dekripsi. Proses pembentukan kunci publik dan privat akan ditunjukan pada gambar 3.6.



Gambar 3.6 Proses pembentukan kunci

Berikut adalah penjelasan mengenai proses pembentukan kunci :

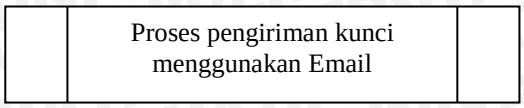
1. Menjalankan proses pencarian bilangan prima.

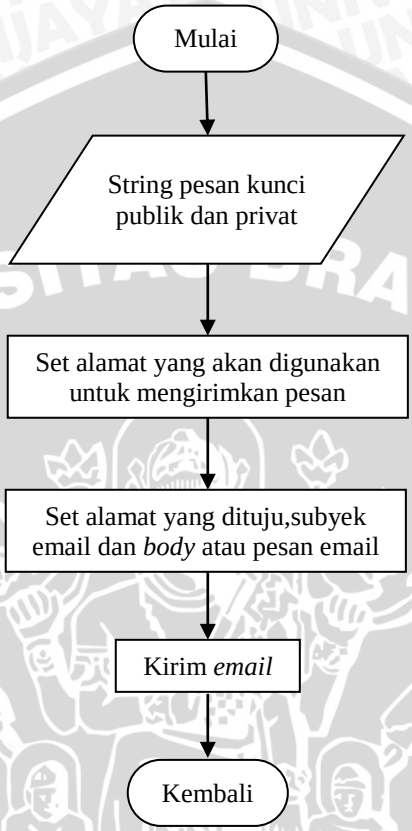
2. Setelah mendapatkan kumpulan bilangan prima, maka sistem akan memilih secara acak dua bilangan terbesar prima *(p dan q)* yang dapat digunakan untuk membangkitkan kunci publik dan privat. Syarat pemilihan dua bilangan besar prima tersebut adalah $p \neq q$.
3. Hitung $n = p\ q$
4. Hitung $\phi(n) = (p - 1)(q - 1)$
5. Selanjutnya pilih bilangan e yang relatif prima dengan $\phi(\nu)$
6. Bangkitkan kunci privat dengan rumus $e \cdot d = k\ \phi(n)+1$
7. Mendapatkan pasangan kunci publik (e,n) dan pasangan kunci privat (d,n)

Dari pembentukan kunci seperti pada gambar 3.6, kunci – kunci tersebut akan digunakan dalam proses selanjutnya, yaitu proses enkripsi sebuah *plaintext* dan proses dekripsi sebuah *chipertext* menjadi sebuah *plaintext*.

3.2.2.4 Proses pengiriman kunci menggunakan Email

Proses ini digunakan untuk mengirimkan pesan yang berisi kunci privat dan kunci publik melalui *email*. Dalam proses ini peneliti menggunakan salah satu *library* yang ada di dalam *Visual Studio* untuk mengirimkan *email* menggunakan *account* dari *gmail.com*. Gambar 3.8 menjelaskan alur dari pengiriman pesan kunci menggunakan *email*.

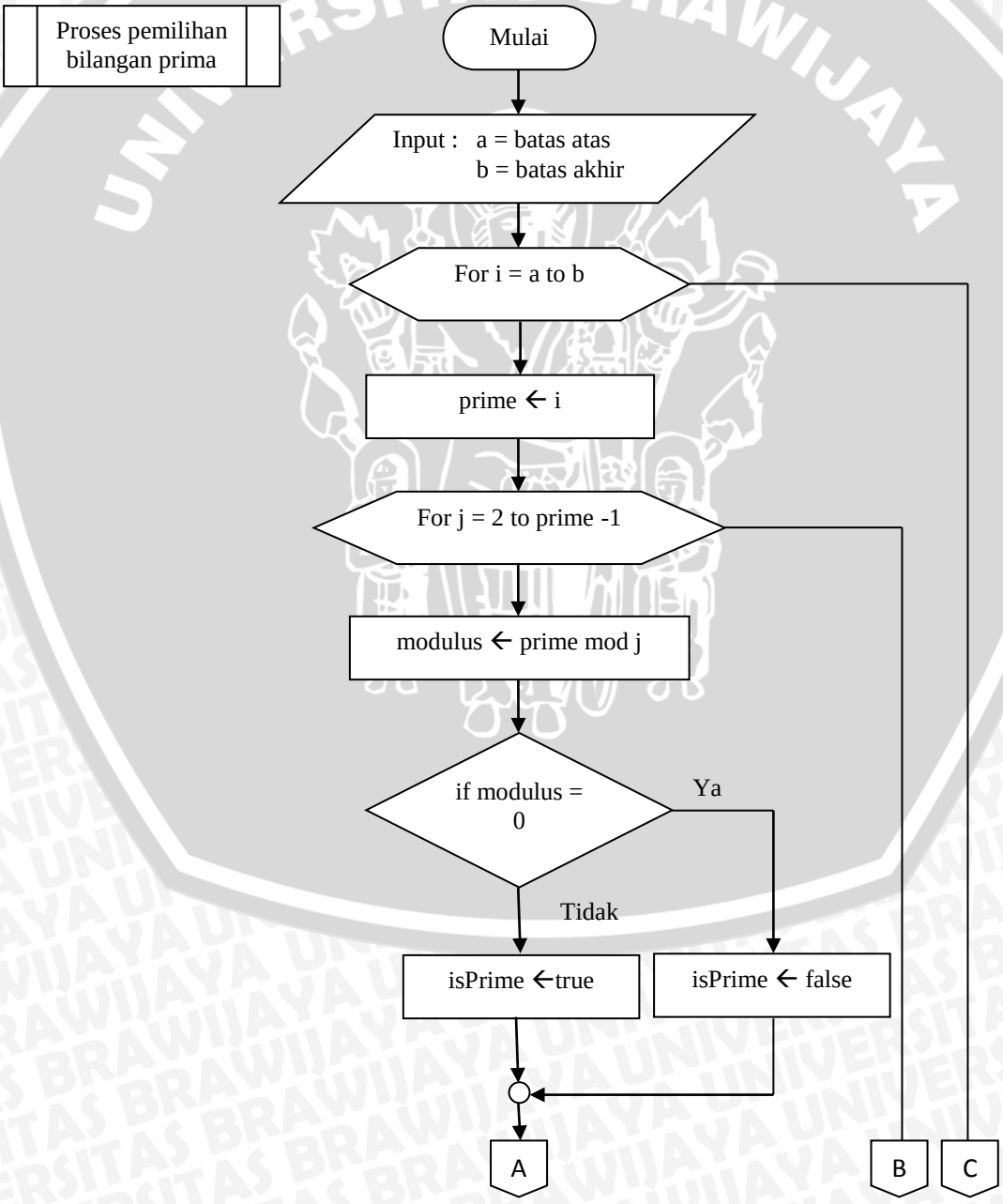


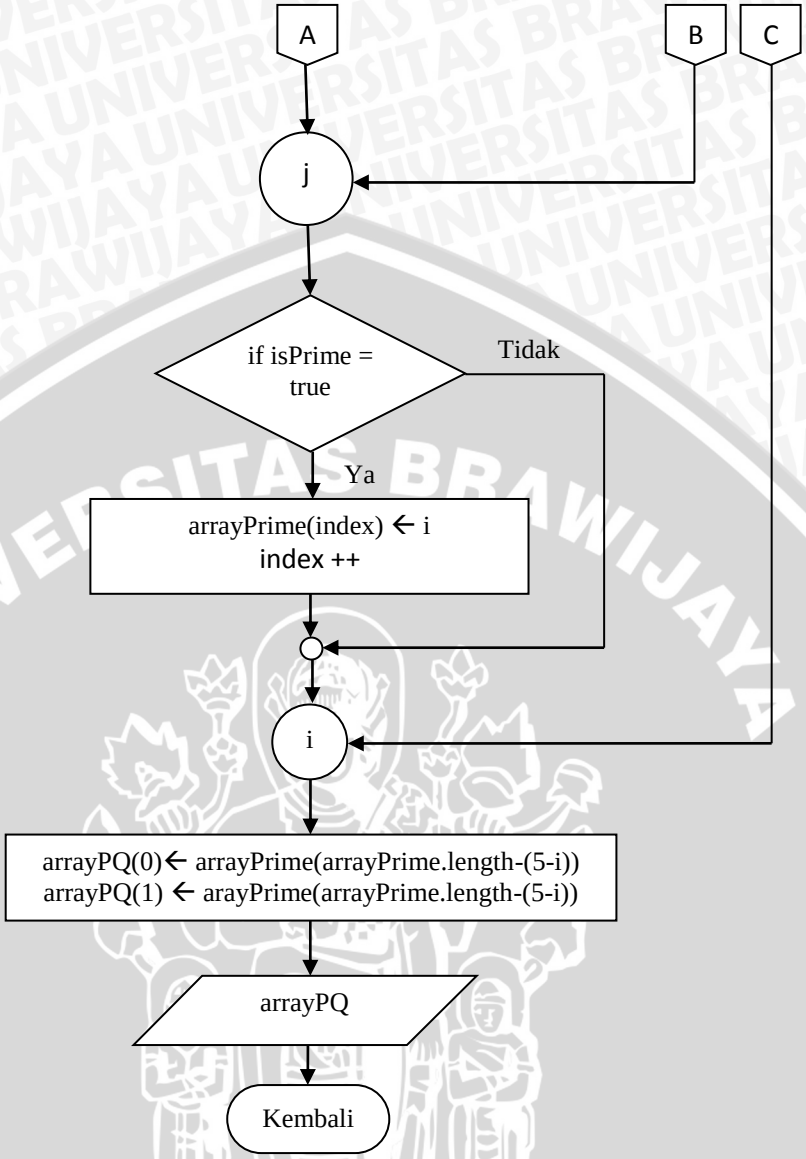


Gambar 3.7 Pengiriman pesan key menggunakan email

3.2.2.5 Proses pemilihan bilangan prima

Algoritma yang ditemukan oleh seorang matematikawan bernama Eratosthenes ini, sesuai dengan namanya, melakukan "penyaringan" terhadap suatu kumpulan bilangan menjadi kumpulan bilangan prima dengan meng-eliminasi bilangan yang bukan bilangan prima. Lebih jelasnya, metode *Sieve of Eratosthenes* digambarkan pada langkah-langkah pada gambar 3.8 tersebut terdapat suatu proses dimana proses tersebut digunakan untuk memilih dua bilangan prima yang besar [ALG-10].





Gambar 3.8 Proses pemilihan bilangan prima

Berikut adalah penjelasan dari pencarian bilangan prima :

1. Langkah pertama adalah memasukan batasan bilangan *k* yang akan digunakan sebagai batasan dalam proses pencarian bilangan prima.
2. Setelah mendapatkan nilai batas awal dan batas akhir nilai bilangan bulat, maka langkah berikutnya adalah melakukan looping antara batas awal hingga batas akhir. Bilangan tersebut akan dicek satu persatu apakah prima atau bukan dengan cara membagi bilangan tersebut dengan bilangan 2 hingga bilangan sebelum bilangan itu sendiri. Apabila nilai pembagianya tidak menghasilkan sisa atau modulus tidak sama dengan 0, maka bilangan

tersebut merupakan bilangan prima dan selanjutnya akan disimpan ke dalam arrayPrime.

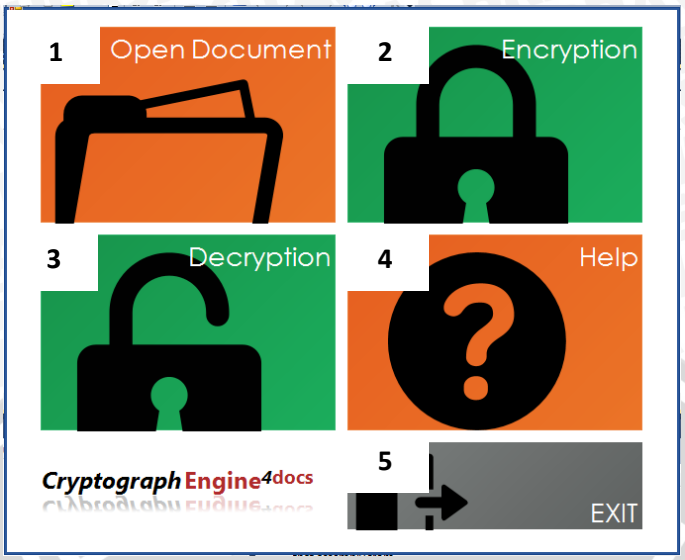
3. Selanjutnya adalah melakukan proses pemilihan bilangan prima yang terbesar. Prosesnya adalah dengan memilih bilangan prima yang berada di dalam index arrayPrime yang terakhir, namun dikurangi lima index dari total panjang array dan begitu juga dengan bilangan prima yang kedua dipilih index yang dikurangi lima index dari nilai index bilangan prima pertama. Misal panjang arrayPrime 100, maka index yang dipilih adalah 95 untuk bilangan prima pertama, dan index 90 untuk nilai bilangan prima yang kedua. Cara ini untuk mempersulit pihak yang ingin mencari atau menebak bilangan prima p dan q yang digunakan untuk mendapatkan kunci privat dan kunci publik.

Dengan menggunakan pembatasan nilai kelipatan bilangan $X_k \geq \sqrt{k}$ maka dapat menyederhanakan proses pencarian bilangan prima pada algoritma ini. Dengan begitu kompleksitas dari algoritma ini juga dapat lebih mangkus dan cepat dari segi eksekusi program.

3.3 Perancangan Antar Muka

Perancangan antar muka merupakan sebuah desain dari aplikasi atau perangkat lunak yang akan digunakan dalam penelitian ini. Dalam penelitian ini terdapat beberapa halaman yang akan dijelaskan seperti dibawah ini.

A. Halaman pertama ini menampilkan tampilan awal program yang berisi menu *File,* enkripsi, dekripsi dan help. Pada tampilan awal ini pula terdapat *icon* dari aplikasi kriptografi ini dan terdapat 3 buah control *window,* yaitu *close, minimize* dan *maximize*. Tampilan awal dalam aplikasi ini masih kosong dan hanya ada sebuah gambar berupa gembok yang melambangkan kemanan dari sistem kriptografi. Gambar 3.9 adalah desain tampilan awal program kriptografi menggunakan RSA.



Gambar 3.9 Tampilan awal program

Keterangan :

1. Menu yang ditunjukan pada nomor satu merupakan menu yang digunakan untuk membuka file dokumen yang akan dienkripsi. Tujuan dar menu ini adalah supaya pengguna yang ingin melakukan proses editing sebelum melakukan proses enkripsi *file* dapat dilakukan menggunakan pilihan menu nomor satu. Setelah melakukan proses editing pengguna juga dapat langsung melakukan proses enkripsi *file*.
2. Pada pilihan ini merupakan pilihan menu enkripsi. Menu ini digunakan untuk melakukan proses enkripsi *file*. Berbeda dengan menu pada nomor satu, pada menu ini pengguna tidak dapat melihat isi *file* apalagi melakukan proses editing. Dengan menggunakan menu ini pengguna diharuskan memilih langsung *file* yang akan dienkripsi.
3. Pilihan ini merupakan pilihan menu untuk melakukan proses dekripsi *file*. Pengguna yang ingin melakukan proses dekripsi dapat menggunakan menu ini dengan memasukan *file* yang akan didekripsi dan memasukan kunci privat dan publik.

4. Menu selanjutnya adalah menu *help* yang ditunjukan pada nomor 4. Menu ini merupakan menu pendukung yang ditujukan kepada pengguna yang akan menggunakan aplikasi ini. Menu ini bertujuan untuk memandu atau menuntun pengguna untuk melakukan proses enkripsi dan dekripsi file.

B. Halaman selanjutnya adalah halam yang digunakan untuk melakukan pemilihan *file* yang akan dibuka dan dilakukan proses editing. Di dalam halaman ini terdapat 4 menu pilihan. Pada saat melakukan pemilihan *file* yang akan dibuka maka akan muncul sebuah open dialog yang akan membantu pengguna untuk memilih *file* mana yang akan dibuka.Halaman ini ditunjukan pada gambar 3.10.

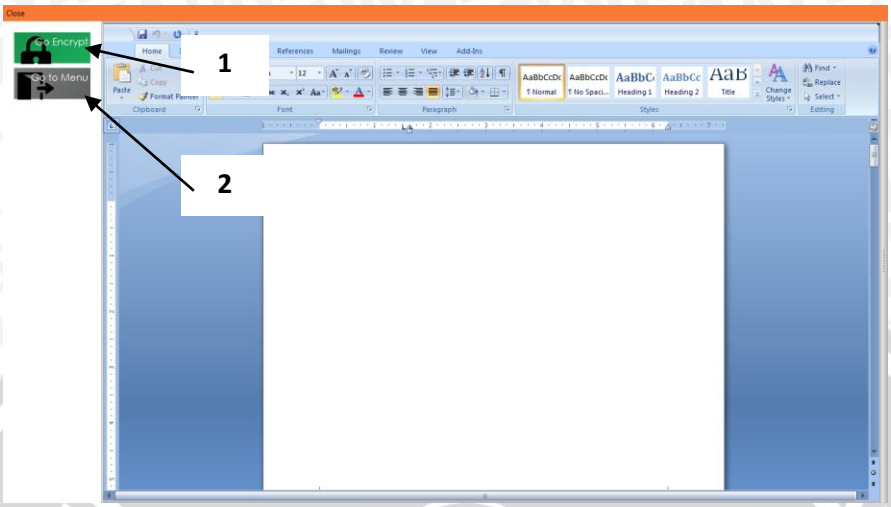
Gambar 3.10 Tampilan halaman *format file*

Keterangan :

1. Pilihan nomor satu digunakan untuk membuka *fileMicrosoft Office Word*. Sehingga ekstensi yang dapat dipilih dalam pilihan ini adalah .doc atau .docx
2. Pilihan selanjutnya adalah pilihan yang digunakan untuk membuka *file Microsoft Office Excel*. Pada pilihan ini pengguna hanya dapat memilih *file* yang berekstensi .xls atau .xlsx
3. Pilihan nomor 3 adalah pilihan yang digunakan untuk membuka *file Microsoft Office Power Point*. Disini pengguna juga hanya dapat membuka *file* yang berekstensi .ppt dan .pptx
4. Pilihan yang terakhir adalah tombol exit yang digunakan untuk kembali ke menu utama.

C. Halaman selanjutnya merupakan *window* yang digunakan untuk membantu pengguna untuk membuka *file* yang yang akan dilakukan proses editing terlebih dahulu. Sebelumnya *file* yang akan dibuka dipilih seperti pada gambar 3.10. Pada bagian ini pengguna bebas melakukan

proses editing pada dokumen sebelum dilakukan proses enkripsi terhadap *file* tersebut. Gambar 3.11 menunjukan desain halaman *open filechipertext.*

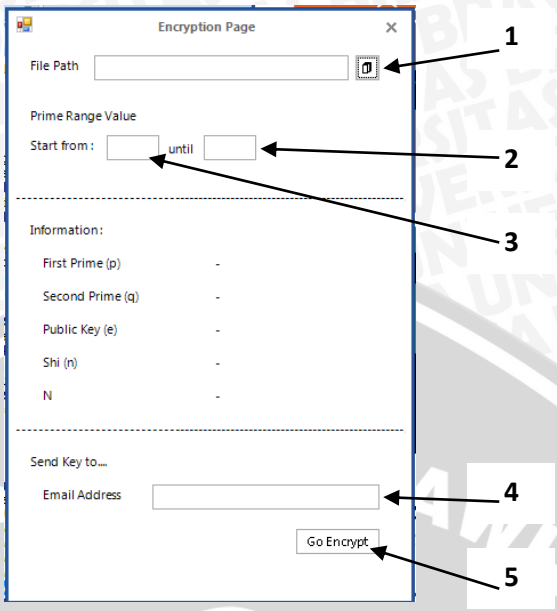


Gambar 3.11 Tampilan halaman *open file*

Keterangan:

1. Pada pilihan ini terdapat tombol enkripsi. Tombol ini berfungsi untuk melakukan proses enkripsi pada*file* yang sedang kita buka.
2. Pilihan pilihan ini digunakan untuk kembali ke dalam menu utama seperti pada gambar 3.9

D. Pada halaman ini terdapat sebuah rancangan antarmuka yang digunakan untuk melakukan proses enkripsi *file*. Berbeda pada antarmuka *open file* yang ditunjukan pada gambar 3.11, halaman ini tidak memungkinkan pengguna untuk melakukan proses editing terlebih dahulu. Jadi pada halaman ini pengguna hanya dapat memilih dan menentukan range bilangan primanya saja. Gambar 3.12 berikut ini adalah desain halaman *create key.*

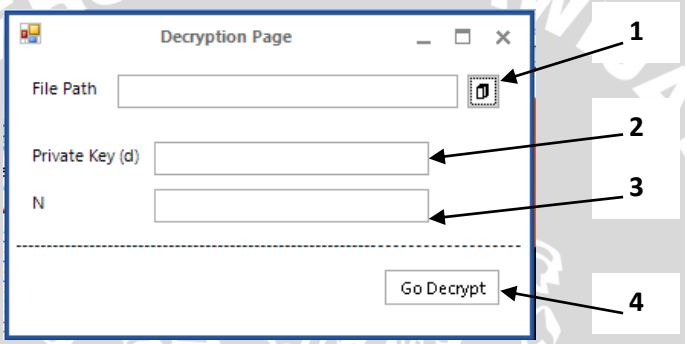


Gambar 3.12 Halaman *encryption*

Keterangan :

1. Pada tombol ini digunakan untuk melakukan pemilihan *file* yang akan dienkripsi. Setelah tombol ini di *click* selanjutnya akan muncul sebuah *open dialog* yang akan mengarahkan pengguna untuk memilih sebuah *file* yang akan dienkripsi.
2. Pada pilihan ini merupakan sebuah *text box* yang digunakan untuk memasukan batas akhir bilangan yang akan digunakan untuk membangkitkan suatu bilangan prima.
3. Sama seperti pilihan nomor 2 namun pada *text box* ini bilangan yang dimasukan adalah bilangan batas awal yang digunakan untuk pencarian bilangan prima
4. Pada bagian ini digunakan untuk menampung *email address* yang dimasukan pengguna.
5. Pada pilihan ini merupakan tombol yang digunakan untuk memulai pembangkitan kunci dan proses enkripsi terhadap *file* yang telah dipilih dan dengan menggunakan *range* bilangan prima yang telah dimasukan dan kunci privat dan publik akan dikirim ke alamat *email* yang telah dimasukan.

E. Pada halaman selanjutnya adalah halaman dimana *user* akan melakukan proses dekripsi *file* dokumen *Microsoft Office.* Akses ke dalam halaman ini terdapat pada *file menu*yang berjudul *decryption.* Langkah pertama adalah memproses*key* yang akan digunakan dalam proses dekripsi *file.* Oleh karena itu *window* pertama yang akan terbuka pada saat mengakses pilihan dekripsi adalah *window open file* yang digunakan untuk melakukan navigasi ke *filekey* yang terbentuk pada saat proses enkripsi tadi. Gambar 3.13 adalah desain dari *open key* yang digunakan untuk memudahkan pengguna membuka dan mencari *file key* yang akan digunakan.



Gambar 3.13 Halaman *Decryption*

Keterangan :

1. Pada pilihan ini terdapat tombol *open file* dimana tombol ini berfungsi untuk melakukan pemilihan *file* yang akan di dekripsi.
2. Merupakan suatu *text box* yang digunakan untuk memasukan kunci privat yang akan digunakan dalam proses dekripsi.
3. Sama seperti nomor 2, *text box* ini digunakan untuk memasukan nilai N yang merupakan pasangan kunci privat *d.*
4. Bagian ini merupakan tombol yang digunakan untuk memulai proses dekripsi apabila semua masukan sudah diisi.

3.4 Perhitungan Manual

a. Pilih bilangan prima *p* dan *q*

p=31

q=37

n=p.q

n=31 x 37

n = 1147

b. Hitung ϕ(n)

ϕ(n)=(*p*-1)(*q*-1)

=(31-1)(37-1)

=1080

c. Pilih *e* yang relative prima terhadap ϕ(n) yaitu 29 dimana PBB(1080,29) adalah 1

d. Hitung $d = \frac{1+(k\ .1080)}{29}$

$= \frac{1+(4\ .1080)}{29}$

$= 149$

Dimana *k* adalah nilai 1,2,3…n yang menghasilkan nilai *d* bulat.

e. Maka terdapat kunci publik (29,1147) dan kunci privat (149,1147)

f. Selanjutnya dimisalkan terdapat kata "HARI INI" yang akan disandikan maka kata tersebut akan dirubah ke dalam bentuk desimal dengan menggunakan tabel ASCII

Maka akan didapatkan sebuah bilangan desimal : 7265827332737873

g. Setelah mendapatkan nilai desimalnya, langkah berikutnya adalah memecah bilangan decimal tersebut menjadi beberapa blok yang berukuran 3 digit, 726|582|733|273|787|3 dan dimasukan dalam variable m_{1-x} (x adalah banyak blok) :

m_1 = 726 m_4 = 273

m_2 = 582 m_5 = 787

m_3 = 733 m_6 = 003 (karena blok terakhir hanya 2 digit,maka ditambahan 0 di depannya)

h. Langkah selanjutnya adalah melakukan proses enkripsi menggunakan rumus $C_1 = {m_1}^e mod n$

Maka :

$C_1 = 726^{29} mod\ 1147 = 415$

$C_2 = 582^{29} mod\ 1147 = 270$

$C_3 = 733^{29} mod\ 1147 = 913$

$C_4 = 273^{29} mod\ 1147 = 1028$

$C_5 = 787^{29} mod\ 1147 = 137$

$C_6 = 003^{29} mod\ 1147 = 176$

Maka didapatkan suatu *chipertext* = 4152709131028137176

i. Setelah didapatkan nilai *chipertext* , berikutnya adalah melakukan proses dekripsi dimana nilai dari *chipertext* tersebut dipecah kembali menjadi blok-blok sesuai dengan hasil perhitungan proses *chipertext*. Proses dekripsi menggunakan rumus $m_1 = {C_1}^d mod n$ maka :

$m_1 = 415^{149} mod\ 1147 = 726$

$m_2 = 270^{149} mod\ 1147 = 582$

$m_3 = 913^{149} mod\ 1147 = 733$

$m_4 = 1028^{149} mod\ 1147 = 273$

$m_5 = 137^{149} mod\ 1147 = 787$

$m_6 = 176^{149} mod\ 1147 = 3$

j. Setelah mendapatkan nilai dari proses dekripsi, maka nilai tersebut kembali digabungkan dan didapatkan hasil sebuah *plaintext* 7265827332737873 yang apabila dirubah kembali sesuai bilangan ASCII akan dihasilkan karakter semula yaitu “HARI INI”

3.5 Perancangan Uji Coba

Pengujian pada algoritma RSA dilakukan dengan cara melakukan percobaan *brute force* terhadap kunci yang telah terbentuk. Dalam hal ini, pengujian dilakukan dengan cara melakukan pencarian kunci dengan cara mencoba satu persatu semua bilangan prima dan dihitung jumlah iterasi yang dilakukan. Dengan melakukan perhitungan iterasi tersebut maka dapat dilihat seberapa kuat algoritma pembentukan kunci yang digunakan dalam penelitian ini. Pengujian ini menggunakan rumus 2.8 dan 2.9 dan tabel uji ditunjukan pada tabel 3.1.

Tabel 3.1 Tabel uji coba

p	*q*	*n*	*e*	*d*	iterasi	Waktu (second)

BAB IV
IMPLEMENTASI

Untuk melakukan implementasi sistem yang telah dirancang, diperlukan sebuah lingkungan sistem yang dapat mendukung dan memenuhi kebutuhan program yang akan diterapkan.

4.1 Perangkat Sistem

Lingkungan sistem yang digunakan untuk melakukan implementasi program berupa lingkungan perangkat keras *(hardware)* dan lingkungan perangkat lunak *(software).*

4.1.1 Perangkat Keras *(Hardware)*

Perangkat keras yang digunakan untuk melakukan implementasi program "Kriptografi pada file dokumen *Microsoft Office* menggunakan metode RSA" adalah sebagai berikut :

1. Processor AMD Athlon 2.50 Ghz
2. Memory 2 GB DDR2
3. Hardisk ATA 80 GB
4. VGA NVIDIA GeForce 825MB (On board)
5. Monitor 19"
6. Keyboard + Mouse

4.1.1 Perangkat Lunak *(Software)*

Perangkat lunak *(software)* yang digunakan untuk melakukan implementasi program "Kriptografi pada file dokumen *Microsoft Office* menggunakan metode RSA" adalah sebagai berikut :

1. Microsoft Visual Studio 2010
2. Sistem Operasi Windows 7 32 bit
3. Microsoft Office 2010

4.2 Implementasi Program

Implementasi program ini terbagi menjadi beberapa *class*. Masing – masing *class* memiliki dan fungsi tersendiri dalam proses enkripsi dan dekripsi suatu file dokumen. Tujuan dalam pembagian ini adalah untuk memudahkan dalam penyusunan dan pemakaian tiap – tiap fungsi dan *procedure* yang digunakan dalam aplikasi atau program kriptografi ini.

4.2.1 Proses Sistem Keseluruhan

Proses sistem secara keseluruhan dimulai dari *form* pertama yaitu *MainMenu.* Pada *form* tersebut, dilakukan proses utama dan pemanggilan *class* – *class* yang lain sesusai fungsi dan alur yang telah ditentukan. Berikut adalah *class* – *class* yang digunakan, antara lain :

1. *Class* “krptEnkripsi”

   Class ini merupakan *class* yang digunakan untuk melakukan proses enkripsi. Di dalam *class* ini terdapat sebuah *procedure* yang digunakan untuk pelakukan pemanggilan *class* lain yang berfungsi untuk menghitung tiap - tiap *byte*s yang diambil dari fileyang akan dienkripsi.

2. *Class* “krptDekripsi”

   Class ini mempunyai fungsi yang hampir mirip dengan *class* krptEnkripsi. *Class* ini digunakan untuk melakukan proses perulangan untuk mengambil tiap – tiap bytes yang akan dienkripsi dan selanjutnya akan dilakukan proses perhitungan dengan cara memanggil *class* yang lain.

3. *Class* “krptEratosthenesPrime”

   Class ini digunakan untuk melakukan perhitungan pencarian bilangan prima dengan batasan *n* yang ditentukan oleh user. *Procedure* di dalam *class* ini mengimplementasikan metode pembentukan bilangan prima *Eratosthenes.*

4. *Class* “krptFastExponentiation”

   Class ini berfungsi untuk melakukan perhitungan bilangan exponen dengan cepat. Perhitungan cepat sangat dibutuhkan untuk menghitung *bytes* – *bytes* fileyang akan dienkripsi. Selain cepat metode yang

digunakan dalam perhitungan ini juga menyederhanakan proses perhitungannya sehingga mampu mengatasi bilangan besar.

5. *Class* "krptFindD"

   Class krptFindDini digunakan dalam proses perhitungan *private key*. Hasil perhitungan ini nanti akan digunakan sebagai kunci untuk melakukan proses dekripsi. Kunci privat tersebut akan dikirimkan melalui email kepada pengguna yang berhak melakukan proses dekripsi.

6. *Class* "krptGetFileHeaderBytes"

   Class ini digunakan untuk mengambil *bytes* pada fileyang akan dienkripsi dan dekripsi. *Class* ini akan selalu digunakan dalam proses enkripsi maupun proses dekripsi file.

7. *Class* "krptGreatestCommonDivision"

   Di dalam *class* ini terdapat *procedure* findGCD. *Procedure* ini berfungsi untuk melakukan proses pencarian bilangan yang relative prima terhadap suatu bilangan yang lainya. Proses ini digunakan untuk menentukan *public_key* yang relative prima dengan bilangan *Ѳn (shi n).*

8. *Class* "krptWriteFileBinary"

   Class ini berfungsi untuk melakukan proses penulisan *bytes – bytes* ke dalam suatu file. Di dalam *class* tersebut terdapat dua *procedure* yang berbeda dalam proses penulisan *bytes* yaitu *procedure write bytes* untuk proses enkripsi dan *write bytes* untuk proses dekripsi.

9. *Class* "krptByteReConstruction"

   Class ini digunakan untuk melakukan proses rekonstruksi *bytes* hasil dari perhitungan enkripsi yang bertipe string dirubah ke dalam tipe *bytes* dengan melakukan pemecahan *bytes – bytes* menggunakan *class* ini.

10. *Class* "krptEmailSender"

    Class ini berfungsi untuk melakukan proses pengiriman email kepada alamat email yang dimasukan oleh *user*. Proses ini mengirimkan *private key* dan *N* kepada pengguna yang mempunyai hak untuk melakukan proses dekripsi. Alamat email yang akan menerima pesan

yang berisikan *private key* dan *N* ini akan diset oleh pengguna yang melakukan proses enkripsi file.

4.2.2 Implementasi Proses Enkripsi

Proses enkripsi merupakan proses awal dari program kriptografi. Di dalam proses ini terdapat beberapa *procedure* yang dibutuhkan untuk mendapatkan hasil enkripsi yang sesuai dengan metode RSA. Berikut adalah *procedure* detail dari implementasi proses enkripsi.

4.2.2.1 Procedure pembacaan file bytes

Di dalam program, *procedure* ini berada di dalam *class* krptGetFileHeaderBytes.vb. Berikut adalah penggalan kode dari *class* krptGetFileHeaderBytes.vb:

```
1  PublicFunction getBinerData(ByVal path AsString) AsByte()
2  Dim streamBinary AsNewFileStream(path, FileMode.Open)
3  Dim readInput AsNewBinaryReader(streamBinary)
4  Dim lengthFile AsInteger = getLengthData(path)
5  Dim input AsByte() = readInput.ReadBytes(lengthFile)
6
7          streamBinary.Close()
8          readInput.Close()
9
10 Return input
11
12 EndFunction
13
14 PrivateFunction getLengthData(ByVal path AsString)
15 Dim info AsNewFileInfo(path)
16 Return info.Length
17       EndFunction
```

***Source Code* 4.1 *Procedure* getBinerData**

Source code 4.1 merupakan proses pembacaan *bytes* suatu fileyang akan dienkripsi. Di dalam kode tersebut terdapat dua *procedure*, yaitu getBinerDatadan getLengthData. *Procedure* yang pertama adalah *procedure* yang digunakan untuk membuka suatu filemenggunakan *FileStream*, dan dibaca menggunakan *BinaryReader*. Pada *procedure* yang pertama ini membutuhkan suatu file *info* dari file dan proses pengambilan file *info* tersebut diimplementasikan pada *procedure* yang kedua yaitu getLengthData, dimana *procedure* tersebut mengembalikan suatu panjang *bytes* dari filetersebut (baris 14-17). Setelah kebutuhan untuk

membaca *bytes* dari fileyang akan dienkripsi didapatkan, maka langkah selanjutnya adalah melakukan pembacaan dan penyimpanan *bytes* ke dalam suatu *array* yang mempunyai tipe *byte* (baris 5). Setelah melakukan penyimpanan *byte*, maka dilakukan proses penutupan file *stream* dan *BinaryReader* (baris 7 dan 8). Dari *procedure* inilah *bytes* – *bytes* tersebut akan dikembalikan ke dalam *class* enkripsi utama untuk selanjutnya diolah kembali.

4.2.2.2 Procedure untuk mendapatkan public key

Proses ini terdapat dalam *class* yang bernama krptGreatestCommonDivison.vb. *Procedure* ini berfungsi untuk mendapatkan *public key* yang dibutuhka dalam proses enkripsi. Berikut adalah penggalan kode dari *class* krptGreatestCommonDivision.vb.

```
1  PublicFunction findGCD(ByVal shi_n AsInteger, ByVal prime() AsString) AsString
2  Dim a, b, c AsInteger
3  Dim sisa AsInteger = Nothing
4  Dim e_key AsString = Nothing
5  Dim b_temp AsInteger
6
7          a = shi_n
8  For i = 0 To prime.Length - 2
9              b = Integer.Parse(prime(i))
10             b_temp = b
11 Do
12                 c = a \ b
13                 sisa = a Mod b
14 If sisa = 0 Then
15 Exit Do
16 EndIf
17                 a = b
18                 b = sisa
19 Loop
20 If b = 1 Then
21 'pilih e yang paling besar atau e yang paling terakhir
22                 e_key = b_temp.ToString
23 EndIf
24 Next
25 Return e_key
26 EndFunction
```

***Source Code* 4.2 *Procedure* findGCD**

Pada *source code* 4.2, terdapat sebuah *procedure* yang bernama findGCD dan mempunyai parameter berupa *shi_n* dan *prime* (baris 1). Variabel *prime* tersebut merupakan *array* dari bilangan prima yang nantinya akan diolah untuk mendapatkan bilangan prima yang relative prima dengan *shi_n*. Pada prosesnya,

terdapat tiga variabel yang masing – masing adalah *a,b* dan *c* (baris 2). Dalam hal ini *a* merupakan *shi_n* sedangkan *b* adalah bilangan prima yang *looping* (baris 7 & 8). Untuk tiap – tiap bilangan prima yang terdapat di dalam *arrayprime* akan dihitung dengan *shi_n*. Pertama – tama *shi_n* akan dibagi dengan *b* yang hasilnya nanti akan dimasukan ke dalam variabel *c* (baris 12). Selanjutnya dicari sisa modulus antara variabel *a* dengan *b* dan hasilnya akan disimpan ke dalam variabel *sisa* (baris 13). Apabila hasil dari modulus sebelumnya adalah 0 maka akan keluar dari *loopingdo-loop* (baris 14-16). Tetapi jika tidak maka akan dilakukan *looping* terus menerus hingga hasil modulus bernilai 0. Selama *looping* maka akan dilakukan pertukaran nilai dari variabel *a,b* dan *c* dimana *a* akan ditukar dengan nilai dari *b* dan nilai dari variabel *b* akan diisi dengan nilai dari hasil sisa modulus sebelumnya (baris 17 & 18). Pada saat *looping for* akan dicari nilai variabel *b* yang menghasilkan nilai 1 (baris 20-23). Apabila syarat tersebut terpenuhi maka akan disimpan sebagai *public key* dan begitu seterusnya hingga perulangan *for* berakhir. Secara otomatis nilai dari *public key* tersebut merupakan bilangan yang mempunyai nilai paling besar. Selanjutnya nilai dari *public key* tersebut akan dikembalikan ke dalam proses sebelumnya.

4.2.2.3 Procedure untuk mendapatkan private key

Procedure ini merupakan suatu proses yang digunakan untuk mendapatkan nilai dari *privat key* yang akan digunakan dalam proses enkripsi. *Class* yang menampung *procedure* ini adalah *class* krptFinD.vb. Berikut adalah potongan kode yang terdapat dalam *class* krptFindD.v*b.*

```
1  PublicFunction find_d(ByVal e AsInteger, ByVal shi_n AsInteger) AsString
2
3  Dim d_stat AsInteger
4  Dim status AsBoolean = False
5  Dim result AsString
6  Dim d AsInteger
7  Dim k AsInteger
8
9          k = 0
10 While status = False
11             k += 1
12             d_stat = (1 + (k * shi_n)) Mod e
13 If d_stat = 0 Then
14                 d = (1 + (k * shi_n)) \ e
15                 result = d.ToString
```

```
16  Return result
17                  status = True
18  Else
19                  status = False
20  EndIf
21  EndWhile
22  Return vbNull
23  EndFunction
```

***Source Code* 4.3 *Procedure* find_d**

Pada *source code* 4.3, terdapat sebuah fungsi *find_d* yang memiliki parameter berupa variabel *e* dan *shi_n* (baris 1). Di dalam proses ini digunakan perulangan *while-do* dengan syarat berupa variabel boolean yang bernama *status* (baris 10). Apabila *status* masih bernilai *false* maka akan dilakukan perulangan sampai *status* bernilai *true*. Di dalam kode di atas, terdapat variabel *k*, variabel tersebut merupakan variabel yang nilainya akan bertambah satu setiap sekali *looping* (baris 11). Ini bertujuan untuk mencari nilai *k* yang dapat membuat variabel *d* mempunyai nilai bulat setelah dilakukan operasi modulus *e* (baris 12). Jika nilai dari variabel *d* sudah menjadi bulat, maka proses akan keluar dari *loop* dan akan mengembalikan nilai dari variabel *d* (baris 13-18).

4.2.2.4 Procedure untuk mengirimkan email

Procedure ini digunakan untuk mengirimkan informasi tentang proses enkripsi kepada pengguna lainnya. *Procedure* ini ada di dalam *class* krptEmailSender. Setelah proses pembangkitan *private key* dan *N*, maka selanjutnya nilai dari masing – masing variabel tersebut akan dikirimkan kepada pengguna yang berhak untuk melakukan proses dekripsi file. Berikut adalah penggalan kode dari *class* krptEmailSender.vb.

```
1   #Region"Get_Data_Mail_From_User"
2   PublicSub sendMail(ByVal accountEmail AsString, ByVal password
    AsString, ByVal destAddres AsString, ByVal subject AsString, ByVal
    bodyMessages AsString)
3           account = accountEmail
4           pass = password
5           smtp = "smtp.gmail.com"
6           destinationAddres = destAddres
7           subjectMail = subject
8           bodyMessage = bodyMessages
9           btnSend_Click()
10  EndSub
11  #EndRegion
```

```
12
13  #Region"Button_Send"
14  PrivateSub btnSend_Click()
151
61  Dim mail AsNewMailMessage
7   Dim smtpClient AsNewSmtpClient
18  Dim smtpFailedReceipient AsNewSmtpFailedRecipientsException
19
20  Try
21              mail.From = NewMailAddress(account)
22              mail.To.Add(destinationAddres)
23              mail.Subject = subjectMail
24              mail.Body = bodyMessage
25              mail.IsBodyHtml = True
26              mail.DeliveryNotificationOptions =
    DeliveryNotificationOptions.OnFailure
27              smtpClient.Host = smtp
28              smtpClient.Port = 587
29              smtpClient.EnableSsl = True
30              smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network
31              smtpClient.UseDefaultCredentials = False
32              smtpClient.Credentials = New
    Net.NetworkCredential(account, pass)
33              smtpClient.Send(mail)
34  'MessageBox.Show(smtpFailedReceipient.FailedRecipient)
35  Catch ex AsException
36              ExceptionTrans(ex.Message)
37  Exit Sub
38  EndTry
39  MessageBox.Show("Pesan kunci dekripsi berhasil dikirim pada pukul :
    "&DateTime.Now.Hour &
40  ":"&DateTime.Now.Minute &":"&DateTime.Now.Second)
41  EndSub
42  #EndRegion
43
44  #Region"Exception_Handle"
45  PrivateSub ExceptionTrans(ByVal messages AsString)
46  If messages.Contains("The specified string is not in the form
    required for an e-mail address") Or _
47  messages.Contains("An invalid character was found in the mail
    header") Then
48  MessageBox.Show("Format alamat email yang anda masukan salah, cek
    kembali alamat email tujuan")
49  EndIf
50  EndSub
51  #EndRegion
```

***Source Code* 4.4 *Class* krptEmailSender**

Pada *source code* 4.4 terdapat tiga *procedure*, yaitu sendMail, btnSendClickdan ExceptionTrans.*Procedure* sendMail berfungsi untuk melakukan deklarasi awal mulai dari *email account, password, email* yang akan dituju, isi pesan dan judul pesan (baris 2-10). Setelah semuanya dimasukan dan diproses oleh *procedure* tersebut, langkah selanjutnya adalah melakukan *setting* pada *email client* (baris 21-33).*Settingemail* untuk tiap – tiap *account* memiliki

perbedaan *port number*, namun dalam skripsi ini, program menggunakn *account* dari *google* sesuai yang ada di dalam batasan masalah. *Procedure* ExceptionTrans, digunakan untuk melakukan proses *handling error* apabila *user* melakukan kesalahan pada saat memasukan alamat *email* yang akan dituju (baris 45-49).

4.2.2.5 Procedure untuk enkripsi

Procedure ini merupakan *procedure* yang paling utama dalam aplikasi kriptografi. *Procedure* ini terdapat dalam *class* krptEnkripsi.vb. Di dalam *class* tersebut terdapat rumus yang digunakan untuk merubah *plaintext* mendjadi *chipertext* menggunakan rumus yang sesuai dengan metode RSA. Berikut adalah penggalan kode dari *class* krptEnkripsi.vb.

```
1  PublicFunction hitungRumus(ByVal arrBytes AsByte(), ByVal e
   AsInteger, ByVal n AsInteger)
2  Dim rmsFastE AsNewkrptFastExponentiation
3  Dim arrBytesEncrypted(511) AsString
4  For i = 0 To 511
5  arrBytesEncrypted(i)=rmsFastE.rumusFastExponentiation(arrBytes(i),e,n
   )
6  Next
7  Return arrBytesEncrypted
8  EndFunction
```

***Source Code* 4.5 *Procedure* hitungRumus**

Pada *source code* 4.5 terdapat fungsi yang digunakan untuk melakukan perhitungan rumus sesuai dengan metode yang digunakan yaitu RSA. *Procedure* tersebut memiliki beberapa parameter sebagai *input* untuk melakukan perhitungan. Parameter tersebut adalah *arrBytes,e* dan *N* (Baris 1). Parameter yang pertama adalah *arrBytes* yang memiliki tipe *array bytes*. Parameter inilah yang nanti akan dihitung dan dirubah menjadi *chipertext* menggunakan perulangan untuk menghitung satu persatu *bytesfile* yang akan dienkripsi. Parameter selanjutnya adalah *e*, parameter ini memiliki tipe *integer*. Parameter ini adalah *public key* yang akan menjadi kunci dalam proses enkripsi. Sedangkan parameter yang terakhir adalah *N*. Parameter ini juga memiliki tipe *integer* sama dengan *e*. Parameter ini juga nantinya akan dibutuhkan dalam rumus enkripsi

yang menggunakan metode RSA. Di bawah ini adalah proses perulangan dan rumus yang digunakan untuk merupah *plaintext* menjadi *chipertext.*

Pada *source code* 4.5 dilakukan perulangan atau *looping* sebanyak 512 kali, ini karena panjang *bytes* yang dienkripsi memiliki panjang 512 *bytes* (baris 4-6). Setelah semua *bytes* telah dihitung dan dirubah kedalam bentuk *chipertext*, maka selanjutnya kumpulan *bytes* yang telah dienkripsi tersebut ke dalam proses utama.

4.2.2.6 Procedure untuk melakukan proses rewrite pada file

Dalam proses ini, *array* yang telah dienkripsi selanjutnya akan dimasukan ke dalam *procedure* penulisan ulang atau *rewrite* ke dalam *file* yang akan dienkripsi. Berikut adalah penggalan kode dari *class* krptWriteFileBinary.vb.

```
1  PublicSub writeForEncrypt(ByVal arrayByteWillEncryptedString() AsString, ByVal arrayByteNotEncryptedString() AsString, ByVal outPath AsString)
2  Dim arrayListbyte(0) AsByte
3  Dim index AsBigInteger = 0
4  Dim indexInc AsBigInteger = 0
5  Dim longByte AsInteger
6  Dim print AsNewPrintOut
7
8  'Proses yang digunakan untuk melakukan proses pemecahan byte yang telah di enkripsi dan dimasukan ke dalam array Byte
9  For i = 0 To arrayByteWillEncryptedString.Length - 1
10             index += arrayByteWillEncryptedString(i).Length
11 ReDimPreserve arrayListbyte(index)
12             longByte = arrayByteWillEncryptedString(i).Length - 1
13 For j = 0 To arrayByteWillEncryptedString(i).Length - 1
14 If j = 0 Then
15    arrayListbyte(indexInc)= Byte.Parse(arrayByteWillEncryptedString(i).Substring(0, 1) & longByte)
16                     indexInc += 1
17 Else
18                     arrayListbyte(indexInc) = Byte.Parse(arrayByteWillEncryptedString(i).Substring(j, 1))
19                     indexInc += 1
20 EndIf
21 Next
22 Next
23
24 'Proses pengosongan 4 array awal
25 Dim indexArr AsInteger = 4
26 Dim tempArrayByteEncrypted(arrayListbyte.Length + 3) AsByte
27 For i = 0 To arrayListbyte.Length - 1
28             tempArrayByteEncrypted(indexArr) = arrayListbyte(i)
29             indexArr += 1
30 Next
31 'Proses perulangan yang digunakan untuk memasukan panjang file header yang dienkripsi ke dalam array yang telah terenkripsi
```

```
32  Dim indexInc2 AsInteger = 0
33  Dim lengthByteEncrypted AsBigInteger = tempArrayByteEncrypted.Length
    - 1
34  For i = 0 To 3
35  If lengthByteEncrypted > 255 Then
36                  lengthByteEncrypted = lengthByteEncrypted - 255
37                  tempArrayByteEncrypted(i) = 255
38  ElseIf lengthByteEncrypted <= 255 Then
39                  tempArrayByteEncrypted(i) = lengthByteEncrypted
40                  lengthByteEncrypted = 0
41  ElseIf lengthByteEncrypted = 0 Then
42                  tempArrayByteEncrypted(i) = 0
43  EndIf
44  Next
45
46  'Proses penggabungan array file header yang di enkripsi dengan byte
    yang tidak terenkripsi
47  Dim lastIndexInTempArray AsInteger = tempArrayByteEncrypted.Length
48  Dim indexForTempArray AsInteger = 0
49  Dim    finalArrayByteEncrypted(((tempArrayByteEncrypted.Length)    +
    (arrayByteNotEncryptedString.Length)) - 1) AsByte
50  For i = 0 To tempArrayByteEncrypted.Length - 1
51              finalArrayByteEncrypted(i)=
    tempArrayByteEncrypted(indexForTempArray)
52              indexForTempArray += 1
53  Next
54  For i = 0 To arrayByteNotEncryptedString.Length - 1
55              finalArrayByteEncrypted(indexForTempArray)=
    Byte.Parse(arrayByteNotEncryptedString(i))
56              indexForTempArray += 1
57  Next
58          writeFileBinary(finalArrayByteEncrypted, outPath)
59  EndSub
60
```

***Source Code* 4.6 *Procedure* writeForEncrypt**

Source Code 4.6 merupakan sebagian dari *class* krptWriteFileBinary. Di dalam kode di atas, terdapat dua *procedure* yang memiliki fungsi masing – masing. *Procedure* writeForEncrypt merupakan *procedure* yang digunakan untuk menyatukan *bytes* yang telah dienkripsi dengan *bytes* sisa yang tidak ikut dienkripsi. Sehingga kedua *byte* tersebut digabungkan kembali menjadi satu antara yang terenkripsi dengan *byte* asli. Namun dalam prosesnya terdapat beberapa tahapan yang cukup kompleks. Pertama, *array bytes* yang berasal dari parameter *procedure*, dirubah terlebih dahulu ke dalam bentuk *bytes* (baris 9-22). Sebelumnya kumpulan *byte* tersebut bertipe *string*. Ini dikarenakan *bytes* hasil dari perhitungan enkripsi tadi memiliki nilai yang lebih dari 255 sehingga tidak dapat ditampung ke dalam bentuk *bytes*. Untuk mengatasinya, hasil perhitungan tadi dipecah menjadi beberapa digit untuk disimpan ke dalam bentuk *bytes*. Misal

hasil perhitungannya adalah 1203, dikarenakan nilai tersebut lebih dari 255, maka sistem akan memecah nilai tersebut menjadi beberapa digit. Sistem akan menghitung terlebih dahulu panjang digit dari nilai tersebut. Di dalam contoh panjang digit adalah 4, lalu sistem akan melakukan perulangan 3 kali (4-1) untuk memecah nilai menjadi 1,2,0,3. Karena telah menjadi satu digit dan kurang dari 255 maka nilai tersebut dapat ditampung kedalam tipe *bytes*. Namun muncul kendala untuk menyatukannya dalam proses dekripsi. Oleh karena itu, sebelum merubah ke dalam bentuk *byte*, maka digit pertama dari proses pemecahan ditambahkan nilai sesuai dengan panjang digit semula. Di dalam contoh panjang digit adalah 4, sehingga digit pertama akan menjadi 13 (4-1) dan diikuti dengan digit yang dipecah yaitu 2,0,3. Sehingga saat dibaca dalam proses dekripsi nanti, sistem dapat mengetahui nilai asli dari *bytes* tersebut dengan membaca *bytes* yang mempunyai panjang 2 digit dan dari digit terakhir itulah panjang *bytes* yang akan digabungkan menjadi 1203. Langkah selanjutnya *array bytes* yang digunakan untuk menggabungkan *byte* terenkripsi dengan byte yang tidak terenkripsi tersebut diambil 4 *byte* awal (baris 25-30). 4 *byte* awal ini merupakan slot yang disediakan untuk menampung panjang *bytes* yang terenkripsi. Cara ini dipilih untuk memudahkan dalam proses dekripsi, agar sistem tidak salah mengambil *byte* untuk dirubah kembali ke *plaintext*. Langkah selanjutnya, Setelah penambahan panjang *bytes* yang terenkripsi sudah disisipkan di 4 *bytes* awal, langkah terakhir adalah proses penulisan kembali *bytes-bytes* tersebut ke dalam *file* yang akan dienkripsi. Berikut adalah kode untuk proses *rewrite* ke dalam *file*.

```
1  PublicSub writeFileBinary(ByVal bytesData AsByte(), ByVal outpath
   AsString)
2  Dim fs As System.IO.FileStream
3      fs = New System.IO.FileStream(outpath, System.IO.FileMode.Create)
4      fs.Write(bytesData, 0, bytesData.Length)
5      fs.Close()
6  EndSub
```

***Source Code* 4.7 *Procedure* writeFileBinary**

Setelah selesai proses penulisanya, maka *file* tersebut secara otomatis tidak akan bisa dibuka dikarenakan nilai dari *bytes* yang ada di dalam *file* tersebut telah berubah.

4.2.3 Implementasi Proses Dekripsi

Setelah proses enkripsi selesai, langkah selanjutnya adalah proses dekripsi. Pengguna yang berhak melakukan proses dekripsi akan mendapatkan kunci *privat* untuk melakukan proses dekripsi, karena tanpa mengetahui berapa nilai dari *privat key* tersebut maka pengguna tidak akan bisa membuka dokumen tersebut. Berikut adalah *procedure – procedre* yang digunakan dalam proses dekripsi.

4.2.3.1 Procedure untuk melakukan proses rekonstruksi bytes

Proses ini merupakan proses kedua setelah pembacaan *bytes*, namun untuk pembacaan *bytes* sama dengan pembacaan *bytes* pada proses enkripsi, sehingga tidak dijelaskan kembali pada implementasi proses dekripsi. Proses rekonstruksi merupakan proses yang sangat penting. Pada tahap ini, *bytes* yang akan di dekripsi akan disusun kembali seperti susunan awal setelah proses enkripsi. Selain itu proses ini juga harus sangat tepat dalam menyusun kembali, ini dikarenakan apabila terjadi kesalahan penyusunan ulang hanya pada satu *bytes* saja maka perhitungan proses dekripsi akan menghasilkan nilai yang berbeda dan dokumen tetap tidak akan bisa dikembalikan seperti semula. Berikut adalah potongan kode untuk *classkrptReConstruction.*

```
1  PublicFunction setBytes(ByVal bytes() AsByte) AsString()
2  Dim digit1 AsString
3  Dim digit2 AsString
4  Dim lengthOfBytes AsInteger = 0
5  Dim byteString(bytes.Length) AsString
6
7  For i = 0 To bytes.Length - 1
8              byteString(i) = bytes(i)
9  Next
10 Do
11 If byteString(indexBytes).Length = 2 Then
12                 digit1 = byteString(indexBytes).Substring(0, 1)
13                 digit2 = byteString(indexBytes).Substring(1, 1)
14 If digit2.Equals("0") Then
15  newBytes(indexNewBytes) = digit1
16 Else
17  newBytes(indexNewBytes) = digit1
18 For j = 0 ToInteger.Parse(digit2) - 1
19  indexBytes += 1
20  newBytes(indexNewBytes) &= byteString(indexBytes)
21 Next
22 EndIf
23 Else
24   newBytes(indexNewBytes) = byteString(indexBytes)
25 EndIf
26 If indexNewBytes = 511 Then
```

```
27  Exit Do
28  EndIf
29   indexNewBytes += 1
30   indexBytes += 1
31  ReDimPreserve newBytes(indexNewBytes)
32  Loop
33  Return newBytes
34  EndFunction
```

***Source Code* 4.8 *Procedure* setBytes**

Source Code 4.7 merupakan kode yang digunakan untuk melakukan proses rekonstruksi *array bytesfile* yang akan di dekripsi. Nama dari *procedure* ini adalah *setBytes*. Parameter yang menjadi *input* untuk *procedure* ini adalah berupa *array bytes* hasil dari hasil pembacaan *bytes* pada *file* yang telah dienkripsi. Parameter *array bytes* dari *file* yang akan dienkripsi tersebut masih berupa *string*, oleh karena itu dilakukan proses perubahan tipe dari tipe *string* ke dalam bentuk *bytes*. Selanjutnya adalah proses penyusunan kembali *bytes* – *bytes* ke susunan semula seperti pada saat susunan *bytes* setelah enkripsi. Pada tahapan ini, program akan membaca tiap – tiap *bytes* yang dimasukan ke dalam perulangan atau *looping*. Saat panjang *bytes* yang terbaca bernilai 2 digit, maka *bytes* inilah yang perlu dilakukan penyusunan ulang, sedangkan *bytes* yang memiliki panjang 1 digit, maka *bytes* tersebut akan diabaikan dan akan tetap dicetak bernilai sama dengan sebelumnya. Pada saat program membaca *bytes* yang memiliki panjang 2 digit, maka langkah berikutnya sistem akan memecah *bytes* tersebut menjadi dua bagian. Bagian yang pertama merupakan *bytes* asli, yang nantinya *bytes* tersebut akan menjadi digit pertama. Lalu digit yang kedua merupakan digit yang menunjukan panjang dari digit *bytes* yang asli. Jadi apabila digit yang kedua bernilai 3, maka diasumsikan 3 digit setelah *index array* tersebut merupakan digit yang asli dan selanjutnya ketiganya nanti akan digabungkan dengan *bytes* yang pertama yang pada saat pembacaan pertama kali telah tersimpan. Maka setelah digabungkan nilai tersebut akan dimasukan ke dalam *array* yang memiliki tipe string, ini dikarenakan tipe *bytes* tidak dapat menampung bilangan yang memiliki nilai lebih dari 255. Setelah semua *bytes* selesai diproses dan disusun seperti semula, maka langkah selanjutnya adalah mengembalikan nilai *array* yang

memiliki tipe string dan akan dilakukan proses perhitungan dekripsi RSA oleh *procedure* yang lain.

4.2.3.2 Procedure untuk proses dekripsi

Procedure ini digunakan untuk melakukan proses prehitungan untuk mengembalikan *bytes* yang berupa *chipertext* ke dalam bentuk *bytes* asli atau *plaintext.* Proses ini dilakukan sesuai dengan rumus dekripsi RSA. Berikut adalah potongan kode yang terdapat dalam *class krptDekripsi.*

```
1  PublicFunction DecryptionProcess(ByVal byteString() AsString, ByVal privateKey_d AsInteger, ByVal n AsInteger)
2  Dim fasExponen AsNewkrptFastExponentiation
3  Dim resultInteger AsBigInteger
4  Dim newBytesByte(511) AsByte
5
6  For i = 0 To 511
7  If byteString(i) = NothingThen
8  Exit For
9  EndIf
10  resultInteger = Byte.Parse(fasExponen.rumusFastExponentiation(Integer.Parse(byteString(i)),privateKey_d, n))
11  newBytesByte(i) = resultInteger
12 Next
13
14 Return newBytesByte
15 EndFunction
```

***Source Code* 4.9 *Procedure* DecryptionProcess**

Dalam *source code* 4.9, parameter yang menjadi *input* untuk *procedure* ini adalah *arraybytes* yang memiliki tipe *string*. Selain itu juga terdapat parameter berupa *private key*, dan *n*. Perulangan yang diunakan dalam kode ini adalah dari 0 – 511 atau digunakan untuk mengambil nilai *bytes* – *bytes* yang memiliki panjang 512 *bytes*. Masing – masing *bytes* yang diambil akan dihitung menggunakan rumus dekripsi RSA. Setelah dihitung, tiap – tiap *bytes* nantinya akan disimpan ke dalam suatu *array* yang memiliki tipe *bytes*. Lalu setelah semuanya selesai dihitung, langkah selanjutnya adalah mengembalikan *array bytes* tersebut kedalam proses sebelumnya untuk diproses dan ditulis ke dalam *file* tersebut.

4.2.3.3 Procedure untuk melakukan proses rewrite pada file

Procedure ini merupakan proses yang digunakan untuk melakukan penulisan ulang *bytes-bytes* tersebut ke dalam *file* yang dienkripsi. Langkah ini sama seperti penulisan kembali *bytes* – *bytes* pada saat proses enkripsi. Berikut ini adalah potongan kode yang terdapat dalam *class krptWriteFileBinary*.

```
1  PublicSub writeForDecrypt(ByVal decryptedBytes AsByte(), ByVal
   oriBytes AsByte(), ByVal path AsString)
2  Dim filebytes((512 + oriBytes.Length) - 1) AsByte
3  Dim index AsInteger = 512
4  For i = 0 To 511
5    filebytes(i) = decryptedBytes(i)
6  Next
7
8  For i = 0 To oriBytes.Length - 1
9    filebytes(index) = oriBytes(i)
10   index += 1
11 Next
12
13         writeFileBinary(filebytes, path)
```

***Source Code* 4.10 *Procedure* writeForDecrypt**

Dalam kode di atas parameter yang menjadi *input* adalah *array* yang telah didekripsi, *array* yang belum didekripsi dan alamat *file* yang akan di dekripsi. Pada prosesnya, *array* yang telah didekripsi akan dimasukan kembali ke dalam *array* baru yang memiliki tipe *bytes*. Lalu selanjutnya akan dilakulan dengan *looping* sebanyak 512 kali (sesuai dengan panjang *bytes* yang dienkripsi). Setelah proses terhadap *array* hasil dekripsi selesai dilakukan, selanjutnya proses berganti terhadap *array* yang belum didekripsi namun dimulai dari index 513 atau index setelah *bytes* yang dienkripsi tadi. Ini dimaksudkan untuk menggabungkan *bytes* yang dienkripsi dan dekripsi dengan *bytes* yang tidak terkena proses enkripsi dekripsi. Setelah kedua *array bytes* tersebut menjadi satu pada suatu variabel *array* yang memiliki tipe *bytes*, langkah selanjutnya adalah melakukan penulisan ulang ke dalam *file* yang di dekripsi tadi. Berikut adalah potongan kode yang digunakan untuk melakukan proses penulisan ulang.

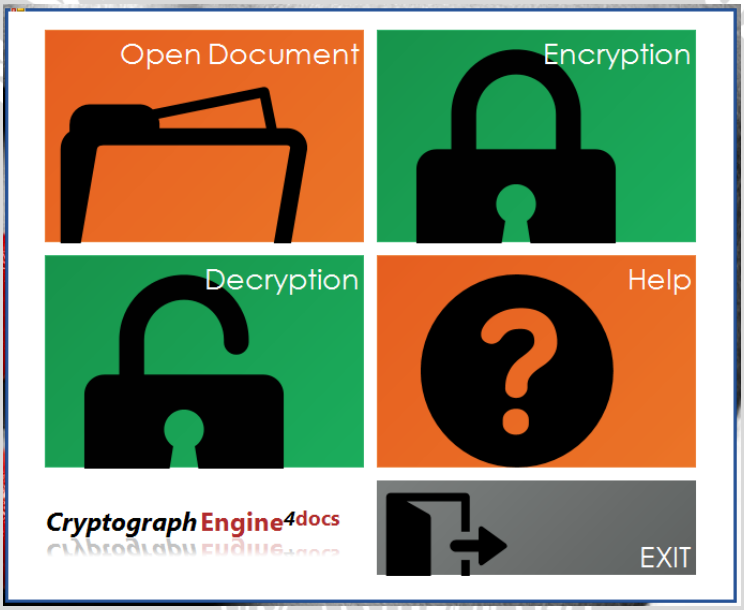
```
1  PublicSub writeFileBinary(ByVal bytesData AsByte(), ByVal outpath
   AsString)
2  Dim fs As System.IO.FileStream
3     fs = New System.IO.FileStream(outpath, System.IO.FileMode.Create)
4           fs.Write(bytesData, 0, bytesData.Length)
5           fs.Close()
6  EndSub
```

***Source Code* 4.11 *Procedure* writeFileBinary**

Source code 4.11 digunakan untk membuka path yang akan didekripsi, setelah itu akan ditulisi dengan *bytes* yang telah digabungkan tadi menggunakan operasi *file stream* yang terdapat dalam VB.Net. Setelah penulisan selesai *file stream* akan ditutup atau *close*.

4.3 Implementasi Antar Muka Aplikasi

Berikut ini adalah implementasi antar muka aplikasi yang ada di dalam program. Program kriptografi ini memiliki *form* utama yang digunakan untuk menyediakan pilihan bagi pengguna dalam menggunakan program ini.



Gambar 4.1 Halaman utama antar muka aplikasi kriptografi

Pada tampilan antar muka di atas terdapat 5 pilihan yang dapat digunakan oleh pengguna untuk menjalankan program kriptografi. Pilihan tersebut terdiri dari :

1. *Open Document*

   Pilihan ini disediakan bertujuan agar pengguna dapat melihat terlebih dahulu dokumen yang akan dienkripsi. Pengguna juga dapat melakukan *editingi* terlebih dahulu dalam sebelum melakukan proses enkripsi *file*.

2. *Encryption*

   Pilihan ini digunakan untuk melakukan proses enkripsi terhadap suatu *file*. Perbedaan pilihan ini dengan sebelumnya adalah dalam pilihan ini pengguna tidak dapat melakukan proses *editing*terhadap *file*. Pengguna hanya melakukan pemilihan terhadap *file* yang akan dienkripsi saja.

3. *Decryption*

   Pada pilihan ini, pengguna dapat melakukan proses terhadap *file* yang akan dipilih untuk didekripsi. Proses ini sama seperti proses *enkripsi* yaitu pengguna akan memilih *file* secara langsung dan memasukan parameter – parameter yang menjadi syarat untuk melakukan proses dekripsi.

4. *Help*

   Pilihan ini disediakan untuk pengguna yang membutuhkan petunjuk dalam pemakain program kriptografi ini. Sehingga diharapkan pengguna dapat menggunakan program ini dengan baik dan benar.

5. *Exit*

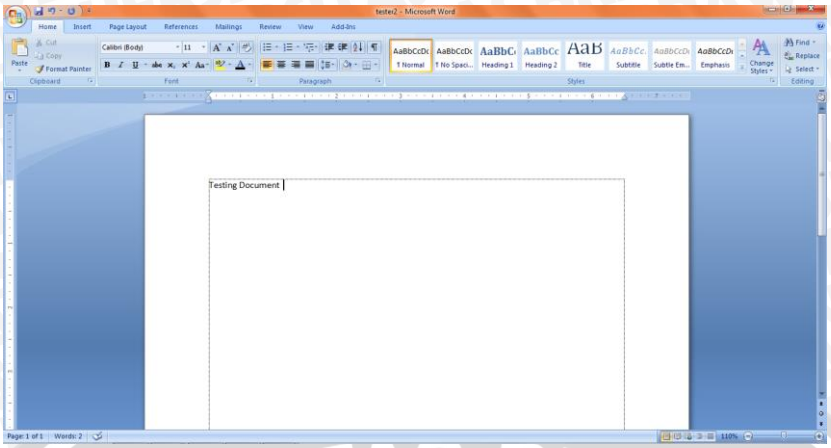
   Pilihan ini digunakan untuk mengakhiri aplikasi atau digunakan untuk menutup aplikasi kriptografi ini.

4.3.1 Halaman *Open Document*

Pada halaman ini pengguna dapat membuka *file* dokumen yang akan dienkripsi. Selain itu pengguna juga dapat melakukan *editing* terhadap *file* tersebut seperti pada *Microsoft Office*. Berikut adalah tampilan halama *open document.*

Gambar 4.2 Antar muka *open dokumen* untuk pilihan tipe dokumen

Pada gambar di atas, terdapat empat pilihan yang dapat dipilih oleh pengguna. Pilihan tersebut sesuai dengan dokumen yang akan dibuka oleh pengguna.

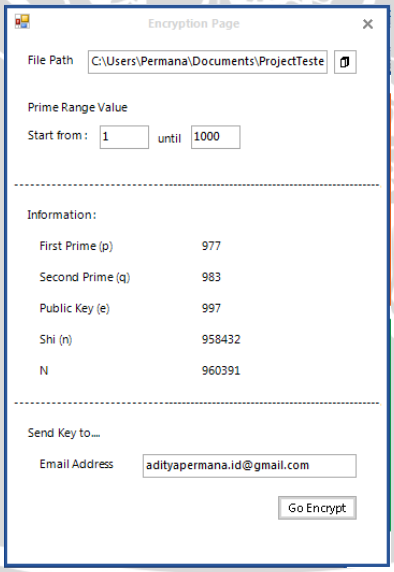


Gambar 4.3 Antar muka *open document*

Gambar 4.3 merupakan antar muka *open document MS.Office Word*. Disini pengguna bebas melakukan *editing.* Sehingga ini dapat memudahkan pengguna tanpa harus membuka terlebih dahulu di *Microsoft Office* untuk melakukan proses *editing*.

4.3.2 Halaman *Encryption*

Pada halaman ini, pengguna dapat memilih *file* dokumen yang akan dienkripsi. Berikut adalah tampilan pada halaman *encryption*.



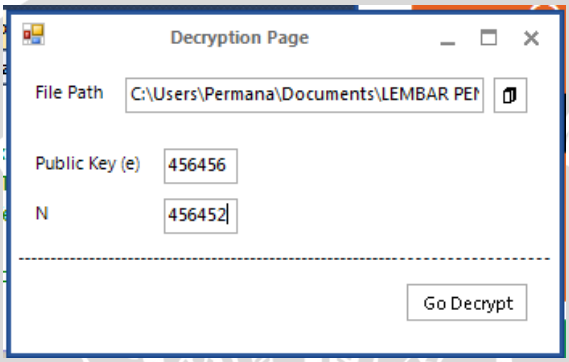
Gambar 4.4 Halaman *Encryption*

Pada halam ini pengguna membuka *file dialog* untuk memilih dokumen yang akan dienkripsi. Selanjutnya pengguna akan memasukan batasan atau *range* yang digunakan untuk menentukan bilangan prima yang akan dipakai dalam perhitungan enkripsi dan dekripsi. Setelah itu pengguna juga diharuskan

memasukan email pengguna yang akan ditujukan untuk melakukan proses dekripsi *file*. Setelah itu, penguna melakukan enkripsi, dan akan muncul beberapa informasi pada halaman ini seperti bilangan prima pertama dan kedua, shi n dan N.

4.3.3 Halaman *Decryption*

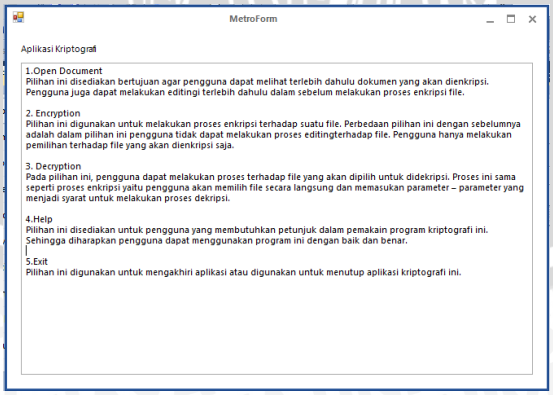
Pada halaman ini pengguna akan melakukan proses pemilihan *file* untuk didekripsi. Berikut ini adalah tampilan antar muka *decryption.*



Gambar 4.5 Halaman *Decryption*

Pada halaman ini pengguna akan melakukan pemilihan *file* yang akan didekripsi menggunakan *file dialog*. Setelah pengguna memilih *file* yang akan dienkripsi, selanjutnya pengguna akan memasukan *public key* dan *N* yang didapat dari *email* pengguna yang memiliki hak untuk melakukan dekripsi *file*. Setelah dimasukan, selanjutnya pengguna akan melakukan proses dekripsi dengan menekan tombol *Go Decrypt.*

4.3.4 Halaman *Help*



Gambar 4.6 Halaman *Help*

Pada halaman ini pengguna akan dapat membaca langkah – langkah atau tahapan – tahapan yang dibutuhkan untuk melakukan proses dekripsi atau proses

enkripsi *file*. Diharapakan dengan disediakannya halaman ini pengguna dapat dengan mudah mendapatkan informasi tentang penggunaan aplikasi kriptografi ini.

BAB V
PEMBAHASAN

Bab ini membahas mengenai tahapan pengujian pada program kriptografi yang menggunakan metode RSA pada dokumen *Microsoft Office*. Pengujian akan dilakukan pada kunci privat yang menjadi kunci untuk melakukan proses dekripsi dokumen. Metode pengujian yang digunakan adalah metode *Brute Force Attack.* Metode ini digunakan dengan cara melakukan iterasi atau perulangan untuk mencari kunci privat.

5.1 Implementasi Uji Coba

Pada implementasi uji coba kunci privat menggunakan serangan metode *brute force*, proses pertama yang dilakukan adalah mencari faktor dari nilai *n* yang diinputkan user dengan cara membagi nilai *n* tersebut dengan bilangan *p* dimana nilai *p* diperoleh dari proses *brute force*. Dari proses tersebut akan dihasilkan nilai *q*. Setiap kali percobaan *brute force* akan menghasilkan dua bilangan *p* dan *q* yang merupakan faktor dari bilangan *n* dan merupakan bilangan prima. Kedua bilangan *p* dan *q* tersebut akan dimasukan kedalam proses pembangkitan kunci dengan menggunakan rumus 2.8 dan 2.9. Setelah program berhasil mendapatkan *d* atau kunci privat, maka akan dicocokan dengan kunci privat yang sebenarnya, apabila memiliki nilai yang sama maka perulangan akan berakhir namun apabila tidak sama akan dilanjutkan ke perulangan berikutnya dengan membagi nilai *n* kembali dengan nilai *p* yang baru yang diperoleh dari proses *brute force*. Proses diatas akan dilakukan secara berturut-turut hingga menemukan *private key* yang diinputkan. Hasil uji coba yang dilakukan untuk mendapatkan kunci privat dengan menggunakan metode *brute force* ditunjukan pada tabel 5.1.

Tabel 5.1 Hasil uji coba *brute force* pada kunci privat dengan *range* 1000 -10000

p	*q*	*n*	*e*	*d*	iterasi	Waktu (second)
977	983	960391	997	313389	976	1
1987	1993	3960091	1999	1304191	1986	2
2963	2969	8797147	2999	6384551	2962	3
3943	3947	15563021	3989	11152589	3942	3
4973	4987	24800351	4999	15750007	4972	5
5939	5953	35354867	5987	28501067	5938	7
6977	6983	48720391	6997	41439101	6976	7
7949	7951	63202499	7993	49700257	7948	8
8963	8969	80389147	8999	11351463	8962	11
9941	9949	98903009	9973	85339117	9940	15

Keterangan :

p = Bilangan prima pertama

q = Bilangan prima kedua

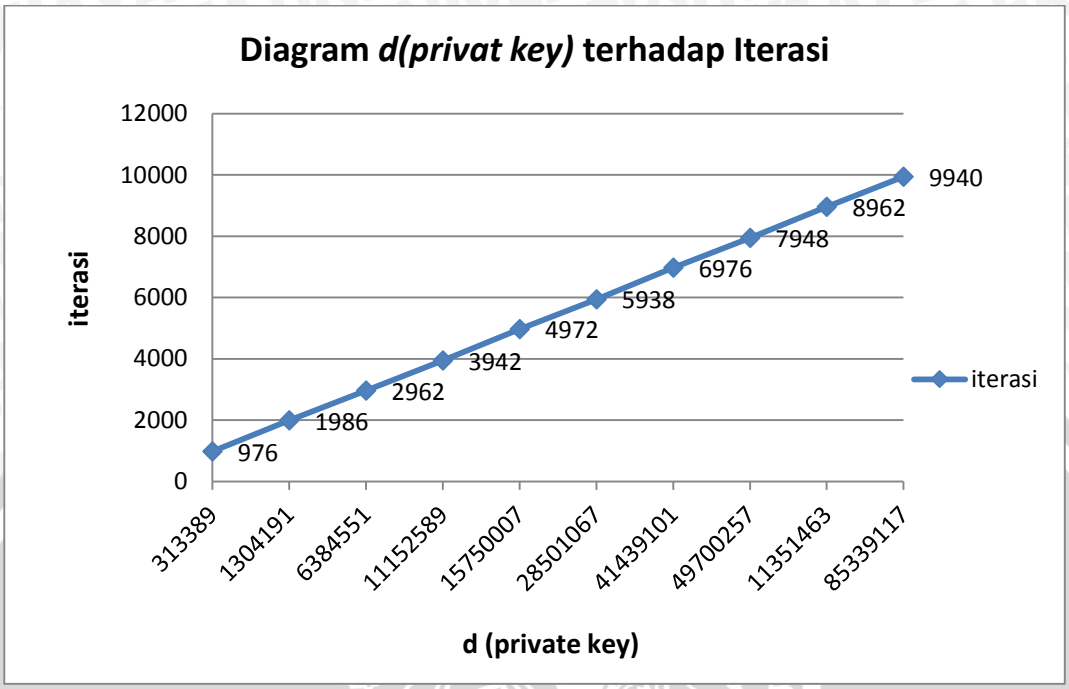
n = Hasil perkalian bilangan *p* dan *q*

e = Kunci publik

d = Kunci privat yang akan dicari

iterasi = Jumlah iterasi yang dibutuhkan untuk mendapatkan kunci privat (*e*)

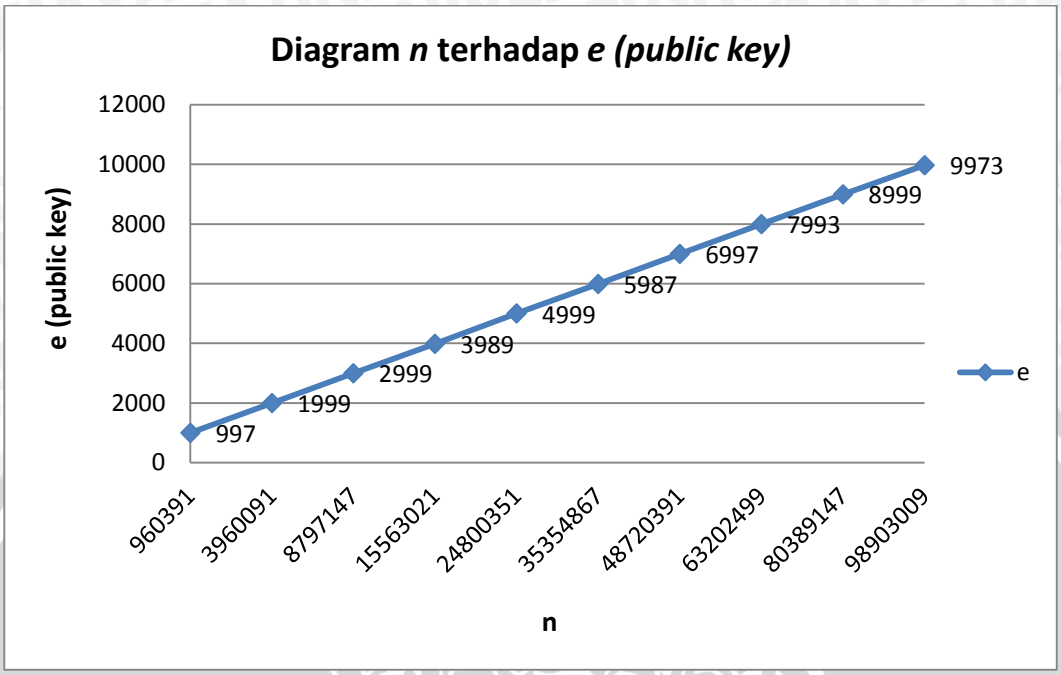
Berdasarkantabel 5.1, didapatkan sebuah diagram kunci privat terhadap jumlah iterasi yang ditunjukan pada gambar 5.1



Gambar 5.1 Diagram *d (private key)* terhadap iterasi

Berdasarkan hasil yang digambarkan pada gambar 5.1 disana terlihat suatu diagram yang memiliki tipe liniear, dimana hubungan antara panjang kunci privat terhadap jumlah iterasi berbanding lurus. Ini terbukti dari semakin besar kunci yang digunakan maka iterasi yang dibutuhkan untuk memecahkan atau menebak kunci privat semakin banyak pula iterasi yang dibutuhkan. Nilai atau jumlah iterasi yang terlihat dalam gambar 5.1 mempunyai pola naik teratur ini membuktikan bahwa cepat atau lama proses yang dibutuhkan untuk menebak kunci privat tergantung pada panjang kunci privat itu sendiri. Oleh sebab itu penemu RSA menyarankan kepada pengguna metode RSA untuk dapat membangkitkan kunci privat dengan panjang lebih dari 200 digit kunci. Dengan menggunakan semakin banyak digit maka semakin sulit dan lama untuk mendapatkan sebuah kunci dengan metode *brute force*.

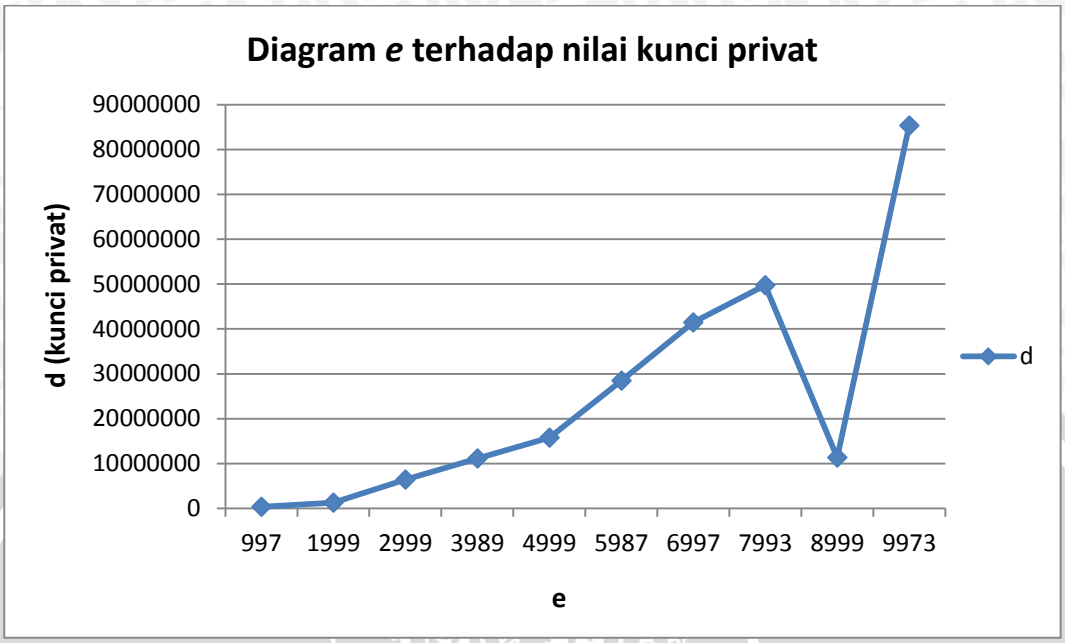
Selanjutnya berdasarkan tabel 5.1 dapat digambarkan menggunakan diagram hubungan antara nilai *n* terhadap nilai dari publik key.



Gambar 5.2 Diagram *n* terhadap nilai *e (public key)*

Nilai *n* merupakan nilai yang dihasilkan dari perkalian dua bilangan prima besar yaitu *p* dan *q*. Dengan menggunakan tabel 5.1, dapat dihasilkan sebuah diagram yang seperti pada gambar 5.2 yang merupakan diagram suatu hubungan antara nilai *n* dengan kunci publik. Dalam gambar 5.2 tersebut terlihat nilai *e* atau publik key sebanding dengan nilai *n*. Berdasarkan diagram tersebut, semakin besar nilai *n* maka semakin besar pula kunci publik yang dibangkitkan, atau dapat dikatakan juga semakin besar nilai *p* dan *q* maka semakin besar pula kunci publik yang dibangkitkan karena *n* merupakan perkalian antara *p* dan *q*. Oleh karena itu untuk meningkatkan keamanan suatu kunci maka gunakan bilangan prima *p* dan *q* sebesar mungkin agar kunci publik yang dibangkitkan juga menghasilkan nilai yang besar.

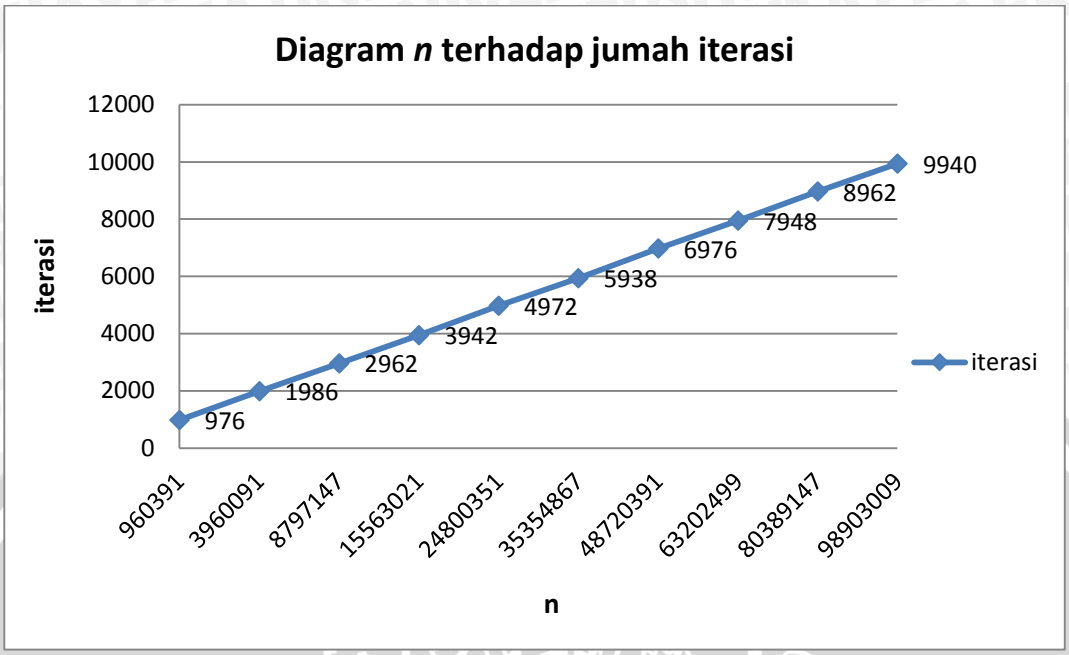
Selain itu pada tabel 5.1 juga dapat digambarkan sebuah diagram hubungan antara *e* (kunci privat) terhadap nilai *d* (kunci privat).



Gambar 5.3 Diagram hubungan nilai *e* dengan nilai kunci privat

Pada diagram yang ditunjukan oleh gambar 5.3 memiliki bentuk yang berbeda daripada bentuk diagram yang lain. Pada gambar 5.3 diperlihatkan sebuah nilai *d* atau yang disebut juga sebuah privat key memiliki bentuk atau pola naik turun. Pada diagram tersebut nilai dari kunci privat akan berubah naik hingga nilai *e* = 7993 namun setelah itu terdapat satu nilai *e* yang menghasilkan nilai *d* atau kunci publik yang kecil daripada privat key. Ini dikarenakan pada saat melakukan perhitungan menggunakan rumus 2.9 nilai *k* merupakan hasil dari iterasi nilai 1 hingga nilai *x* yang menghasilkan bilangan bulat *d.* Alasan nilai kunci privat tersebut turun adalah pada saat nilai *e* atau kunci publik 8999 dan dengan nilai *k* yang kurang dari nilai *k* pada percobaan kunci sebelumnya, telah menghasilkan nilai *d* bulat sehingga menghasilkan nilai *d* yang kurang dari nilai *d* pada percobaan sebelumnya. Jadi menurut diagram yang terdapat dalam gambar 5.3 menunjukan tidak selalu nilai *e* besar akan menghasilkan sebuah kunci privat yang besar juga. Namun menurut diagram, rata-rata nilai *d* akan berbanding lurus dengan nilai *e*.

Berdasarkan tabel 5.1 dapat digambarkan sebuah diagram hubungan *n* terhadap iterasi seperti pada gambar 5.3



Gambar 5.4 Diagram *n* terhadap jumlah iterasi

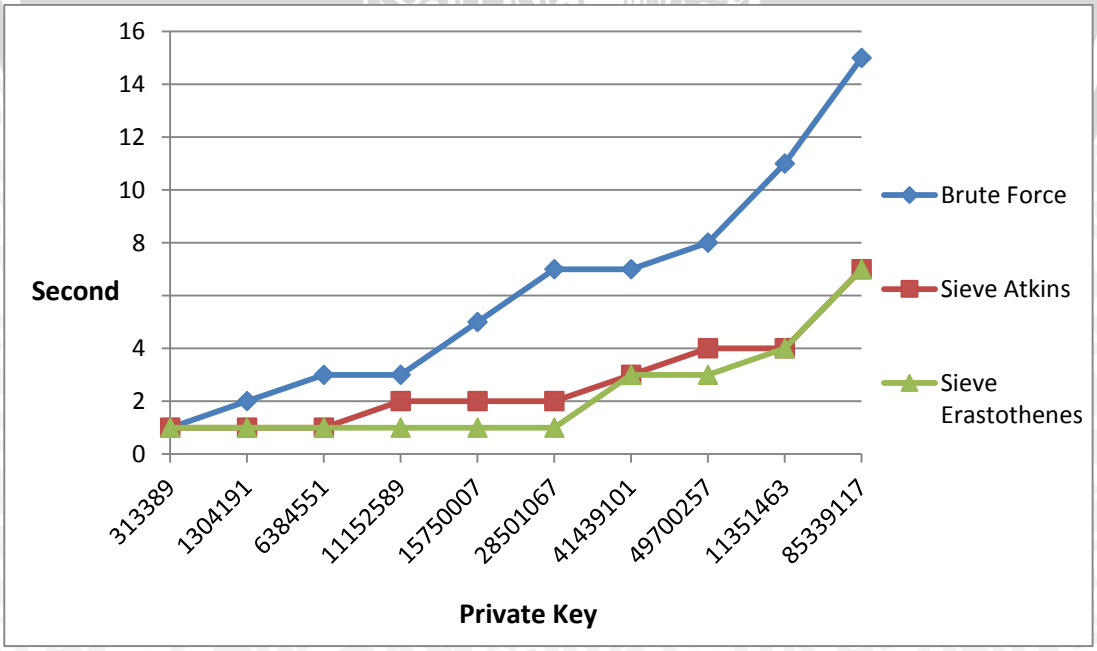
Berdasarkan gambar 5.4 dapat menunjukan bahwa jumlah *n* yang terbentuk dari perkalian dua bilangan prima *p* dan *q* mempengaruh banyaknya iterasi yang terjadi dalam melakukan *brute force attack* terhadap kunci privat. Semakin besar nilai *n* maka semakin besar pula nilai iterasi yang dibutuhkan untuk dapat mendapatkan sebuah kunci privat. Sesuai pada diagram pada gambar 5.4 nilai iterasi tidak mengalami penurunan, ini dikarenakan nilai dari *n* semakin meningkat sehingga jumlah iterasi yang dibutuhkan juga semakin meningkat seiring peningkatan nilai dari *n*.

Pengujian selanjutnya adalah membandingkan waktu pencarian kunci privat menggunakan metode brute force dengan beberapa pilihan metode pembangkitan bilangan prima. Tabel 5.2 adalah hasil pengujian *brute force* terhadap kunci privat menggunakan beberapa metode pemilihan bilangan prima yaitu *brute force, sieve atkins* dan *sieve erastothenes*.

Tabel 5.2 Hasil pengujian mengunakan metode pembangkitan bilangan prima *brute force*

No	*p*	*q*	*n*	*e*	*d*	iterasi	Waktu (*Brute Force*)	Waktu (Sieve Atkins)	Waktu (Sieve Erastothenes)
1	977	983	960391	997	313389	976	1	1	1
2	1987	1993	3960091	1999	1304191	1986	2	1	1
3	2963	2969	8797147	2999	6384551	2962	3	1	1
4	3943	3947	15563021	3989	11152589	3942	3	2	1
5	4973	4987	24800351	4999	15750007	4972	5	2	1
6	5939	5953	35354867	5987	28501067	5938	7	2	1
7	6977	6983	48720391	6997	41439101	6976	7	3	3
8	7949	7951	63202499	7993	49700257	7948	8	4	3
9	8963	8969	80389147	8999	11351463	8962	11	4	4
10	9941	9949	98903009	9973	85339117	9940	15	7	7

Dari hasil tabel 5.2 dapat digambarkan sebuah diagram untuk membandingkan hasil dari pengujian dengan menggunakan beberapa metode pembangkitan bilangan prima. Diagram tersebut membandingkan waktu tempuh yang dibutuhkan untuk mendapatkan kunci privat yang diinginkan.



Gambar 5.5 Diagram perbandingan waktu pengujian *brute force*

Dari hasil diagram yang ditunjukan pada gambar 5.5 menghasilkan suatu nilai yang berbeda-beda untuk masing-masing metode pembangkitan bilangan prima.Namun dapat disimpulkan bahwa penggunaan metode dalam pembangkitan bilangan prima yang terbaik dari tiga metode yang dipilih untuk dilakukan pengujian adalah metode *Sieve Erastosthenes.* Ini ditunjukan oleh grafik yang Sedangankan metode pembangkitan bilangan prima *brute force* merupakan metode pembangkitan yang memerlukan waktu paling lama dalam mencari kunci privat menggunakan pencarian *brute force*.

Pada pengujian berikutnya adalah menguji kunci privat yang dipilih dan dengan waktu yang telah ditentukan sebelumnya apakah kunci tersebut dapat dipecahkan dalam rentan waktu tersebut atau tidak. Ini digunakan untuk mengetahui kekuatan kunci privat dengan panjang tertentu yang nantinya dapat digunakan sebagai rujukan atau saran dalam melakukan penelitan lebih lanjut.

Tabel 5.3 Hasil pengujian kunci privat terhadap batas waktu yang ditentukan

Privat Key	Batas waktu (menit)				
	2	4	6	8	10
2135933839	0	1	1	1	1
2966986799	0	1	1	1	1
4100221979	0	1	1	1	1
5327721089	0	0	1	1	1
7696516365	0	0	0	1	1
507600703	0	0	0	1	1
7191174385	0	0	0	0	1
7997226411	0	0	0	0	0

Keterangan :

1 = Berhasil ditemukan.

2 = Gagal ditemukan.

Berdasarkan tabel di atas disimpulkan bahwa terdapat 0% kunci yang dapat ditemukan dibawa 2 menit, sedangkan di bawah 4 menit terdapat 37.5%. Lalu pada waktu kurang dari 6 menit terdapat 50% kunci yang dapat ditemukan dari delapan kunci yang dicoba. Sedangkan pada waktu dibawah 8 menit prosentase kunci yang dapat ditemukan adalah 75% dan dibawah 10 menit adalah 87.5%.

BAB VI
PENUTUP

6.1 Kesimpulan

Dari hasil uji dan analisis yang telah dilakukan dapat diambil kesimpulan sebagai berikut :

1. Dihasilkan suatu aplikasi kriptografi yang dapat menerapkan algoritma RSA dalam proses enkripsi dan dekripsi *file* dokumen *Microsoft Office* yang dapat digunakan untuk melakukan proses kriptografi terhadap *file .doc, .xls* dan *.ppt.* Aplikasi kriptografi melakukan proses enkripsi *file* dan dekripsi *file* dengan cara mengambil bytes-bytes dari *file* yang akan dienkripsi dan dekripsi. Sebelum melakukan proses enkripsi dan dekripsi, proses pertama adalah membangkitkan kunci publik untuk enkripsi dan kunci privat untuk dekripsi dengan menggunakan bilangan-bilangan prima. Kumpulan *bytes* tersebut merupakan suatu *plaintext* yang akan dirubah ke dalam bentuk *chipertext* menggunakan proses enkripsi menggunakan kunci publik. Selanjutnya untuk membuka *file,* digunakan proses dekripsi. Proses ini merubah *chipertext* ke dalam bentuk *plaintext* menggunakan kunci privat. Setelah semua proses dekripsi selesai, *file* tersebut dapat dibuka kembali.
2. Pada hasil pengujian *brute force attack* terhadap kunci privat yang terbentuk pada saat proses enkripsi, dapat disimpulkan bahwa panjang *n* dapat mempengaruhi jumlah iterasi yang diperlukan untuk mendapatkan kunci privat yang diinginkan. Selain mempengaruhi jumlah iterasi, panjang *n* juga mempengaruhi waktu proses yang diperlukan untuk mendapatkan kunci privat. Sesuai dengan gambar 5.1 dapat dilihat garis iterasi semakin ke kanan semakin naik. Jadi bilangan prima yang dipilih juga mempengaruhi kekuatan kunci privat. Ini dikarenakan *n* merupakan hasil perkalian antara bilangan prima pertama atau *p* dengan bilangan prima yang kedua atau *q*.

6.2 Saran

Saran yang dapat diberikan berdasarkan hasil yang didapat dari penelitian ini yaitu :

1. Dalam proses pembangkitan bilangan prima yang digunakan diharapkan untuk kedepan dapat menghasilkan bilangan prima dengan jumlah digit yang lebih banyak, sehingga dapat membangkitkan kunci dengan panjang maksimal sesuai saran dari penemu RSA yang mengatakan bahwa panjang kunci disarankan mempunyai panjang lebih dari 200 digit.
2. Dari hasil penelitian ini diharapkan dapat dikembangkan lebih luas untuk proses enkripsi dan dekripsi *file* yang bermacam-macam tidak hanya *file* dokumen *Microsoft Office.*

DAFTAR PUSTAKA

[ALG-10] Al-Ghazali, M.R. 2010. *Sieve of Eratosthenes, Algoritma Bilangan Prima*. . Jurnal Informatika Program Studi Teknik Informatika ITB. Bandung.

[ARI-08] Aribowo, Eko. 2008. *Aplikasi Pengamanan Dokumen Office Dengan AlgoritmaKriptografi Kunci Asimetris Elgamal*. Jurnal Informatika Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Ahmad Dahlan. Yogyakarta.

[ARI-09] Arifin, Zainal. 2009. *Studi Kasus Penggunaan Algoritma RSA Sebagai Algoritma Kriptografi yang Aman*. Jurnal Informatika Program Studi Ilmu Komputer FMIPA Universitas Mulawarman. Samarinda.

[BUC-02] Buchmann, Johannes A. (2002). *Introduction to Cryptography*. New York: Springer-Verlag.

[HAR-09] Haro, Gok Asido. 2009. *Algoritma Pencarian Bilangan Prima*. Jurnal Informatika Program Studi Teknik Informatika ITB. Bandung.

[HID-03] Hidayat, Taufik. 2003 *Sistem Kriptografi IDEA*. Thesis S2 Teknik Informatika ITB. Bandung.

[LAB-00] Laboratories, RSA. 2000. *Frequently Asked Questions about Today's Cryptography, version 4.1*. New York : RSA Security, Inc.

[MEN-96] Menezes, Alfred J. 1996. *Handbook of Applied Cryptography*. CRC Press.

[MOR-11] Morgana. 2011. Dasar Kriptografi. http://koboyit.com/2011/03/dasar-kriptografi.html. Diakses tanggal 18November 2012.

[MUN-04] Munir, Rinaldi. 2004. *Teori Bilangan*. Bandung : Informatika.

[MUN-04] Munir, Rinaldi. 2004. *Algortima RSA dan Elgamal*. Bandung : Informatika

[ORI-04] Ortiz, Andres. 2004. *SMS Transmision Using PDU Mode and 7 Bit Coding Scheme*. Departamento de Arquitectura y Tecnología de Computadores Universidad de Granada. Spanyol.

[PRA-10] Pramudita, Krisnaldi Eka. 2010. *Brute Force Attack dan Penerapannya padaPassword Cracking*. Program Studi Teknik Informatika Institut Teknologi Bandung, Bandung.

[RIY-08] Riyanto, M. Zaki, Ardhian, Ardi. 2008. *Kriptografi Kunci Publik : Sandi RSA*. Jurnal Kelompok Studi Sandi, Yogyakarta.

[SUL-12] Sulaiman, Nur Hadi. 2012. *Penyusupan Pesan Terenkripsi Pada File Audio MP3 Menggunakan Metode Least Significant Bit(LSB).* Program Studi Ilmu Komputer Universitas Brawijaya, Malang.

[STA-07] Stallings, William 2007. *Network Security Essentials*. New Jersey : Pearson Education, Inc.

[VIV-12] Vivanews, 2012. http://teknologi.news.viva.co.id/news/read/372115-menelusuri-sejarah-panjang-pesan-singkat-sms. Diakses tanggal 18 Desember 2012