2018

Implementasi Hybrid Crytosystem Algoritma Rprime RSA dan Algoritma RC4A dalam Pengamanan File Teks

Hanum, Nikmah

Universitas Sumatera Utara

IMPLEMENTASI *HYBRID CRYPTOSYSTEM* ALGORITMA RPRIME RSA DAN ALGORITMA RC4A DALAM PENGAMANAN FILE TEKS

SKRIPSI

NIKMAH HANUM
131401029

PROGRAM STUDI S1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2017

PERSETUJUAN

Judul	: IMPLEMENTASI HYBRID CYRPTOSYSTEM ALGORITMA RPRIME RSA DAN ALGORITMA RC4A DALAM PENGAMANAN FILE TEKS
Kategori	: SKRIPSI
Nama	: NIKMAH HANUM
NIM	: 131401029
Program Studi	: SARJANA (S-1) ILMU KOMPUTER
Departemen	: ILMU KOMPUTER
Fakultas	: FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Komisi Pembimbing :

Pembimbing 2	Pembimbing 1
Jos Timanta Tarigan, S. Kom., M. Sc.	Dr.PoltakSihombing,M.Kom
NIP. 198501262015041001	NIP. 196203171991031001

Diketahui/disetujui oleh
Program Studi S-1 Ilmu Komputer
Ketua,

Dr. Poltak Sihombing, M.Kom.
NIP. 196203171991031001

PERNYATAAN

IMPLEMENTASI HYBRID CRYPTOSYSTEM ALGORITMA
RPRIME RSA DAN ALGORITMA RC4A DALAM
PENGAMANAN FILE TEKS

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, Maret 2017

Nikmah Hanum
131401029

PENGHARGAAN

Alhamdulillah, Puji dan syukur kehadirat Allah SWT, yang hanya dengan rahmat dan izin-Nya penulis dapat menyelesaikan penyusunan skripsi ini, sebagai syarat untuk memperoleh gelar Sarjana Komputer, pada Program Studi S1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.

Pada kesempatan ini penulis juga ingin memberikan ucapan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu penulis dalam memotivasi pengerjaannya. Ucapan terima kasih penulis sampaikan kepada:

1. Bapak Prof. Dr. Runtung Sitepu, SH., MHum. selaku Rektor Universitas Sumatera Utara.
2. Bapak Prof. Dr. Opim Salim Sitompul, M.Sc sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Dr. Poltak Sihombing, M.Kom. selaku Ketua Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara dan Pembimbing I yang telah memberikan bimbingan dan dukungan kepada penulis.
4. Ibu Maya Silvi Lydia, B.Sc., M.Sc. selaku Sekretaris Program Studi S-1 Ilmu Komputer Universitas Sumatera Utara dan Pembanding I yang telah memberikan kritikan dan saran serta masukan yang berharga bagi penulis.
5. Bapak Jos Timanta Tarigan, S. Kom., M.Sc. Selaku Dosen Pembimbing II yang telah memberikan arahan dan dukungan dalam pengerjaan skripsi ini.
6. Ibu Dian Rachmawati, S.Si., M.Kom. selaku Dosen Pembanding I yang telah memberikan kritikan dan saran serta masukan yang berharga bagi penulis.
7. Bapak M. Andri Budiman, S.T., M.Comp.Sc., M.E.M yang senantiasa memotivasi dan menginspirasi penulis.
8. Seluruh tenaga pengajar dan pegawai di Fakultas Ilmu Komputer dan Teknologi Informasi USU, terkhususnya di Program Studi S-1 Ilmu Komputer.
9. Ayahanda tercinta Arman Syah Matondang, Ibunda tercinta Hosimah Hasibuan, Adek tercinta Fitri dan Aidil yang telah memberikan doa, dukungan, kasih sayang, serta

motivasi kepada penulis.

10. Keluarga di UKMI Al Khuwarizmi Fasilkom-TI USU yang telah berbagi pemikiran, motivasi, dan inspirasi kepada penulis.
11. Sahabat-sahabat ku tercinta teruntuk Nurul, Purnama, Adel, Intan, Yuni, Fauza, Sasti, Vivit, Futri, Yoga, Kak Hasbi, Bg Rio, Bg Furqon, Kak Lila, Kak Lala dan Kak Wynda, Muniroh, Kak dwi, Kak dijah, Widi, Desi yang selalu memberikan kritikan, motivasi, dan dorongan yang luar biasa kepada penulis dalam menyelesaikan skripsi ini.
12. Teman-teman kuliah Ilmu Komputer stambuk 2013, khususnya Kom B, yang telah berbagi motivasi, rasa kebersamaan, dan inspirasi kepada penulis.
13. Semua pihak yang terlibat langsung atau tidak langsung yang penulis tidak dapat tuliskan satu per satu.

Semoga Allah SWT melimpahkan berkah kepada semua pihak yang telah memberikan bantuan, perhatian, serta dukungan kepada penulis dalam menyelesaikan skripsi ini.

Medan, Maret 2017

Penulis

ABSTRAK

Informasi memerlukan tingkat keamanan dalam proses penyampaiannya. Salah satu cara untuk kemanan informasi tersebut adalah Kriptografi. *Hybrid cryptosystem* merupakan salah satu metode yang digunakan dalam kriptografi yang menggabungkan algoritma simetris dan asimetris. Algoritma RC4A merupakan algoritma simetris yang digunakan untuk mengamankan file teks. File teks yang digunakan berekstensi *.txt, *.doc. Algoritma asimetris digunakan untuk pengamanan kunci RC4A yaitu Algoritma Rprime RSA dari pesan yang dienkripsi. Hasil dari penelitian ini menunjukkan bahwa dengan menggunakan *hybrid cryptosystem* algoritma RC4A dan algoritma Rprime RSA dapat mengamankan file teks dan mengamankan kunci untuk keutuhan data. Waktu pemrosesan enkripsi kunci dari algoritma Rprime RSA lebih besar dibandingkan waktu yang digunakan untuk pemrosesan dekripsi kunci. Waktu tersebut juga berbanding lurus dengan panjang plainteks.

Kata kunci: Kriptografi, RC4A, Rprime RSA, File Teks, *Hybrid cryptosystem*

IMPLEMENTATION OF HYBRID CRYPTOGRAPHY ALGORITHM RPRIME RSA AND ALGORITHM RC4A IN SECURING TEXT FILE

ABSTRACT

Information requires a level of security in the process of delivery. One way for information security is cryptography. Hybrid cryptosystem is one method used in the cryptographic algorithm which combines symmetric and asymmetric algorithms. RC4A algorithm is a symmetric algorithm used to secure a text file. The text file used with extension *.txt, *.doc. Asymmetric algorithms are used for key security RC4A ie Rprime RSA algorithm of the encrypted message. The results of this study showed that using a hybrid cryptosystem RC4A algorithms and RSA algorithms Rprime can secure files and secure key text for the integrity of the data. Processing time Rprime key encryption algorithm RSA is greater than the time spent on processing the decryption key. This time is also proportional to the length of the plaintext.

Keywords: Cryptography, RC4A, Rprime RSA, Text Files, Hybrid cryptosystem

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Hampir seluruh kehidupan saat ini menggunakan teknologi informasi. Informasi berfungsi sebagai sarana untuk saling berintraksi. Salah satu cara bertukar informasi menggunakan internet. Pertukaran informasi tersebut memerlukan keamanan dalam proses penyampaiannya. Tingkat keamanan dalam informasi ini digunakan untuk menjaga kerahasian informasi dari penyadap atau orang yang ingin mencuri informasi tersebut. Oleh sebab itu, untuk menjamin kerahasiaan dari penyampaian diperlukan sistem keamanan informasi.

Ada beberapa cara dan teknik yang digunakan untuk menjaga keamanan informasi dari sebuah informasi yang dikirim. Salah satunya adalah kriptografi.

Kriptografi berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua Kriopto dan Graphia. Kripto berarti *secret* (rahasia) dan Graphia berarti Writing (tulisan). Menurut terminologinya Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ketempat yang lain (Ariyus, 2006). Pesan asli biasanya disebut juga sebagai *Plaintext*. Sedangkan pesan yang sudah diamankan disebut *Ciphertext*.

Algoritma Rprime RSA adalah algoritma pengembangan dari algoritma varian RSA. Dalam jurnal Ceaser memperkenalkan varian RSA dengan menggabungkan antara algoritma Mprime-RSA dan algoritma Rebalanced RSA. Gabungan dari kedua algoritma tersebut adalah Rprime RSA (Padhey, 2002).

Algoritma RC4A diusulkan pada tahun 2004 yang di rancang oleh Souradyuti Paul and Bart Preneel. Prinsip desain utama RC4A adalah untuk mengurangi korelasi antara *output byte* dan variabel internal dengan membuat *output byte* tergantung pada variabel yang lebih acak. Algoritma RC4A sangat mirip dengan algoritma RC4

(Karahan, 2015). Algoritma RC4A mempunyai kelebihan dua keluaran byte yang dihasilkan berturut-turut.

Berdasarkan penjelasan di atas, penulis akan membuat sistem untuk mengimplementasikan *hybrid cryptosystem* antara algoritma Rprime RSA dan algoritama RC4A dalam pengamanan file teks.

1.2. Rumusan Masalah

Berdasarkan masalah diatas, Rumusan masalah pada penelitian ini adalah bagaimana mengimplementasikan pengamanan file teks menggunakan metode *hybrid cryptosystem* antara algoritma simetris dan algoritma asimetris.

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah :

1. Teks inputan yang digunakan berupa file .txt dan .doc.
2. Pada algoritma Rprime RSA panjang bilangan prima 3 sampai 15 digit.
3. Pada algoritma Rprime RSA nilai kunci private pada p dan dp sebanyak 3.
4. Bahasa pemrograman yang digunakan yaitu C#.
5. Karakter yang digunakan pada *plaintext* dan *ciphertext* adalah *printable characters* ASCII (*American Standard for Information Interchange) printable character* (kode karakter 0-255).

1.4. Tujuan Penelitian

Tujuan penelitian ini untuk membuat aplikasi pengamanan file teks dengan mengkombinasikan antara algoritma Rprime RSA dan algoritma RC4A.

1.5. Manfaat Penelitian

Manfaat penelitian ini adalah sebagai berikut:

1. Penelitian ini dapat memperoleh satu aplikasi untuk pengamanan file teks.
2. Menambah pengetahuan bagi penulis tentang ilmu kriptografi.
3. Sebagai bahan referensi bagi peneliti lain yang berkaitan dengan topik ini.

1.6. Metodologi Penelitian

Langkah – langkah yang dilakukan dalam penelitian ini adalah :

1. Studi Literatur

Pada tahap ini peneliti dilakukan dengan pengumpulan referensi yang berkaitan dengan penelitian. Hal ini dilakukan untuk memperoleh informasi dan data yang diperlukan untuk penulisan skripsi ini. Referensi yang digunakan dalam penulisan ini berupa buku, jurnal, tesis, skripsi, artikel yang berkaitan dengan kriptografi, algoritma Rprime RSA, dan algoritma RC4A.

2. Analisis dan Perancangan

Dengan adanya rumusan masalah dan batasan masalah, kebutuhan perancangan dianalisis disertai dengan pembuatan *flowchart, UML, Design Interface.*

3. Implementasi

Pada tahap ini perancang mengimplementasikan dengan menggunakan bahasa C#.

4. Pengujian

Pada tahap ini prototipe sistem yang telah diimplementasikan dilakukan pengujian dengan file teks yang berekstensi .txt dan .doc.

5. Dokumentasi

Setelah implementasi, maka penulis akan membuat dokumentasi atau laporan tiap tahap dari program yang penulis rancang.

1.7. Sistematika Penulisan

Sistematika penulisan skripsi ini terdiri dari beberapa bagian utama, yaitu:

BAB I: PENDAHULUAN

Bab ini merupakan penjelasan mengenai latar belakang pemilihan topik penelitian "Implementasi *Hybrid cryptosystem* Algoritma Rprime RSA dan Algoritma RC4A dalam Pengamanan File Teks", rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian dan sistematika penulisan.

BAB 2: TINJAUAN PUSTAKA

Bab ini berisi teori – teori yang berkaitan dengan penelitian yang dilakukan penulis. Teori – teori tersebut antara lain : kriptografi, *hybrid*

cryptosystem, teori *pretty good privacy* (PGP), teori bilangan, kriptografi asimetris dan kriptografi simetris (khususnya Rprime RSA dan RC4A).

BAB 3: ANALISIS DAN PERANCANGAN

Bab ini berisi tentang analisis masalah penelitian dan perancangan sistem yang akan dibuat.

BAB 4: IMPLEMENTASI DAN PENGUJIAN

Bab ini berisikan pembahasan yang berkaitan dengan implementasi perangkat lunak yang telah dirancang berdasarkan data yang diperoleh. Kinerja sistem dilihat dari keberhasilan enkripsi-dekripsi.

BAB 5: KESIMPULAN DAN SARAN

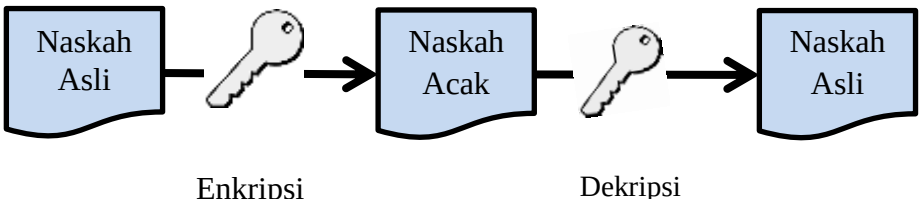
Bab ini berisi tentang kesimpulan secara umum dari uraian pada bab-bab sebelumnya, serta saran untuk pengembangan lebih lanjut dari penelitian ini.

BAB 2

TINJAUAN PUSTAKA

2.1. Kriptografi

Kriptografi adalah ilmu mengenai teknik enkripsi dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi (Kromodimoeljo, 2010). Metode untuk mengirim pesan secara rahasia sehingga hanya penerima pesan yang dituju yang dapat menghilangkan penyamaran dan membaca atau memahami isi pesan yang sebenarnya disebut kriptografi (Misfar, 2016). Dalam kriptografi pesan asli disebut *plaintext* dan pesan yang disamarkan disebut *ciphertext*. Siapapun yang terlibat dalam kriptografi disebut *cryptographer* (Mollin, 2007). Contoh dari proses enkripsi dan dekripsi dapat dilihat pada Gambar 2.1.

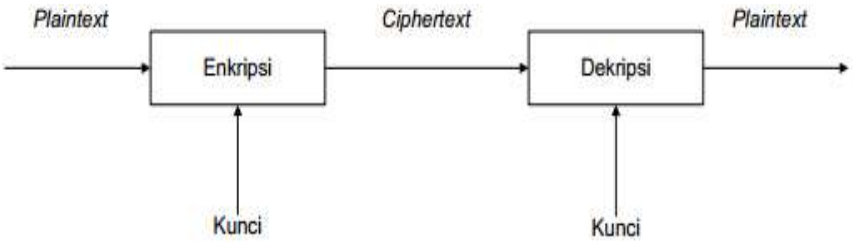


Gambar 2.1. Proses enkripsi dan dekripsi (Kromodimoeljo, 2009)

2.2. Jenis – jenis Kriptografi

2.2.1. Algoritma Simetris

Algoritma simetris adalah algoritma yang menggunakan kunci enkripsi yang sama dengan kunci dekripsinya (Widarma, 2016). Jika mengirim pesan dengan algoritma ini antara pengirim dan penerima harus sama-sama mengetahui kuncinya. Oleh karena itu pengirim harus memastikan bahwa jalur yang dikirim aman. Pada Gambar 2.2. merupakan diagram algoritma simetris, dimana pada saat proses enkripsi dan dekripsi mempunyai kunci yang sama.

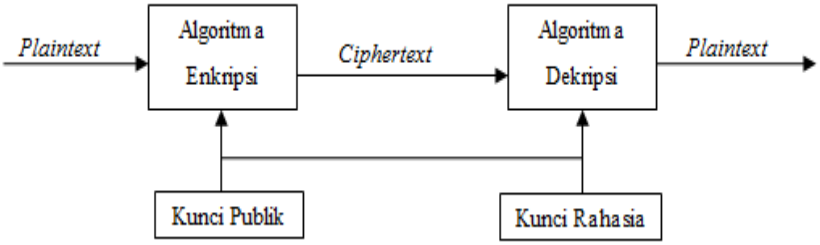


Gambar 2.2. Diagram Algoritma Simetris (Sujiono, 2016)

Adapun yang termasuk algoritma simetris adalah RC4, RC4A, Hill Cipher, Affine Cipher, OTP, dan lain-lainnya.

2.2.2. Algoritma Asimetris

Algoritma Asimetris sering juga disebut algoritma kunci publik, artinya algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya (Widarma, 2016). Adapun diagram dari algoritma asimetris dapat dilihat pada Gambar 2.3.



Gambar 2.3. Diagram Algoritma Asimetris (Masya, 2016)

Pada gambar 2.3. terlihat bahwa algoritma asimetris pada enkripsi menggunakan kunci pablik dan dekripsi kunci rahasia. Adapun algoritma yang termasuk dalam algoritma asimetris ini adalah RSA, ElGamal, dan lain-lainnya.

2.3. Hybrid cryptosystem

Hybrid cryptosystem adalah mengkombinasikan antara algoritma asimetris dengan algoritma simetris. Pada *hybrid cryptosistem* ini mendapat keuntungan dari algoritma asimetris dan algoritma simetris yaitu kelemahan dari tiap algoritma asimetris dan algoritma simetris akan saling menutupi. Berikut ini langakah-langkah umum proses enkripsi dan dekripsi pada *hybrid crptosystem* (Scheiner, 1996):

1. Bob mengirimkan Alice sebuah kunci publik

2. Alice membuat kunci simetris dan mengenkripsi *plaintext* menggunakan kunci tersebut
3. Alice mengenkripsi kunci simetris menggunakan kunci publik pemberian Bob, dan mengirimkan *ciphertext* utama dan *ciphertext* kunci kepada Bob.
4. Bob mendekripsikan *ciphertext* kunci yang di peroleh dari Alice menggunakan kunci pribadinya untuk memulihkan kunci simetris.
5. Bob mendekripsikan *ciphertext* utama menggunakan kunci simetris untuk memperoleh *plaintext*.

2.4. Teori *Pretty Good Privacy* (PGP)

Pretty Good Privacy (PGP) merupakan salah satu implementasi dari metode enkrpsi dalam sebuah program. PGP dikembangkan oleh Phil Zimmerman pada tahun 1991. Pengamanan *email, file*, dan program dapat menggunakan PGP. Oleh karena itu, PGP juga disebut dengan enkripsi hibrida karena PGP memanfaatkan dua tingkatan kunci, yaitu kunci rahasia (simetris) – yang disebut juga *session key* – untuk enkripsi data dan pasangan kunci privat – kunci umum untuk pemberian tanda tangan digital serta melindungi kunci simetris (Ariyus, 2008). Pada dasarnya, PGP dapat melakukan lima operasi utama, yaitu otentikasi, kerahasiaan, kompresi, kompatibilitas *e-mail* dan segmentasi.

2.5. Teori Bilangan

2.5.1. Bilangan Prima

Bilangan prima adalah bilangan yang habis di bagi satu dan dirinya sendiri. Sebuah bilangan integer a>1 disebut bilangan prima jika dan hanya jika pembaginya hanya ±1 dan ±a selain itu disebut bilangan komposit (Sadikin, 2012). Contoh bilangan prima adalah 3 karena habis dibagi 1 dan 3 dan contoh bilangan komposit adalah 4.

2.5.2. Greatest Common Divisor (GCD)

Salah satu cara untuk mendapatkan GCD(a, b) adalah dengan membuat daftar semua faktor dari a dan membuat daftar semua faktor dari b (Mizfar, 2016). Jika a, b ϵ Z keduanya tidak nol, maka gcd dari a dan b adalah g sehingga g|a, g|b, dan g habis dibagi oleh a dan b, dan di lambangkan dengan g = gcd (a,b) (Rosen, 2007). Contoh: gcd (36,88) = ?

88 mod 36 = 16

36 mod 16 = 4

16 mod 4 = 0

Jadi, gcd (36,88) adalah 4.

2.5.3. Relatif Prima

Jika a, b ϵ Z, dan gcd (a,b) = 1, maka a dan b dikatakan relative prima (Rosen, 2007). Contoh bilangan yang relatif prima adalah gcd (11,7) =...?

11 mod 7 = 4

7 mod 4 = 1

2.5.4. Aritmatika Modulo

Aritmatika modular digunakan agar transformasi penyandian selalu bernilai {0,...,25} sehingga memiliki pasangan simbol yang digunakan (Sadikin, 2012). Domain dari aritmatika modular adalah {0,1,2,...,n-1}, dimana n adalah besarnya domain. Aritmatika disebut aritmatika modulo n, dengan penambahan dan perkalian seperti aritmatika biasa jika menghasilkan bilangan yang termasuk dalam domain (Kromodimoeljo, 2010). Notasi operasi mod ditulis:

$$a \bmod n = r \qquad (1)$$

sebagai contoh 32 mod 7 = 4 dengan sisa 4, 32 = (7×4) + 4.

2.5.5. Modulo Eksponensial

Permasalahan pada operasi modulo adalah bagaimana menghitung x^y (mod n) dengan y yang sangat besar sehingga dapat diperoleh hasil yang benar, cepat tanpa kesalahan komputasi dan overflow (Nasution, Nurhaliza 2016).

2.5.6. Invers Modulo

Suatu bilangan a mempunyai invers modulo n jika dan hanya jika ($\leftrightarrow$) gcd (a,n) = 1 (Kromodimoeljo, 2010). Notasi yang digunakan untuk mempresentasikan invers perkalian bilangan a adalah a^{-1} (Sadikin, 2012). Pada perkalian invers modulo n ϵ Z_m nilai invers gcd(m,n) = 1.

Contoh invers modulo dari 5 mod 12, penyelesaian dapat dilihat pada Tabel 2.1.

Tabel 2.1. Penyelesaian Invers Modulo

m^{-1}	m^{-1}. m mod n
1	5
2	10
3	3
4	8
5	1

2.5.7. Euler Totient Function (ϕ)

Untuk semua n ϵ N fungsi *ϕ-Euler*, ϕ (n) adalah jumlah m ϵ N sehingga m< n dan gcd (m,n)=1 (Mollin, 2007). Beberapa hasil fungsi ϕ (n) adalah (Sadikin, 2012) :

1. ϕ (1) =0
2. ϕ (p) = p-1 jika p adalah bilangan prima. Sebab semua elemen {1,...,p1} merupakan relatif prima dengan p
3. ϕ (m× n) = ϕ (m) × ϕ (n) jika m merupakan relatif prima dengan n
4. ϕ (p^e) = p^e – p^{e-1} dengan p adalah bilangan prima

Contoh dari ϕ (91), ϕ (15), ϕ (64)

1. ϕ (91) = 91-1 = 90, sebab 91 merupakan bilangan prima
2. ϕ (15) = ϕ (5 × 3) = ϕ (5) × ϕ (3) = 4 × 2 = 8
3. ϕ (64) = ϕ (2^6) = 2^6 – 2^5 = 32

2.5.8. Ordo Modulo

Jika m ϵ Z, n ϵ N dan GCD (m,n) =1. Maka ordo dari m mod n adalah nilai e terkecil dalam e ϵ N yang memenuhi $m^e \equiv 1 \pmod n$. Ordo m mod n dinotasikan dengan e = ord_n (m) (Mollin, 2007).

2.5.9. Akar primitif

Jika m ϵ Z, n ϵ N dan ord_n (m) = ϕ (n), maka m merupakan akar primitif modulo n (Mollin, 2007).

2.5.10. Teorema Fermat

Algoritma pengujian bilangan prima yang banyak digunakan dalam sistem kriptografi adalah algoritma fermat.. Jika p merupakan sebuah bilangan prima dan a adalah bilangan integer positif yang tidak habis di bagi p , maka :

$$a^{p-1} = 1 \bmod p \qquad (2)$$

Sebaliknya jika p adalah sebuah bilangan komposit dan gcd(a,p) = 1, maka :

$$a^{p-1} \neq 1 \bmod p \qquad (3)$$

Contoh 8^{12} mod 13.

P= 13, dimana 13 merupakan bilangan prima dan 8 tidak habis dibagi 13 dan 12 = p – 1. Jadi teorema fermat berlaku sehingga 8^{12} mod 13 = 1.

2.6 Algoritma Rprime RSA

Algoritma Rprime RSA merupakan bagian dari algoritma RSA. Pada tahun 2002 Cesar Alison menggabungkan varian RSA. Ide ini mengkombinasikan dua varian RSA yaitu Rebalanced RSA and Mprime RSA untuk mempertinggi kecepatan dekripsi (Verma & Garg, 2011). Umumnya ide skema ini menggunakan algoritma pembangkit kunci dari Rebalanced RSA (dimodifikasi bilangan prima untuk k) bersama dengan algoritma dekripsi Mprime RSA (Verma& Garg, 2011).

2.6.1. Metode Kunci Rprime RSA

Metode kunci Rprime RSA : ambil $s \leq n/k$, integer dan lakukan langkah-langkah sebagai berikut (Verma & Garg, 2015):

1. Menghasilkan k bilangan prima random dari n/k bits $p_1, p_2, ..., p_k$ dengan gcd(p_1-1, p_2-1, ..., p_k-1) =1 dan hitung N = $p_1, p_2, ..., p_k$.
2. Menghasilkan k bilangan acak dari s bits dp1,dp2,....d_{pk}, seperti: gcd(dp_k, p_k-1) =1 and $d_{p1} = d_{p2} = ...,d_{pk}$ mod 2.
3. Cari d seperti : d=d_{p1} mod (p_1-1), d=d_{p2} mod (p_2-1),....., d = d_{pk} mod (p_k-1).
4. Hitung e = d^{-1} mod φ (N).

Kunci publik = (N, e).

Kunci privat = (p_1, p_2,..., p_k, dp_1, dp_2,..., dp_k).

2.6.2. Metode Enkripsi

Metode enkripsi sama dengan algoritma MultiPrime RSA (Verma & Garg, 2009):

1. Dapatkan kunci publik recipient (N,e)

2. Pesan *plaintext* merupakan bilangan bulat positif M
3. Hitung *ciphertext* $C = M^e \bmod N$.
4. Kirim *ciphertext* C ke recipient

2.6.3. Metode Dekripsi

Metode dekripsi sama dengan algoritma MultiPrime RSA (Verma&Garg, 2011):

1. Hitung $M_i = C^{dpi} \bmod p_i$ untuk i, $1 \leq i \leq k$.
2. Terapkan CRT ke M_i untuk mengambil $M = C^{dpi} \bmod p_i$.

2.7. Algoritma RC4A

Pada FSE 2004, RC4A di usulkan oleh Souradyuti Paul dan Bart Preneel (Noman, et al. 2009). *Cipher* ini merupakan modifikasi dari RC4 untuk meningkatkan keamanan tanpa mengurangi efisiensi. Stream cipher RC4A bekerja pada dua fase, fase KSA (Key Scheduling Algorithm) dan fase PRGA (Pseudo Random number Generation Algorithm) (Noman, et al. 2008).

Tujuan RC4A adalah untuk meningkatkan keamanan terutama dengan meningkatkan kompleksitas dalam algoritma (Noman, et al. 2008). RC4A dibuat melalui perbaikan dari RC4, yaitu memberikan array 2 S (S1 dan S2) yang independent satu sama lain, sehingga RC4 seharusnya tidak memiliki bias(prasangka) dalam keluaran byte berurutan (Noman, et al. 2009). RC4A menggunakan tiga counter yaitu i, j1, j2. Variabel j1 dan j2 di perkenalkan kepada S1 dan S2. Lebih spesifik, dalam KSA untuk RC4, array S1 diinisialisasi, menggunakan kunci rahasia K. WK , yang di hasilkan dari array S1 dari PRGA pada RC4. Kemudian, S2 diinisialisasikan dalam KSA pada RC4, menggunakan WK. Tidak seperti RC4, dalam PRGA pada RC4A dua keluaran byte yang dihasilkan berturut-turut (Noman, et al.2009).

2.7.1. Langkah-langkah Aloritma RC4A

Pseudocode Algortima KSA (Noman, et al.2009) :

- **KSA (K)**

```
RC4_KSA(K,S1)
For i= 0...1-1
WK[i] = RC4_PRGA(S1)
RC4_KSA(WK, S2)
```

Pseudocode Algortima PRGA (Noman, et al.2009) :

- **PRGA**

Inisialisasi :

I=0

J1=j2=0

Generation loop:

i = i+1

j1 =j1+s1[i]

Swap(S1[i],S1[j1])

Output z1=S2[S1[i] + S1[j1]]

j2 = j2 +S2[i]

Swap(S2[i],S2[j2])

Output z2= S1[S2[i]+S2[j2]

2.8. Penelirian Relevan

Berikut ini beberapa penelitian tentang kriptografi yang terkait dengan metode algoritma Rprime RSA dan algoritma RC4A :

1. Fauziah, Y (2008) dalam penelitian membuat pengamanan pesan menggunakan *hybrid cryptosystem* dengan algoritma RC4 dan RSA. Hasil penelitan mengungkapkan bahwa keamanan bahwa keamanan data diantaranya berada pada sisi kunci, semakin besar/panjang kunci semakin lama waktu yang dibutuhkan untuk membongkarnya.
2. Nasution, N (2016) dalam judul implementasi *hybrid cryptosystem* algoritma RC4 dan Elgamal dalam pengamanan file text. Enkripsi file teks dengan menggunakan algoritma RC4 dapat melindungi teks yang terdapat dalam file tersebut.
3. Pranata, A (2016) dalam judul implementasi algoritma XOR dan algoritma RC4 pada aplikasi enkripsi dan dekripsi teks berbasis android. Penelitian ini menunjukkan bahwa waktu pemrosesan enkripsi dan dekripsi dari kombinasi algoritma XOR dan algoritma RC4 lebih efisien dibandingkan dengan penjumlahan waktu pada proses enkripsi maupun dekripsi masing-masing algoritma tersebut.

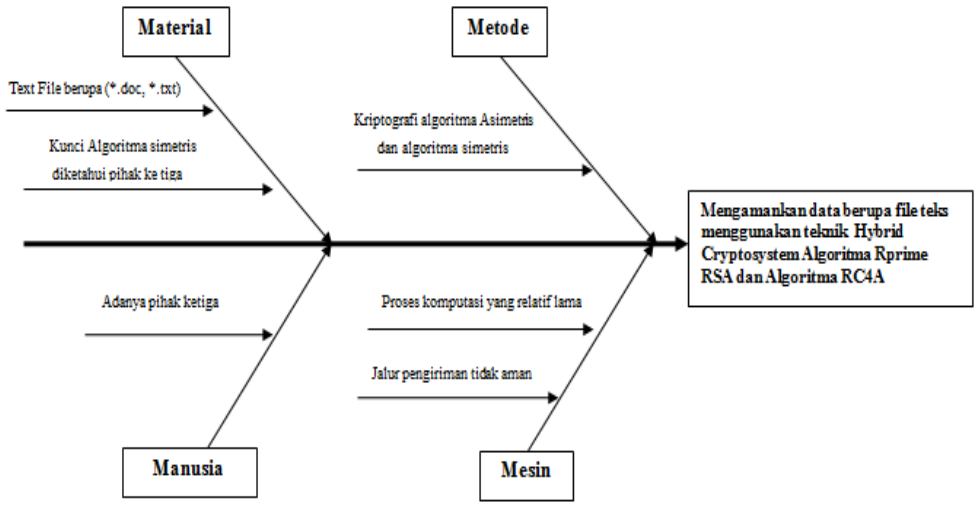
BAB 3
ANALISIS DAN PERANCANGAN

3.1. Analisis Sistem

Analisis sistem merupakan tahap awal yang dilakukan sebelum membuat sebuah sistem. Analisis sistem bertujuan untuk memecah sistem ke dalam komponen-komponen subsistem yang lebih kecil untuk mengetahui hubungan setiap komponen tersebut dalam mencapai tujuan (Whitten, 2007).

3.1.1. Analisis Masalah

Analisis masalah bertujuan untuk mengidentifikasi masalah dan memahami kelayakan masalah. Permasalahan yang akan dibahas adalah bagaimana menerapkan *hybrid cryptosystem* dalam pengamanan file yang akan dikirim. Masalah pada penelitian ini diidentifikasi dengan menggunakan diagram *Ishikawa.* Analisi masalah ini secara umum diilustrasikan pada gambar 3.1 yang dirancang dalam bentuk diagram *Ishikawa*.



Gambar 3.1 Diagram Ishikawa

Pada gamabar 3.1 dapat disimpulkan ada dua hal yang utama yaitu bagian permasalahan dan penyebabnya. Permasalahannya adalah merahasiakan kunci simetris dan mengamankan data teks, Sedangkan penyebabnya adalah metode yang

digunakan *hybrid cryptosystem* (menggabungakan algoritma asimetris (Rprime RSA) dan algoritma simetris (RC4A), Manusia sebagai pengirim dan penerima pesan yang menginginkan data yang dikirim aman dan menghindari pihak ketiga yang ingin mengambil informasi atau data dari pengirim atau penerima, material dan mesin.

3.1.2. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk menjelaskan fungsi-fungsi yang ditawarkan dan mampu dikerjakan sistem. Dalam analisis kebutuhan ini terdapat dua kebutuhan yang dapat dilakukan yaitu kebutuhan fungsional dan nonfungsional.

1. Kebutuhan Fungsional

   Kebutuhan fungsional mendeskripsikan fungsi-fungsi yang harus dulakukan oleh sebuah sistem untuk mencapai tujuan. Kebutuhan fungsional yang dibutuhkan oleh sistem adalah sebagai berikut:

a. Fungsi pembangkit kunci

   Pada pembangkit kunci, sistem memerlukan panjang bilangan prima yang ada pada algoritma Rprime RSA. Akan dilakukan beberapa perhitungan matematis untuk mendapatkan sepasang kunci algoritma Rprime RSA yaitu kunci privat dan kunci publik.

b. Fungsi enkripsi

   Pada proses enkripsi pesan, sistem membutuhkan algoritma RC4A untuk melakukan enkripsi yaitu dengan mengubah plainteks atau teks awal yang berupa file teks dengan format *.txt dan *.doc menjadi sebuah *Ciphertext*.

c. Fungsi dekripsi

   Pada proses dekripsi, sistem membutuhkan algoritma RC4A untuk mengubah kembali *Ciphertext* menjadi plainteks.

d. Fungsi penguncian kunci pesan

   Untuk fungsi penguncian kunci pesan, sistem membutuhkan proses enkripsi menggunakan kunci publik yang dihasilkan dari pembangkit kunci algoritma Rprime RSA sehingga menjadi sebuah *Cipherkey*. Dan proses dekripsi kunci untuk mendapatkan kunci asli menggunakan kunci privat yang juga dihasilkan dari algoritma Rprime RSA.

2. Kebutuhan Nonfungsional

Kebutuhan nonfungsional mendeskripsikan fitur lain seperti karakteristik, batasan sistem, performa, dokumentasi dan yang lainnya agar sistem berjalan sukses (Whitten, 2007). Kebutuhan nonfungsional yang dibutuhkan sistem adalah sebagai berikut :

1. Performa

   Sistem dapat mengenkripsi file teks dengan memerlukan waktu yang singkat dan mendekripsi ciphertext kembali sesuai keadaaan aslinya.

2. User friendly

   Sistem yang dibangun memiliki tampilan yang sederhana dan mudah digunakan

3. Kontrol

   Sistem yang dibangun memiliki pesan error jika pengguna untuk inputan yang tidak sesuai.

4. Dokumentasi

   Perangkat lunak yang dibangun memiliki panduan penggunaan, dan dapat menyimpan hasil enkripsi.

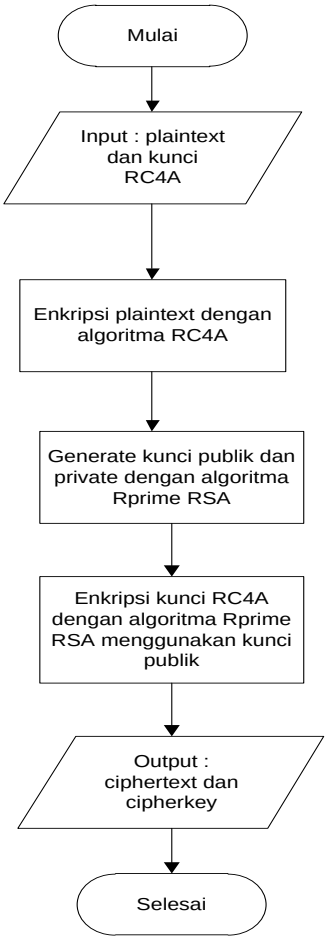
3.1.3. Analisis Pemodelan Sistem

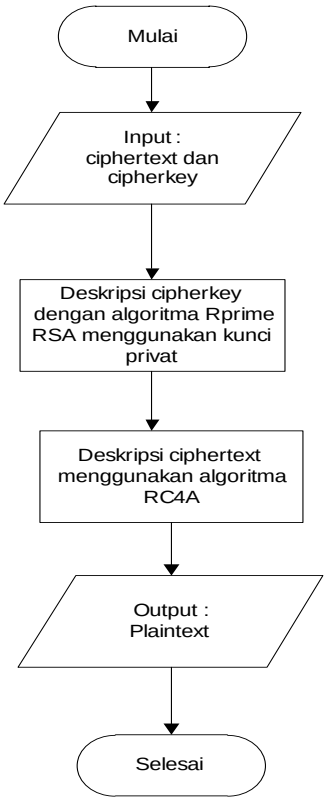
Pemodelan sistem bertujuan untuk menggambarkan kebutuhan dari sebuah perangkat lunak. Pemodelan sistem ditampilkan dengan menggunakan flowchart, use case diagram, sequence diagram, dan activity diagram.

1. Flowchart

Flowchart merupakan diagram alir dari bagan-bagan tertentu yang memiliki arus penggambaran mengenai langkah-langkah penyelesaian suatu permasalahan. Selain itu, flowchart juga memiliki fungsi memudahkan proses pengecekan terhadap sistem yang akan dibuat.

Berikut ini merupakan flowchart *hybrid cryptosystem* yang di bangun dapat dilihat pada Gambar 3.2 :

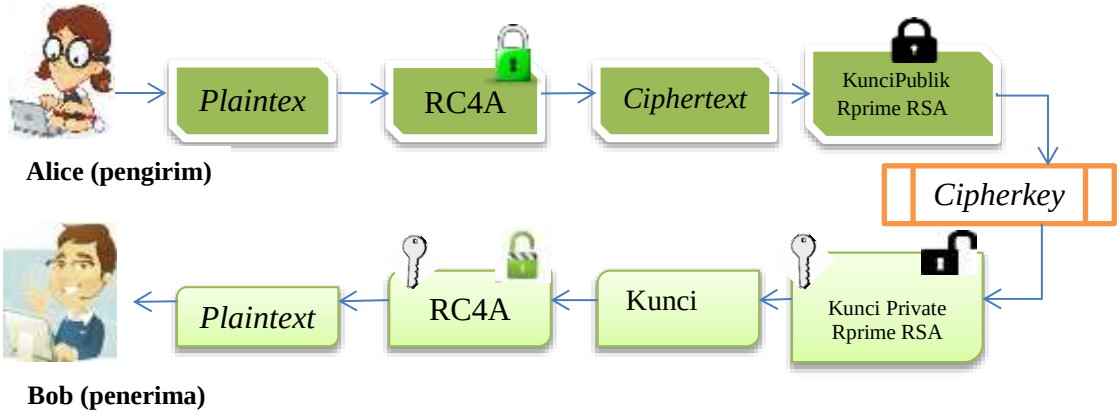




Flowchart enkripsi file dan kunci Flowchart dekripsi ciphertext dan *Cipherkey*

Gambar 3.2 Flowchart *Hybrid cryptosystem*

Berikut ini merupakan ilustrasi pengirim dan penerima pesan dapat dilihat pada Gambar 3.3 :

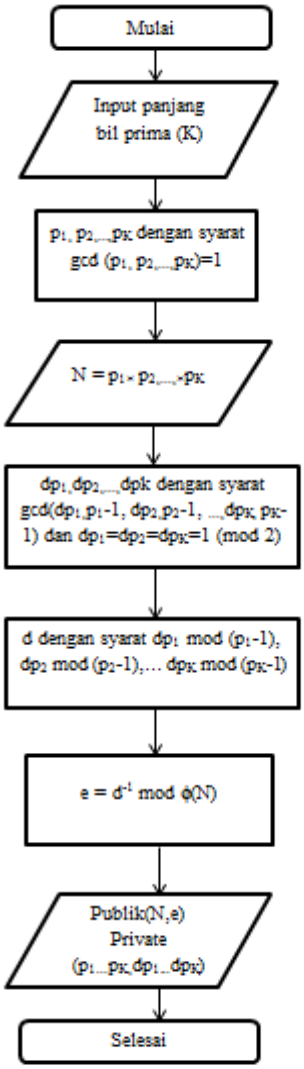


Gambar 3.3 Ilustrasi pengirim dan penerima pesan

Keterangan pada Gambar 3.3 Ilustrasi pengirim dan penerima pesan:

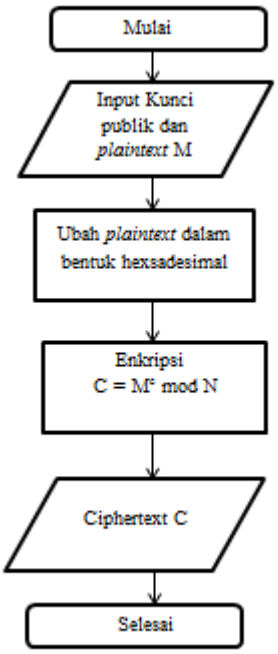
Alice mempunyai pesan yang ingin dirahasiakan, hal pertama kali yang harus dilakukan alice menerima kunci publik dari si bob. Selanjutnya pesan tersebut di enkripsi dengan algoritma RC4A hasilnya *plaintext* dan kunci RC4A dienkripsi menggunakan kunci publik algoritma Rprime RSA hasilnya *Cipherkey*. Kemudian *plaintext* dan *Cipherkey* tersebut dikirimkan ke si bob. Bob mendekripsi *Cipherkey* dengan kunci private algoritma Rprime RSA hasilnya kunci dan bob mendekripsi kunci dengan *plaintext* akhirnya bob bisa mengetahui pesan yang dikirim oleh si alice.

Berikut ini merupakan flowchart algoritma pembangkit kunci Rprime RSA dapat dilihat pada Gambar 3.4 :



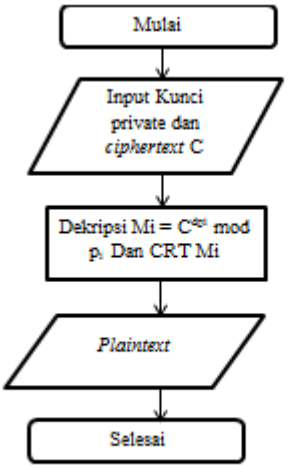
Gambar 3.4 Flowchart Pembangkit Kunci Algoritma Rprime RSA

Berikut ini merupakan flowchart algoritma enkripsi Algoritma Rprime RSA dapat dilihat pada Gambar 3.5 :



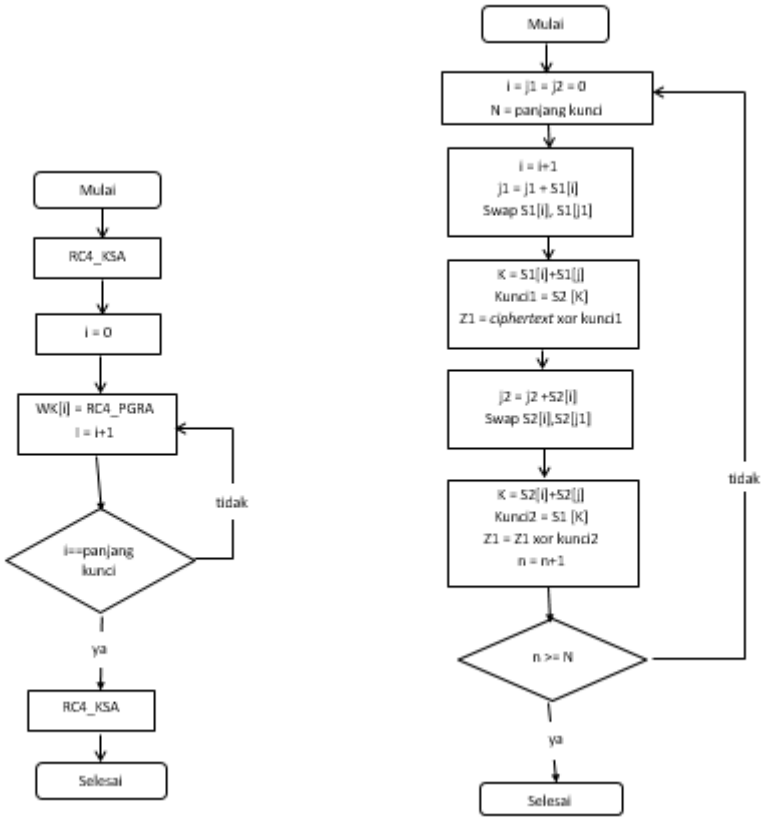
Gambar 3.5 Flowchart Enkripsi Algoritma Rprime RSA

Berikut ini merupakan flowchart algoritma dekripsi Algoritma Rprime RSA dapat dilihat pada Gambar 3.6 :



Gambar 3.6 Flowchart Dekripsi Algoritma Rprime RSA

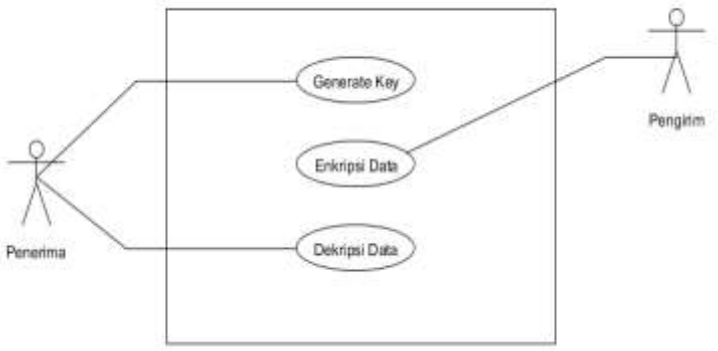
Berikut ini merupakan flowchart algoritma enkripsi dan dekripsi Algoritma RC4A dapat dilihat pada Gambar 3.7 :



Gambar 3.7 Flowchart Enkripsi dan Dekripsi Algoritma Rprime RSA

2. UseCase Diagram

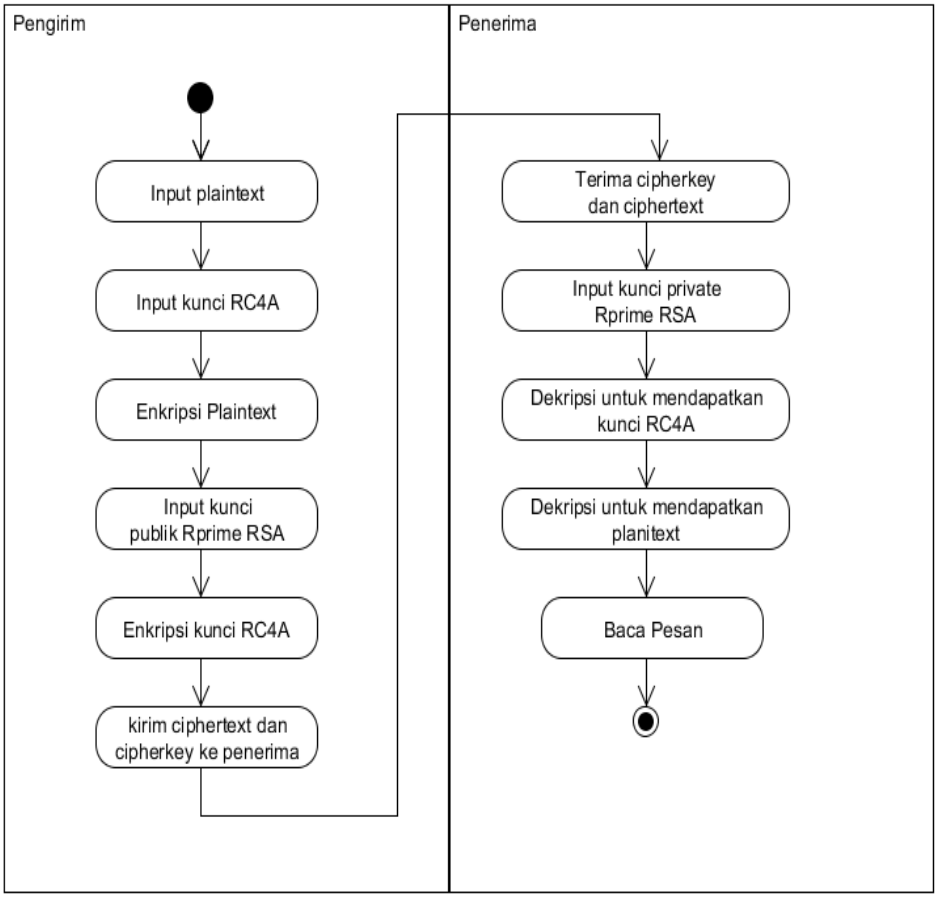
Usecase diagram adalah gambaran skenario penggunaan aplikasi sistem tentang bagaimana cara sistem bekerja dengan pengguna. Pengguna dalam usecase disebut actor. Hubungan antara actor dengan fungsi (usecase) yang tersedia pada sistem dihubungkan oleh garis lurus menyatakan bahwa aktor tersebut melakukan inisiasi permintaan terhadap suatu usecase. Diagram usecase pada penelitian ini dapat dilihat pada Gambar 3.8



Gambar 3.8 Diagram *Use-Case*

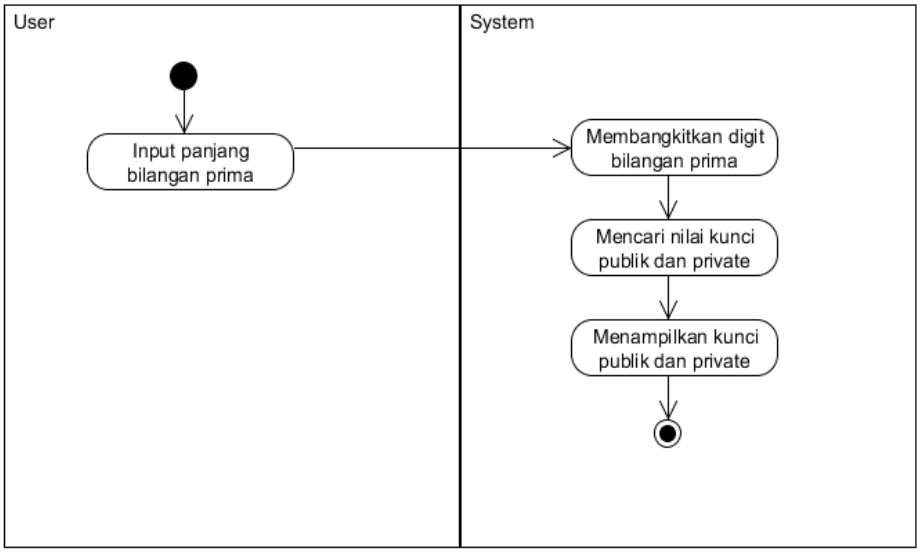
3. Activity Diagram

Activity diagram merupakan proses kerja dalam sebuah sistem yang sedang berjalan. Activity diagram juga bertujuan untuk membantu bagaimana memahami proses dan menggambarkan setiap intakasi yang ada antara beberapa usecase yang digunakan. Activity diagram dari sistem dapat dilihat pada Gambar 3.9



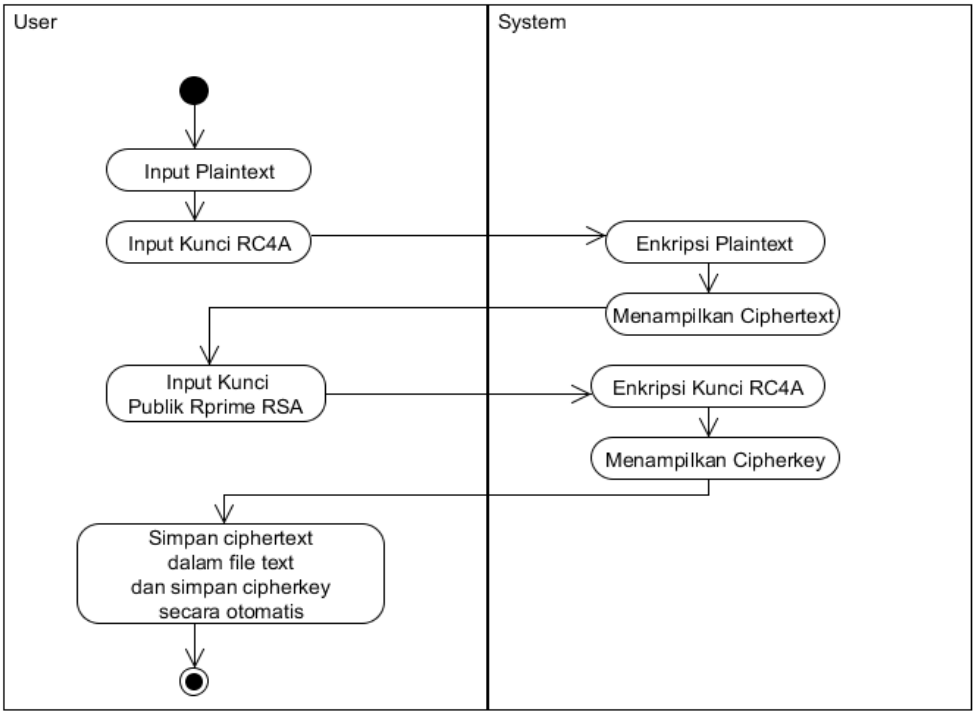
Gambar 3.9 Activity Diagram

Gambar 3.10 merupakan *activity diagram* untuk *use-case* pembangkitan kunci.



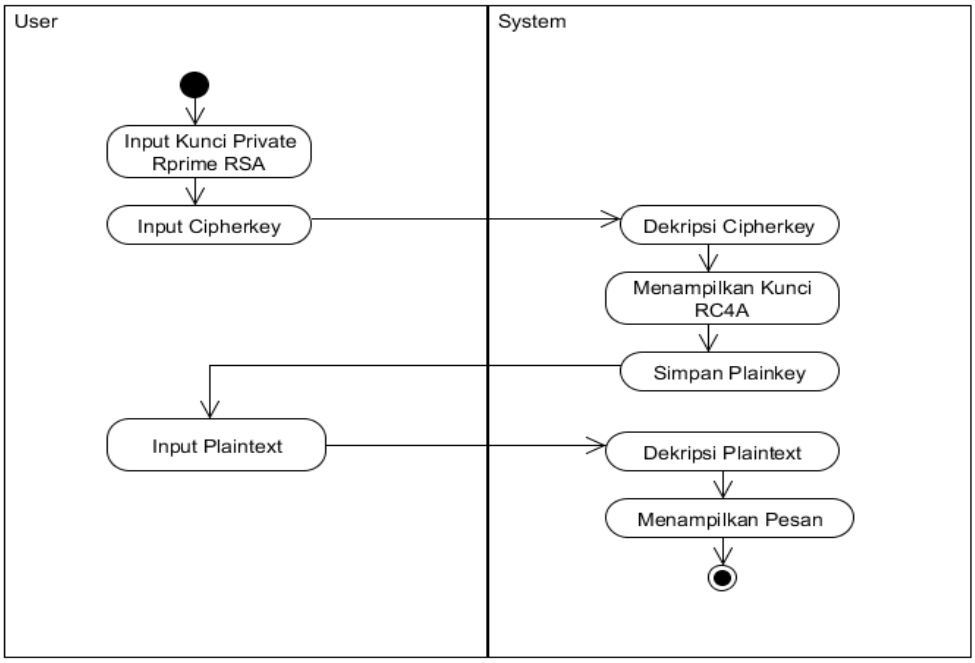
Gambar 3.10 Activity Diagram Pembangkit Kunci

Gambar 3.11 merupakan *activity diagram* untuk *use-case* dari enkripsi.



Gambar 3.11 Activity Diagram Enkripsi

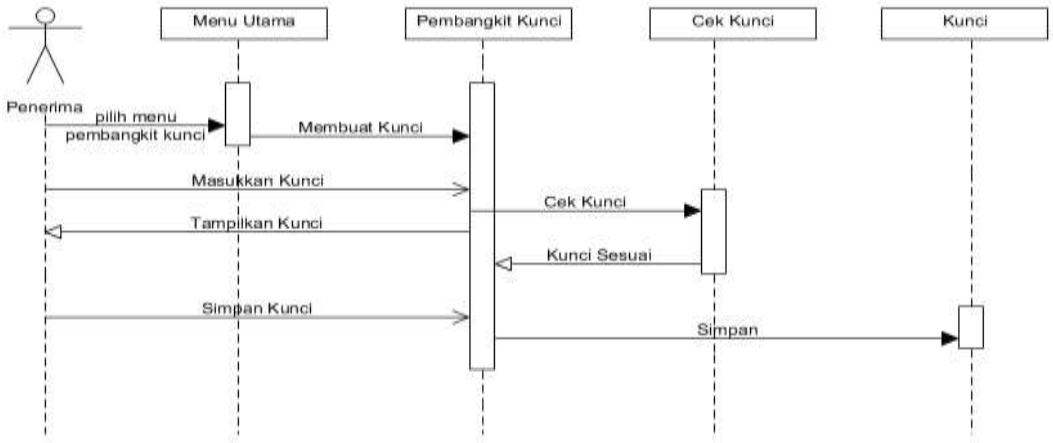
Gambar 3.12 merupakan *activity diagram* untuk *use-case* dari dekripsi.



Gambar 3.12 Activity Diagram Dekripsi

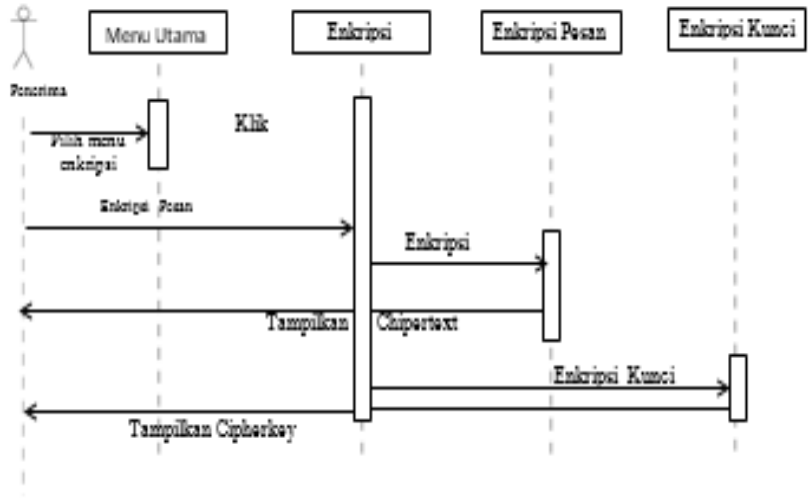
4. Sequence Diagram

Sequence diagram adalah menggambarkan rangkaian pesan yang akan dikirim antara objek yang ada serta intraksi yang terjadi antara objek. Sequence diagram membantu untuk menggambarkan data yang masuk dan keluar pada sistem. Sequence diagram yang sistem bangun pada pembangkit kunci dapat di lihat pada Gambar 3.13



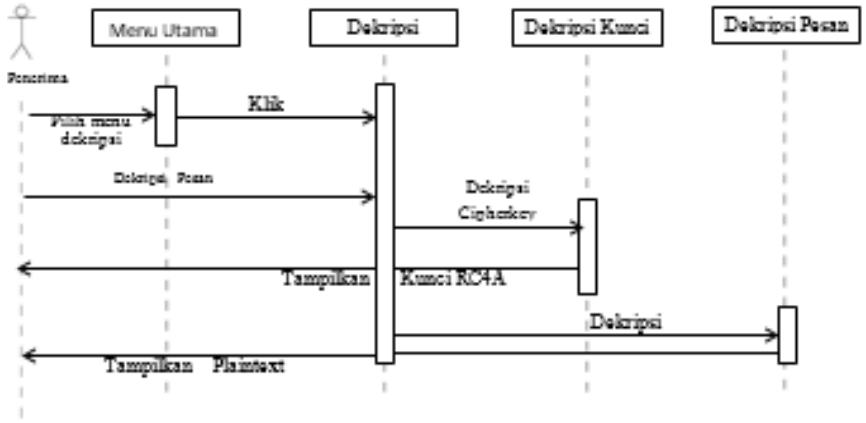
Gambar 3.13 Sequence Diagram untuk Pembangkit Kunci

Sequence diagram yang sistem bangun pada enkripsi dapat di lihat pada Gambar 3.14



Gambar 3.14 Sequence Diagram untuk Enkripsi

Sequence diagram yang sistem bangun pada dekripsi dapat di lihat pada Gambar 3.15.



Gambar 3.15. Sequence Diagram untuk Dekripsi

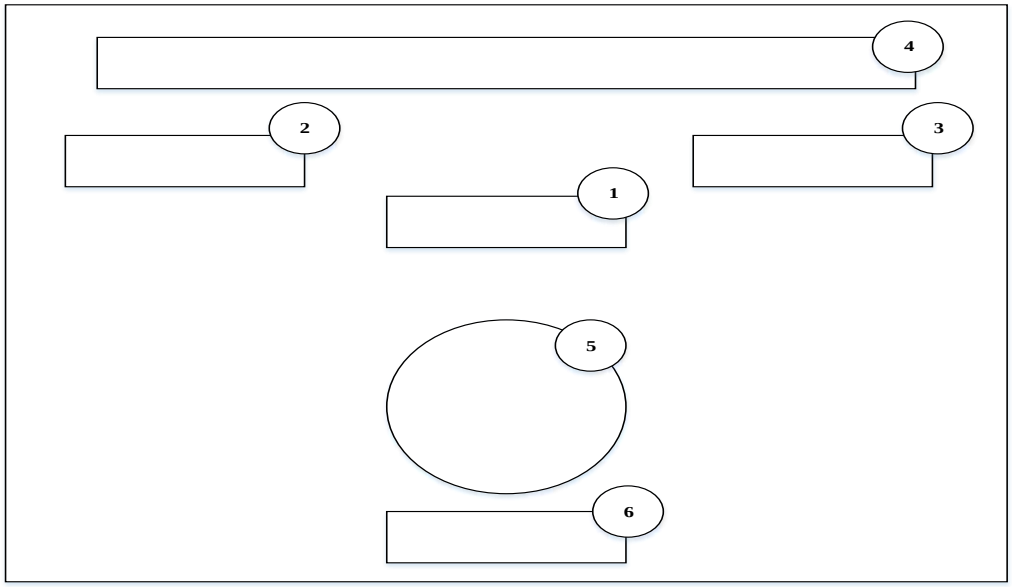
3.2 Perancangan Antarmuka (*Interface*) Sistem

Perancangan antarmuka atau *interface* merupakan desain awal tampilan dari sistem yang akan dibuat. Dalam perancangan antarmuka pada sistem harus memperhatikan faktor kenyamanan dan kemudahan bagi pengguna. Tujuan dari desain antarmuka

adalah membuat pengguna berinteraksi dan mencapai tujuannya dengan cara yang sesederhana dan seefisien mungkin.

3.2.1 Rancangan Form Utama Pada Sistem

Form utama (*home*) atau halaman depan merupakan tampilan awal ketika sistem dijalankan. Pada *form* ini terdapat tiga buah menu yang menampilan pembangkit kunci, enkripsi dan dekripsi. Gambar 3.16. menunjukkan rancangan tampilan sementara dari *form* utama.



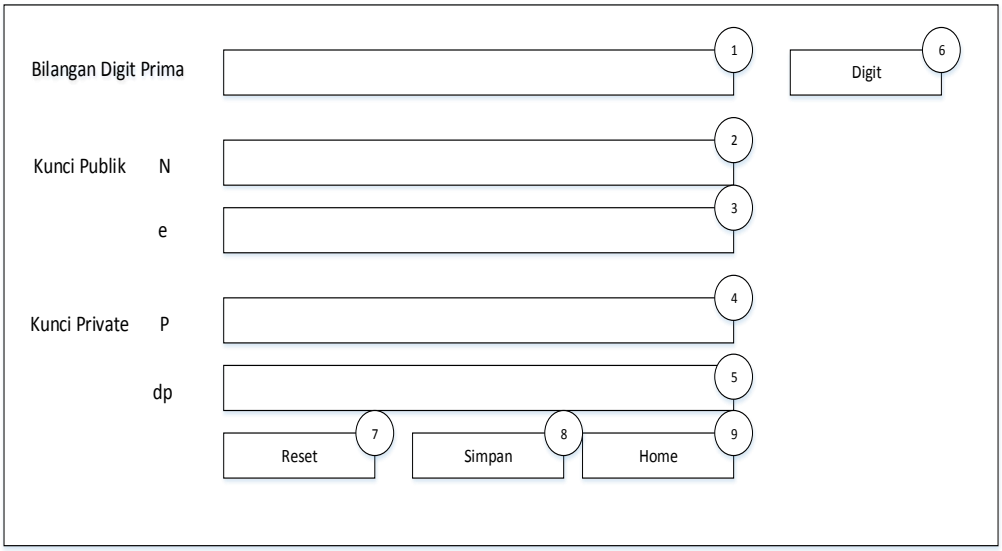
Gambar 3.16 Rancangan Form Utama pada Sistem

Keterangan:

1) Menu Pembangkitan Kunci, menu ini menampilkan *form* untuk melakukan proses pembangkitan kunci algoritma Rprime RSA.
2) Menu Enkripsi, yang berisi sub menu enkripsi pesan dan enkripsi kunci RC4A. Menu ini menampilkan dua *form* untuk proses enkripsi.
3) Menu Dekripsi, yang berisi sub menu dekripsi pesan dan dekripsi kunci RC4A. Menu ini menampilkan dua *form* untuk proses dekripsi.
4) *Label* yang menampilkan judul penelitian.
5) Gambar yang menampilakn logo Fakultas Ilmu Komputer dan Teknologi Informasi.
6) *Label* yang menampilkan nama dan NIM penulis.

3.2.2 Rancangan Form Pembangkitan Kunci Pada Sistem

Untuk menuju ke *form* pembangkit kunci, pengguna dapat memilih menu pembangkit kunci pada *menu button. Form* ini berfungsi untuk memudahkan pengguna dalam menentukan kunci, pengguna dapat memilih panjang digit prima kemudian akan muncul kunci publik dan kunci private. Kunci yang sudah diperoleh dapat disimpan untuk mempermudah proses enkripsi dan dekripsi selanjutnya. Gambar 3.17. menunjukkan rancangan tampilan sementara dari form pembangkit kunci.



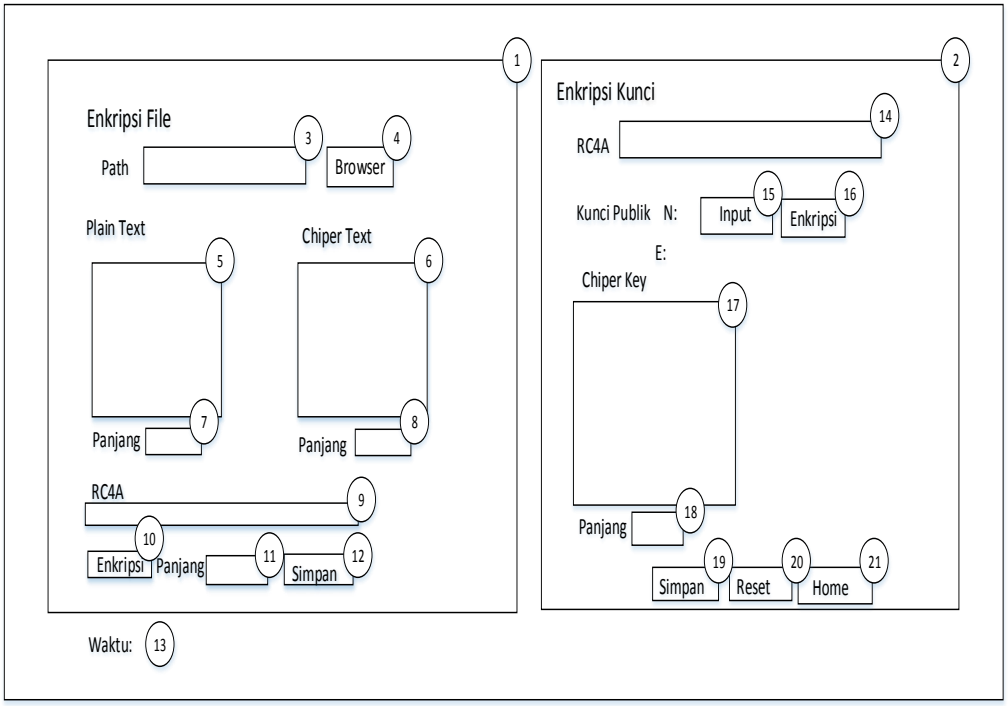
Gambar 3.17 Rancangan Form Pembangkit Kunci pada Sistem

Keterangan:

1) *Combobox* untuk memasukkan digit bilangan prima .
2) *Textbox* untuk menampilkan nilai N .
3) *Textbox* untuk menampilkan nilai e.
4) *Textbox* untuk menampilkan nilai p.
5) *Textbox* untuk menampilkan nilai dp.
6) Tombol hitung kunci untuk melakukan proses pembangkitan kunci.
7) Tombol reset untuk menghapus kunci.
8) Tombol simpan untuk menyimpan kunci.
9) Tombol home untuk kembali ke menu utama.

3.2.3. Rancangan Form Enkripsi Pada Sistem

Untuk mengkases *form* enkripsi, pengguna dapat memilih menu pengirim yang ada di *menu button*. Di *menu button* enkripsi, pengguna dapat memilih submenu untuk proses enkripsi pesan atau proses enkripsi kunci pesan. Pada *form* enkripsi pesan, pengguna dapat melakukan dua proses yaitu proses enkripsi pesan dengan algoritma RC4A dan proses enkripsi kunci pesan dengan algoritma Rprime RSA yang merupakan kunci publik. Gambar 3.18 menunjukkan rancangan tampilan sementara dari *form* enkripsi yaitu proses enkripsi file pesan dan proses enkripsi kunci pesan.



Gambar 3.18 Rancangan Form Enkripsi pada Sistem

Keterangan:

1) *Groupbox* untuk mengidentifikasi area enkripsi file dengan RC4A.
2) *Groupbox* untuk mengidentifikasi area enkripsi kunci RC4A dengan kunci publik Rprime RSA.
3) *TextBox* untuk menampilkan lokasi dan nama file.
4) Tombol Browser, untuk mencari *file* yang akan diinput.
5) *Textbox* untuk menampilkan isi *file*.
6) *Textbox* untuk menampilkan hasil enkripsi *Cipherkey*.
7) *Label* untuk menampilkan panjang karakter *plaintext.*
8) *Label* untuk menampilkan panjang karakter *ciphertext.*
9) *Textbox* untuk mengimput kunci RC4A.
10) Tombol untuk mengenkripsi file.

11) *Label* untuk menampilkan panjang karakter kunci RC4A.

12) Tombol untuk menyimpan *ciphertext.*

13) *Label* untuk menampilkan waktu enkripsi file dan enkripsi kunci.

14) *Label* untuk menampilkan kunci RC4A.

15) Tombol input untuk menampilkan N dan e sebagai kunci publik.

16) Tombol enkripsi untuk melakukan proses enkripsi.

17) *Textbox* untuk menampilkan kunci RC4A yang sudah dienkripsi (*Cipherkey*).

18) *Label* untuk menampilkan panjang karakter *Cipherkey.*

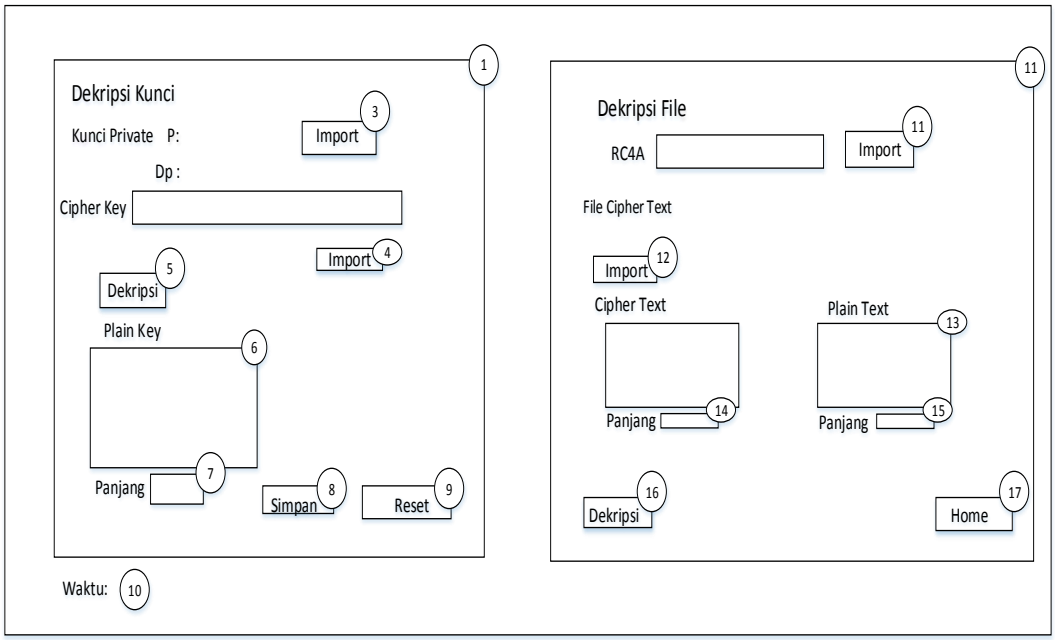
19) Tombol Simpan untuk menyimpan kunci RC4A.

20) Tombol *Reset* unuk menghapus kunci.

21) Tombol *Home* untuk kembali ke menu utama.

3.2.4 Rancangan Form Dekripsi Pada Sistem

Untuk mengakses *form* dekripsi, pengguna dapat memilih menu dekripsi pada button, menu ini berisi sub menu dekripsi kunci RC4A dan dekripsi pesan teks. Pada *form* ini ada dua proses, yang pertama yaitu proses dekripsi kunci pesan dan yang kedua proses dekripsi pesan *Ciphertext*. Gambar 3.19. menunjukkan rancangan tampilan sementara dari *form* dekripsi yaitu proses dekripsi kunci pesan dan proses dekripsi pesan.



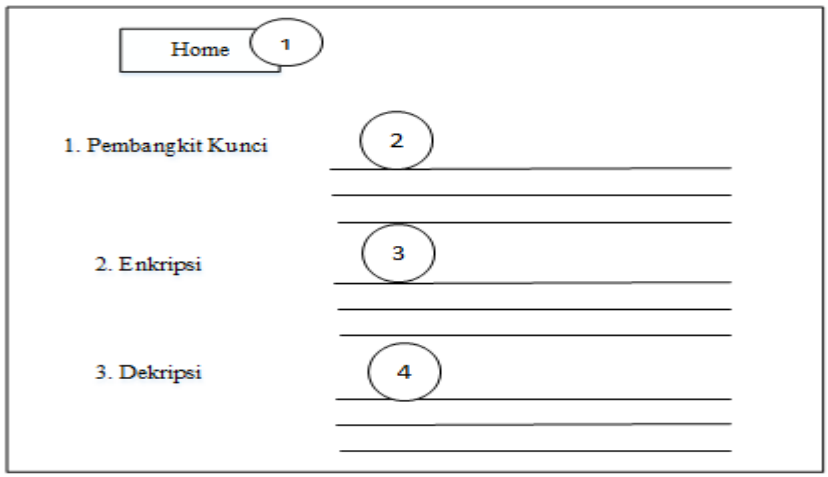
Gambar 3.19 Rancangan Form Dekripsi pada Sistem

Keterangan:

1) *Groupbox* untuk mengidentifikasi area dekripsi kunci RC4A dengan Rprime RSA.
2) *Groupbox* untuk mengidentifikasi area dekripsi file dengan RC4A.
3) Tombol Import Kunci, untuk menampilkan kunci privat Rprime RSA.
4) *Import* untuk menampilkan *Cipherkey*.
5) Tombol dekripsi untuk menampilkan *plainkey*.
6) *Texbox* untuk menampilkan hasil dekripsi.
7) *Label* untuk menampilkan panjang *plainkey*.
8) Tombol Simpan untuk menyimpan *plainkey*.
9) Tombol *Reset* untuk menghapus *plainkey* dan *plaintext*.
10) *Label* untuk menampilkan waktu dekripsi kunci dan dekripsi file.
11) Tombol *Import* untuk menampilkan *plainkey.*
12) Tombol *Import* untuk menampilkan *ciphertext.*
13) *Textbox* untuk menampilkan ciphertext.
14) *Label* untuk menampilkan panjang *ciphertext*.
15) *Label* untuk menampilkan panjang *plaintext*.
16) Tombol Dekripsi untuk mendekripsi file.
17) Tombol *home* untuk kembali ke menu utama.

3.2.5. Rancangan Form Bantuan

Gambar 3.20 merupakan tampilan bantuan. Di *form* bantuan, pengguna dapat melihat langkah-langkah bagaimana menggunakan fungsi pembangkit kunci, enkripsi, dan dekripsi



Gambar 3.20 Rancangan Form Bantuan

Keterangan:

1) Tombol “Home” untuk pindah ke halaman utama.
2) *Label* berisi panduan melakukan pembangkitan kunci.
3) *Label* berisi panduan melakukan enkripsi.
4) *Label* berisi panduan melakukan dekripsi.

3.3 Tahapan Cara Kerja Algoritma RC4A

Algoritma RC4A merupakan salah satu algoritma yang termasuk dalam algoritma kriptografi simetris. Berikut adalah contoh hasil enkripsi manual algoritma RC4A dengan mode 4 *byte* (untuk lebih menyederhanakan). *Plaintext* yang digunakan adalah “NH” dan kunci “12.

Proses KSA dengan masukan K dan S_1 adalah sebagai berikut :

1. Inisialisasi *array* S_1, S_1 = {0, 1, 2, 3}
2. Kunci yang digunakan adalah 1 dan 2, *l* = 2 *byte*. Ulangi kunci hingga memenuhi *array* K, K = {1, 2, 1, 2}
3. j = 0

Iterasi 1 (i = 0, j = 0, S_1 = {0, 1, 2, 3})

j = (j + S_1[i] + K[i mod *l*]) mod 4

j = (0 + S_1[0] + K[0 mod 2]) mod 4

j = (0 + 0 + 1) mod 4 = 1

Swap S_1[0] dengan S_1[1], S_1 = {1, 0, 2, 3})

4. *Iterasi* 2 (i = 1,j = 1, S_1 = {1, 0, 2, 3}

j = 0

Swap S_1[1] dengan S1[0], S_1 = {0, 1, 2, 3}

5. *Iterasi* 3 (i = 2,j = 0, S_1 = {0, 1, 2, 3})

j = 3

swap S_1[2] dengan S_1[3], S_1 = {0, 1, 3, 2})

6. *Iterasi* 4 (i = 3, j = 3, S_1 = {2, 1, 3, 0}

j = 0

Swap S_1[3] dengan S_1[0], S_1= {2, 1, 3, 0}

Proses PRGA untuk mendapatkan nilai WK adalah sebagai berikut :

1. For WK = 0 to l – 1, dimana l = 2

($i = j = 0$, $S_1 = (2, 1, 3, 0\})$

Iterasi 1 (WK – 0)

$i = (i + 1) \bmod 4$

$i = (0 + 1) \bmod 4 = 1$

$j = (j + S_1[i]) \bmod 4$

$j = (0 + S_1[1]) \bmod 4$

$j = (0+ 1) \bmod 4 - 1$

Swap $S_1[i]$ dengan $S_1[j]$

Swap $S_1[1]$ dengan $S_1[1]$, $S_1 = \{2, 1, 3, 0\}$

Output = $S[(S_1[i] + S_1[j]) \bmod 4]$

Output = $S[(1 + 1) \bmod 4]$

Output = $S[2] = 3$

$WK[0] = 3$

2. *Iterasi* 2 (WK = 1)

($i = 1$, $j = 1$, $S_1 = \{2, 1, 3, 0\}$)

$i = (i + 1) \bmod 4$

$i = (1 + 1) \bmod 4 = 2$

$j = (j + S_1[i]) \bmod 4$

$j = (1 + S_1[2] \bmod 4$

$j = (1 + 3) \bmod 4 = 0$

Swap $S_1[i]$ dengan $S_1[j]$

Swap $S_1[2]$ dengan $S_1[0]$, $S_1 = \{3, 1, 2, 0\}$

Output = $S[(S_1[i] + S_1[j] \bmod 4]$

Output = $S[(2 + 3) \bmod 4]$

Output = $S[1] = 1$

$WK[1] = 1$

Maka WK = {3, 1}

Proses KSA dengan masukan WK dan S_2, proses ini sama halnya dengan proses sebelumnya, hanya saja jika sebelumnya menggunakan K dan S_1 maka kali ini yang digunakan adalah WK dan S_2. Pada akhir dari proses ini diperoleh $S_2 = \{3, 1, 2, 0\}$. Setelah S_1 dan S_2 diperoleh dari proses KSA, selanjutnya masuk ke tahap PRGA.

Proses PRGA dengan masukan S_1 dan S_2 adalah sebagai berikut :

1. Iter*asi* 1 ($i = j_1 = j_2 = 0$, $S_1 = \{3, 1, 2, 0\}$, $S_2 = \{3, 1, 2, 0\}$)

$i = (i + 1) \bmod 4$

$i = (0 + 1) \bmod 4 = 1$

$j_1 = (j_1 + S_1[i]) \bmod 4$

$j_1 = (0 + S_1[1]) \bmod 4$

$j_1 = (0 + 1) \bmod 4 = 1$

Swap $S_1[i]$ dengan $S_1[1]$, $S_1 = \{3, 1, 0, 2\}$

Output $= S_2[(S_1[i] + S_1[j_1]) \bmod 4]$

Output $= S_2[(1+1) \bmod 4]$

Output $= S_2[2] = 2$

$j_2 = (j_2 + S_2[i]) \bmod 4$

$j_2 = (0 + S_2[1]) \bmod 4$

$j_2 = (0 + 1) \bmod 4 = 1$

Swap $S_2[i]$ dengan $S_2[j_2]$

Swap $S_2[2]$ dengan $S_2[3]$ $S_2 = \{3, 1, 0, 2\}$

Output $= S_1[(1 + 1) \bmod 4]$

Output $= S_1[2] = 2$

Selanjutnya *plaintext* N di XOR dengan 2 kemudian di XOR lagi dengan 2, maka:

N (78) = 01001110

$S_2(2)$ = 00000010 XOR

= 01001100

$S_1(2)$ = 00000010 XOR

= 01001110 (N)

2. *Iterasi* 2 ($i = 1, j_1 = 1, j_2 = 1, S_1 = \{3, 1, 2, 0\}$, $S_2 = \{3, 1, 2, 0\}$

$i = (j+1) \bmod 4$

$i = (1+1) \bmod 4 = 2$

$j_1 = (j_1 + S_1[i]) \bmod 4$

$j_1 = (j_1 + S_1[2]) \bmod 4$

$j_1 = (1 + 2) \bmod 4 = 3$

Swap $S_1[i]$ dengan $S_1[3]$, $S_1 = \{3, 1, 0, 2\}$

Output $= S_2[(S_1[i] + S_1[j_1]) \bmod 4]$

Output $= S_2[(0 +2) \bmod 4]$

Output $= S_2[2] = 2$

$j_2 = (j_2 + S_2[i]) \bmod 4$

$j_2 = (1 + S_2[2]) \bmod 4$

$j_2 = (1 + 2) \bmod 4 = 3$

Swap $S_2[i]$ dengan $S_2[j_2]$

Swap $S_2[2]$ dengan $S_2[3]$ $S_2 = \{3, 1, 0, 2\}$

Output = $S_1[(0 + 2) \bmod 4]$

Output = $S_1[2] = 0$

Selanjutnya *plaintext* H di XOR dengan 2 kemudian di XOR lagi dengan 0, maka:

H (72) = 01001000

$S_2(2)$ = 00000010 XOR

= 01001010

$S_1(2)$ = 00000000 XOR

= 01001000

3.4. Tahapan Cara Kerja Algoritma Rprime RSA

Algoritma Rprime RSA merupakan salah satu algoritma yang termasuk dalam algoritma kriptografi asimetris. Ada tiga proses pada algoritma ini yaitu, proses pembangkitan kunci, proses enkripsi dan proses dekripsi. Dalam pembangkitan kunci algoritma Rprime RSA akan didapatkan dua kunci yaitu, kunci publik dan kunci privat.

3.4.1. Proses Pembangkitan Kunci Algoritma Rprime RSA

Algoritma Rprime RSA menggunakan dua kunci yaitu kunci privat dan kunci publik. Kunci privat pada algoritma Rprime RSA terdiri dari (N,e) dan kunci publik terdiri dari (p, dp). Untuk mendapatkan kunci tersebut, dapat dilihat pada proses berikut:

- Ambil secara acak bilangan prima, untuk $p_1 = 23$ $p_2 = 19$ $p_3 = 29$ dengan syarat:

$\gcd(p_1-1,p_2-1) = \gcd(p_2-1,p_3-1) = \gcd(p_1-1,p_3-1) = 1$

$\gcd(23,19) = 1$

$\gcd(19,29) = 1$

$\gcd(23,29) = 1$

- Kemudian hitung $N = p \times p \times p$ maka akan didapat N = 12673.
- Ambil secara acak bilangan prima, untuk $dp_1 = 17$ $dp_2 = 5$ $dp_3 = 9$ dengan syarat:

$dp_1 = dp_2 = dp_3 = 1 \pmod 2$ dan $\gcd(dp_1,p_1-1) = \gcd(dp_2,p_2-1) = \gcd(dp_3,p_3-1) = 1$

$\gcd(17,22) = 1$

gcd(5,18) = 1

gcd(9,28) = 1

- Cari nilai d dengan :

$d = dp_1 \pmod{p_1-1}$

d = 17 (mod 22)

d = 5 (mod 18)

d = 9 (mod 28)

gcd (22, 18) = 2

17 mod 2 = 1 (mod 2)

5 mod 2 = 1 (mod 2)

d = 6 (mod 11) ..... persamaan 1

gcd (18, 28) = 2

5 mod 2 = 1 (mod 2)

9 mod 2 = 1 (mod 2)

d = 5 (mod 9) ..... persamaan 2

d = 9 (mod 28) .....persamaan 3

jadi

d = 6 (mod 11)

d = 5 (mod 9)

d = 9 (mod 28)

gcd (11,9) = 1
gcd (11,28) = 1
gcd (9,28) =1

$N = lcm (28 \times 11 \times 9) = 2772$

$m_1 = N / n_1 = 2772 / 11 = 252$

$m_2 = N / n_2 = 2772 / 9 = 308$

$m_3 = N / n_3 = 2772 / 28 = 99$

$d = m_1^{-1} \bmod n_1 = 252^{-1} \bmod 11 = 10$

$d = m_2^{-1} \bmod n_2 = 308^{-1} \bmod 9 = 5$

$d = m_3^{-1} \bmod n_3 = 99^{-1} \bmod 28 = 15$

CRT $d = (n_1.m_1.d_1 + n_1.m_1.d_1 + n_1.m_1.d_1) \pmod{N}$

$d = (11.252.10 + 9.308.5 + 9.99.15) \pmod{2772}$

d = 149

- Hitung nilai $\phi (n) = (p_1-1) \times (p_2-1) \times (p_3-1)$

ϕ (n) = 22 ×18×28

ϕ (n) = 11088

- Tentukan nilai e = d^{-1} (mod ϕ(n)), dapat dilihat pada Tabel 3.1. berikut:

Tabel 3.1. Penyelesaian Nilai e

E	e × 149 (mod 11088)
1	149
2	289
.	.
.	.
.	.
893	1

Dari proses pembangkitan kunci, didapatkan kunci publik (12673,893) dan kunci privat ((23,19,29)(17,5,9)).

3.4.2. Proses Enkripsi Algoritma Rprime RSA

Langkah-langkah melakukan proses enkripsi dengan algoritma Rprime RSA:

- Dapatkan nilai publik key

N = 12673

e= 893

- Misalkan plainteks “ABC” pesan diubah menjadi ASCII.

A = 65

B = 66

C = 67

- Enkripsi dengan rumus C = M^e mod n

untuk karakter A = 41

C = M^e mod N

C = 41^{893} mod 12673 = 4855

Ciphertext karakter A = 4855

untuk karakter B = 42

C = M^e mod N

C = 42^{893} mod 12673 = 2353

Ciphertext karakter B = 2353

untuk karakter C = 43

$C = M^e \bmod N$

$C = 43^{893} \bmod 12673 = 5212$

Ciphertext karakter C = 5212

Jadi *Ciphertext* "ABC" = 4855 2353 5212"

3.4.3. Proses Dekripsi Algoritma Rprime RSA

Langkah-langkah melakukan proses dekripsi dengan algoritma Rprime RSA:

- Dapatkan *Ciphertext*:

Ciphertext = "4855 2353 5212"

- Dekripsi dengan rumus $M = C^{dpi} \bmod p_i$

untuk karakter A = 41

$M = C^{dpi} \bmod p_i$

$M = 4855^{17} \bmod 23 = 18$

$M = 4855^{5} \bmod 19 = 3$

$M = 4855^{29} \bmod 29 = 12$

CRT M = 41

Plainteks karakter A = 41

untuk karakter B = 42

$M = C^{dpi} \bmod p_i$

$M = 2353^{17} \bmod 23 = 19$

$M = 2353^{5} \bmod 19 = 4$

$M = 2353^{29} \bmod 29 = 13$

CRT M = 42

Plainteks karakter A = 42

untuk karakter C = 43

$M = C^{dpi} \bmod p_i$

$M = 5212^{17} \bmod 23 = 20$

$M = 5212^{5} \bmod 19 = 5$

$M = 5212^{29} \bmod 29 = 14$

CRT M = 42

Plainteks karakter C = 42

- Dari seluruh proses dekripsi, pesan yang berupa *Ciphertext* dapat dikembalikan menjadi plainteks seperti semula, yaitu:

  4855 2353 5212

  A B C

BAB 4

IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi Sistem

Pada tahap implementasi sistem merupakan tahap dilakukan proses implementasi terhadap sistem yang sebelumnya telah dirancang dan dianalisis. Dalam implementasi sistem ini digunakan teknik kriptografi hybrid dengan menggunakan dua algoritma yang termasuk dalam algoritma simetris dan asimetris yaitu algoritma RC4A dan algoritma Rprime RSA untuk pengamanan data teks khususnya data teks dengan format *.txt dan *.doc. Sistem ini dibangun dengan bahasa pemrograman C# dan perangkat lunak yang digunakan sebagai *Integrated Development Environment (IDE)* adalah Visual Studio 2012.

Spesifikasi perangkat keras yang digunakan dalam pembangunan sistem dan pengujian sistem adalah sebagai berikut:

Tabel 4.1 Spesifikasi Perangkat Keras untuk Implementasi

Spesifikasi Komputer
CPU : Intel(R) Core(TM) i5-2430M CPU @ 2,40GHz RAM : 4 GB Monitor : 14 Inc Sistem Operasi : Windows 7 Ultimate

4.2. Tampilan Antarmuka Sistem

Dalam membangun sistem kriptografi *hybrid* ini diimplementasikan dengan beberapa tampilan menu yang terdiri dari beberapa *form*, yaitu *form* utama, *form* pembangkit kunci, *form* enkripsi dan *form* dekripsi.

4.2.1. Tampilan Form Utama

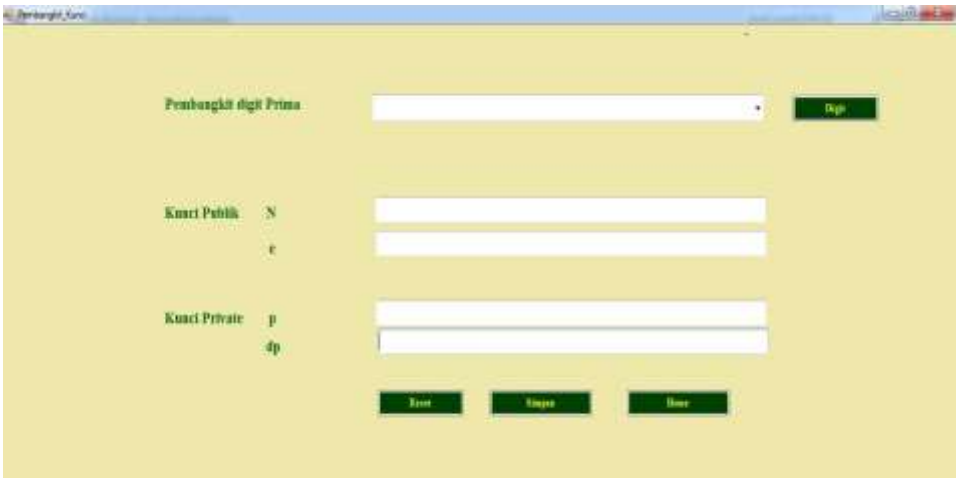
Ketika sistem pertama kali dijalankan maka tampilan awal yang akan muncul adalah *form* utama. *Form* utama ini berisi tampilan judul skripsi, logo fakultas, nama dan nim

penulis. *Form* ini mempunya tiga *button menu* yaitu menu pembangkitan kunci, enkripsi, dan dekripsi. Berikut Gambar 4.1 *form* utama atau home.

Gambar 4.1. Antarmuka Form Utama

4.2.2. Tampilan Form Pembangkitan Kunci

Pada menu pembangkit kunci terdapat *form* algoritma Rprime RSA yang berguna untuk membantu pengguna aplikasi sistem kriptografi *hybrid* dalam membangkitkan kunci yaitu kunci publik yang terdiri dari N, e dan kunci privat yang terdiri dari p, dp. Hasil kunci yang telah dibangkitkan dapat disimpan menjadi sebuah file kunci publik dan kunci privat yang akan digunakan untuk proses selanjutnya. Berikut Gambar 4.2 *form* pembangkit kunci.



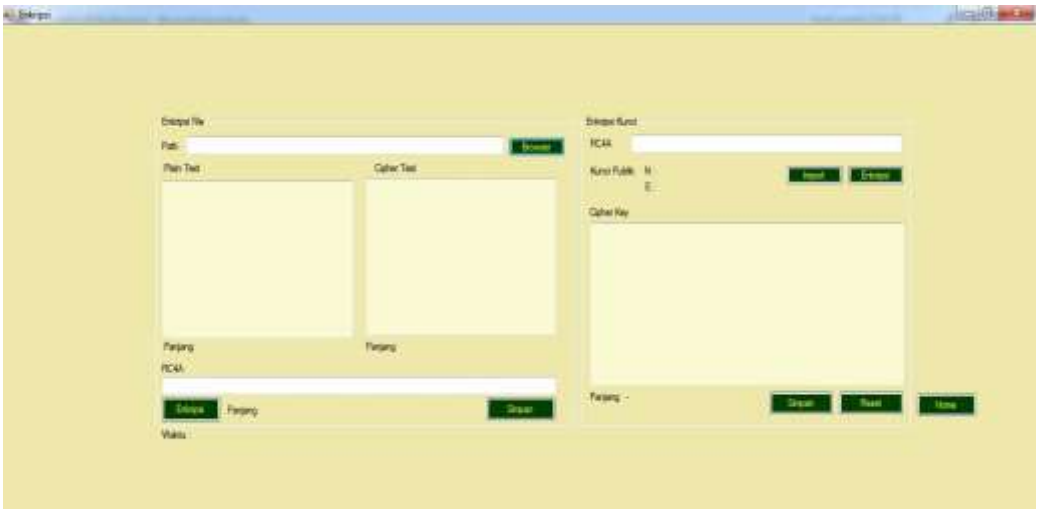
Gambar 4.2. Antarmuka Form Pembangkit Kunci

Penjelasan Gambar 4.2.

1) *User* membutuhkan kunci privat yang terdiri dari p dan dp sebagai kunci privat yang digunakan untuk proses selanjutnya, p dan dp yang dimasukkan harus bilangan prima. Untuk mempermudah *user*, sistem aplikasi ini menggunakan pembangkit bilangan prima otomatis dengan menekan tombol digit.
2) Untuk menentukan kunci publik yang terdiri dari N dan e, *user* dapat menekan tombol Digit agar N dan e dapat dihitung otomatis oleh sistem.
3) Setelah kedua kunci didapatkan, yaitu kunci privat dan kunci publik maka *user* dapat menyimpan kedua kunci tersebut dengan menekan tombol Simpan.

4.2.3. Tampilan Form Enkripsi

Pada form ini terdapat menu enkripsi file dan enkripsi kunci. Didalam *form* ini pengguna dapat melakukan proses enkripsi pesan yang berupa file teks, khususnya file teks dengan format *.txt dan *.doc selain itu dalam *form* ini pengguna juga dapat memasukkan kunci RC4A. Hasil yang di dapat yaitu pesan *Ciphertext* dapat disimpan sebagai file yang akan digunakan untuk proses berikutnya. Proses enkripsi kunci menggunakan algoritma Rprime RSA sehingga pengguna memerlukan kunci publik yang sebelumnya harus dibangkitkan terlebih dahulu. Hasil dari proses enkripsi adalah berupa *Cipherkey* yang dapat disimpan sebagai *file* untuk melanjutkan proses berikutnya. Berikut Gambar 4.3 *form* enkripsi.



Gambar 4.3. Antarmuka Form Enkripsi

Penjelasan dari Gambar 4.3.

1. *User* dapat memilih *file* yang akan dienkripsi dari direktori yang dituju dengan menekan tombol *Browse*. Untuk pemilihan *file* dibatasi hanya *file* dengan format

*.txt dan *.doc yang dapat dienkripsi. Selain itu *user* juga dapat menginput langsung pesan yang akan dienkripsi dengan mengetikkan pesan pada *textbox* dengan label *plainteks*.

2. Selanjutnya user memasukkan kunci RC4A yang diinginkan di textbox dengan labael RC4A sebelum melakukan proses enkripsi.
3. Jika komponen-komponen yang dibutuhkan sudah terisi, maka untuk melakukan proses enkripsi, *user* dapat menekan tombol Enkripsi. Hasil dari enkripsi berupa *Ciphertext* yang akan ditampilkan pada *textbox* dengan label *Ciphertext*.
4. Selanjutnya *user* dapat menyimpan hasil enkripsi yang berupa *Ciphertext* dalam bentuk bilangan heksadesimal.
5. Kemudian pada proses enkripsi kunci user dapat memasukkan RC4A pada *textbox* dengan label yan sudah diinputkan RC4A sebelumnya .
6. Untuk mengenkripsi kunci RC4A menggunakan algoritma Rprime RSA dibutuhkan kunci publik yang sebelumnya harus dibangkitkan terlebih dahulu. Untuk memasukkan kunci publik, *user* dapat menekan tombol import. Setelah itu kunci akan ditampilkan pada *textbox* yang berlabel Kunci Publik.
7. Untuk memulai proses enkripsi, *user* dapat menekan tombol Enkripsi dan hasil dari proses tersebut adalah *Cipherkey* yang akan ditampilkan pada *textbox* dengan label *Cipherkey*.
8. Setelah hasil enkripsi ditampilkan, *user* dapat menyimpan *Cipherkey* tersebut dengan menekan tombol Simpan *Cipherkey*.

4.2.4. Tampilan Form Dekripsi

Form dekripsi *Cipherkey* dan *ciphertext* terdapat di menu dekripsi. Pada *form* ini pengguna dapat melakukan proses dekripsi *Cipherkey* yang bertujuan untuk mengembalikan kunci RC4A. Proses dekripsi *Cipherkey* menggunakan algoritma Rprime RSA sehingga diperlukan kunci privat yang sebelumnya harus dibangkitkan terlebih dahulu. Jika proses eksekusi sudah selesai, maka hasilnya yang berupa kunci RC4A dapat disimpan sebagai sebuah *file* untuk digunakan pada proses selanjutnya. Pada *form ciphertext* ini pengguna dapat melakukan proses dekripsi pesan yang bertujuan untuk mengembalikan *file* yang masih berupa *Ciphertext* menjadi *file plainteks* seperti semula. Proses dekripsi pesan menggunakan algoritma RC4A. Jika proses dekripsi telah selesai maka pesan akan kembali menjadi *plainteks*. Berikut Gambar 4.4 *form* dekripsi.

Gambar 4.4. Antarmuka Form Dekripsi

1. Untuk melakukan proses dekripsi, maka user harus menggunakan kunci privat yang harus dibangkitkan terlebih dahulu. *User* dapat menekan tombol Import Kunci untuk memasukkan kunci privat.
2. *User* dapat menemukan di folder mana *Cipherkey* disimpan dengan menekan tombol Import, setelah *file* dipilih maka akan tampil isi dari *file Cipherkey* di *textbox* dengan label *Cipherkey*.
3. *User* dapat melanjutkan ke proses dekripsi dengan menekan tombol dekripsi, dan hasil dari dekripsi *Cipherkey* akan ditampilkan pada textbox dengan label plainkey.
4. Setelah hasil dekripsi *Cipherkey* didapatkan, user dapat menyimpan kunci RC4A tersebut dengan menekan tombol Simpan.
5. Kemudian user harus memasukkan atau memilih kunci RC4A terlebih dahulu dengan menekan tombol Import. Kunci RC4A yang sudah dimasukkan akan ditampilkan pada *textbox* dengan label RC4A.
6. *User* dapat memilih dan memasukkan *file Ciphertext* yang akan didekripsi dengan menekan tombol *Import*. Setelah *file* telah terpilih, maka isi dari *file* tersebut akan ditampilkan pada textbox dengan label *Ciphertext*.
7. Selanjutnya untuk memproses pendekripsian file *Ciphertext*, *user* dapat menekan tombol dekripsi. Setelah proses dekripsi selesai maka pesan *Ciphertext* akan berubah menjadi pesan *plainteks* yang ditampilkan pada *textbox* dengan label Plainteks.

4.2.5. Tampilan Form Bantuan

Form bantuan memberitahukan langkah-langkah dari proses pembangkitan kunci, enkripsi, dan dekripsi. Gambar 4.5 merupakan tampilan dari *form* bantuan.

Gambar 4.5. Antarmuka Form Bantuan

4.3. Pengujian Sistem

Pengujian yang dilakukan terhadap sistem yang telah dibangun adalah untuk membuktikan bahwa sistem tersebut sudah berjalan dengan baik dan sesuai dengan analisis dan perancangan sistem yang telah dibuat sebelumnya.

Untuk melakukan pengujian terhadap sistem yang dibuat pada penelitian ini akan dirancang suatu skenario dalam mengamankan suatu pesan teks. Adapun skenario yang akan dibuat adalah pengguna sistem akan mengamankan suatu pesan teks dengan cara melakukan proses enkripsi dan selanjutnya akan dilakukan proses enkripsi kembali untuk kunci dari pesan teks tersebut. Sebelum melakukan enkripsi pada kunci, pengguna harus membangkitkan sepasang kunci dari algoritma Rprime RSA. Untuk mengenkripsi kunci dari pesan teks dibutuhkan kunci publik sehingga akan dihasilkan *Cipherkey*. Pada skenario selanjutnya, pengguna mempunyai *Ciphertext*, *Cipherkey* dan kunci privat yang sudah dibangkitkan sebelumnya. Kemudian akan dilanjutkan dengan skenario pengguna melakukan proses dekripsi *Cipherkey* untuk mendapatkan kembali kunci pengaman pesan teks, dan setelah itu dilakukan kembali proses dekripsi pesan untuk mendapatkan kembali pesan plainteks.

4.3.1. Pengujian Pembangkitan Kunci

Misalkan pengirim bernama Alice dan penerima bernama Bob. Alice mempunya pesan rahasia yang ingin dia kirimkan kepada Bob dan pesan tersebut akan diamankan dengan menggunakan sistem, maka hal pertama yang dilakukan Alice adalah menerima kunci publik dari Bob. Sebelumnya Bob harus membangkitkan kunci dari Algoritma Rprime RSA terlebih dahulu. Gambar 4.6. berikut adalah proses membangkitkan kunci, yaitu kunci privat dan kunci publik. Kemudian hasil dari kunci tersebut dapat disimpan untuk melakukan proses selanjutnya.

Gambar 4.6. Pengujian sistem pembangkitan kunci

4.3.2. Pengujian Enkripsi File Teks

Setelah Alice menerima kunci publik yang dibangkitkan oleh Bob, maka langkah selanjutnya adalah Alice akan melakukan proses enkripsi pesan. Pesan yang akan dienkripsi oleh Alice adalah sebuah data teks dengan format *.txt, *.doc dan proses enkripsi menggunakan algoritma RC4A.

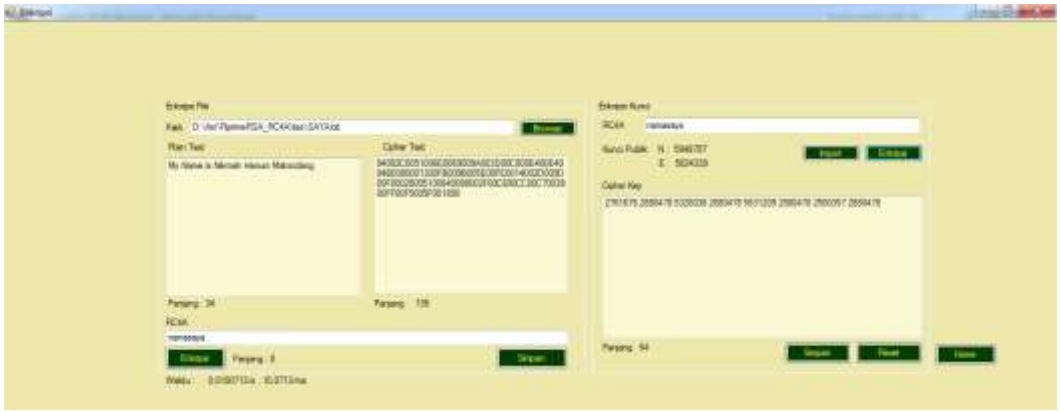
Gambar 4.7. user menekan tombol browse untuk mencari file yang ingin dienkripsi kemudian masukkan kunci RC4A. Hasil proses enkripsi berupa *Ciphertext* yang dapat disimpan untuk melakukan proses selanjutnya.

Gambar 4.7. Pengujian sistem enkripsi file teks

4.3.3. Pengujian Enkripsi Kunci RC4A

Skenario selanjutnya adalah melakukan proses enkripsi pada kunci RC4A untuk membuat kunci tersebut lebih aman. Enkripsi dilakukan dengan menggunakan algoritma Rprime RSA, oleh sebab itu Alice harus mempunyai kunci publik yang sudah dibangkitkan oleh Bob untuk melakukan proses enkripsi pada kunci RC4A. Hasil dari proses enkripsi tersebut adalah berupa sebuah *Cipherkey*.

Gambar 4.8. adalah langkah-langkah proses enkripsi kunci RC4A yang akan dilakukan oleh Alice dengan menggunakan kunci publik dari algoritma Rprime RSA. Dari proses tersebut akan dihasilkan *Cipherkey* yang dapat disimpan untuk melakukan proses berikutnya.



Gambar 4.8. Pengujian sistem enkripsi kunci

4.3.4. Pengujian Dekripsi Kunci

Skenario berikutnya, Bob sebagai penerima pesan rahasia dari Alice akan melakukan proses dekripsi. Yang harus didekripsi terlebih dahulu adalah *Cipherkey* untuk

mendapatkan kunci RC4A. Proses dekripsi dilakukan dengan menggunakan algoritma Rprime RSA, maka Bob memerlukan kunci privat yang sebelumnya telah ia bangkitkan.

Gambar 4.9. merupakan langkah-langkah proses dekripsi *Cipherkey* yang akan dilakukan oleh Bob. Hasil dekripsi berupa kunci RC4A dan dapat disimpan untuk melanjutkan proses berikutnya.

Gambar 4.9. Pengujian sistem dekripsi kunci

4.3.5. Pengujian Dekripsi Pesan Teks

Dalam tahap ini Bob sebagai penerima memiliki *file Ciphertext* dan kunci RC4A untuk melakukan skenario untuk proses dekripsi pesan. Proses dekripsi ini dilakukan dengan menggunakan algoritma RC4A untuk mengubah *Ciphertext* menjadi file pesan asli atau yang sering disebut dengan plainteks. Tujuannya adalah agar Bob dapat membaca dengan jelas pesan yang dikirimkan oleh Alice.

Gambar 4.10. adalah langkah-langkah proses dekripsi *Ciphertext* yang akan dilakukan oleh Bob.

Gambar 4.10. Pengujian sistem dekripsi *file*

4.4. Hasil Pengujian Sistem

Dari Hasil pengujian yang sudah dilakukan dengan beberapa skenario Bob dan Alice, maka didapatkan kesimpulan bahwa dari beberapa keutuhan data untuk proses enkripsi dan dekripsi dapat diselesaikan dengan menggunakan sistem yang telah dibuat, keseluruhannya berjalan dengan baik sesuai seperti rancangan awal. Dalam hal ini pengujian dilakukan 4 kali percobaan. Tabel-tabel berikut merupakan tabel yang menunjukkan hasil percobaan keutuhan data.

Tabel 4.2. Hasil implementasi sistem pada percobaan 1.

Plainteks	My Name Is Nikmah Hanum Matondang
Kunci RC4A	namasaya123
Kunci Publik	N = 16500881, E = 3515063
Kunci Private	P = 283, 293, 199 Dp = 101, 115, 83
Cipherkey	52129368 154880425 135410750 199917551 161281016 199917551 19400039 199917551 161281016 85858522 117256724 135410750 154880425 52129368 161281016
Ciphertext	98002E0044009C001E005700FB001F00D100E500F600F10 0AF003100CB006E004A00CA007800F100080016007400A 900B400BE00340078007000B100D800EA004C00AD00
Ukuran File Enkripsi	140 bytes
Ukuran File Dekripsi	34 bytes

Tabel 4.3. Hasil implementasi sistem pada percobaan 2.

Plainteks	Kriptografi berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua Kriopto dan Graphia. Kripto berarti secret (rahasia) dan Graphia berarti Writing (tulisan). Menurut terminologinya Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan).
Kunci RC4A	Kriptografi RC4A Rprime RSA
Kunci Publik	N = 78725431, E = 45094687
Kunci Private	P = 461, 439, 389 Dp = 223, 235, 219

Cipherkey	10853638 20726187 23996036 44640405 71967697 1208647 69335331 20726187 48896577 63254203 23996036 70859570 5147581 9400555 23573711 27004060 70859570 70859570 5147581 44640405 20726187 23996036 42967582 39372084 70859570 5147581 24826416 27004060
Ciphertext	EB0070005400AA00F9000A00FE008100AB00F70094000A00E300980061000200D000B600E8002C00CA0070009B009D008C00A6009A008200BA0049004B00C9008D00DA00D9008F000A00E5007A0062004B005F00FC00FC005C002500F1008C009000A100FF003000EC001000D8006E005800F0008F008B004800C9007100550036004A00E90026008C00CE0016006800E1003B00B7003500AD002000000013009D0037002C0061008800D4005F0041004C0002004300BF001B00B4003900EB006E00C3005C00A1004000AB00D200CD003600BD0003004700C00080004A003B000D0016001E008000D600E5005A005F00B500AB009B00DC006C00F200C70004001B00030094008F00A100DE0084009F00EC003E00F300B10012009B00D50068004300F100E70095003500B4000100D500C900FF00FE0060002A00E5001500750088004200230005F0009004E00F5007D004400C2005700DB00C800D4006E001800D70097001400460014008300AE00D20074000D00D700FD00F800E500B8007D001F0054009B002F00D400C100540076009A00CF00B0000200FC0016005000BD00A1003800FD00E50000006A00A0009A000000E100E70073009D000C001D005B00CE00C30029005E00DE004D000B00B7006300E900D600B4009400040007005A0086001A0086009800200007F002400F900060006008A003B00D100A500EE00DB00
Ukuran File Enkripsi	23.040 bytes
Ukuran File Dekripsi	22.016 bytes

Tabel 4.4. Hasil implementasi sistem pada percobaan 3.

Plainteks	ILMU KOMPUTER DAN TEKNOLOGI INFORMASI ILMU KOMPUTER UNIVERSITAS SUMATERA UTARA
Kunci RC4A	University
Kunci Publik	N = 172649433, E = 36791761
Kunci Private	P = 561, 467, 659 Dp = 321, 289, 181

Cipherkey	43351392 120711041 7273470 21251359 84758225 46706730 13953307 7273470 99993317 126962836
Ciphertext	6A00570064001F0061003D001D00BE00A8005200 A000EA00E80036000D00BE00560080007600CA00 D600CA006600AA00C9000D007D00BA00AD0058 0063007F009A00FB00BB005E00AD007D00B6004 D00FB000000FF00AC00EA0092005000F900E5001 8003C003000BC00EC0042008500B4005F00500064 006000600095002B001900BE002500FB006500620 02F0057004B00B80011001C002D004B00EE009D0 0
Ukuran File Enkripsi	22.016 bytes
Ukuran File Dekripsi	22.016 bytes

Tabel 4.5. Hasil implementasi sistem pada percobaan 4.

Plainteks	senin selasa rabu kamis jumat sabtu minggu
Kunci RC4A	hari ini cerah
Kunci Publik	N = 215936597, E = 89230853
Kunci Private	P = 683, 677, 467 Dp = 359, 277, 437
Cipherkey	52129368 154880425 135410750 199917551 161281016 199917551 19400039 199917551 161281016 85858522 117256724 135410750 154880425 52129368 161281016
Ciphertext	F5002200BD00E2003C00170055008300690 0D300CB0037 004E005600B6005C000800A0005E00D000AE007600FC0 085009900FE00C800B9000F005400D000B40037005C008 300CD0019000600EC00130086005100
Ukuran File Enkripsi	22.016 bytes
Ukuran File Dekripsi	42 bytes

Pada tabel-tabel diatas dapat dilihat bahwa hasil dari *ciphertext* dan *Cipherkey* berupa bilangan heksadesimal. Percobaan ini berhasil mengembalikan *ciphertext* menjadi *plainteks* dan *Cipherkey* menjadi kunci RC4A. Ukuran file pada saat enkripsi semakin besar dari pada ukuran file pada saat dekripsi. Hal ini dapat disimpulkan bahwa *Hybrid cryptosystem* Algoritma Rprime RSA dan Algoritma RC4A memenuhi keutuhan data.

BAB 5

KESIMPULAN DAN SARAN

5.1. KESIMPULAN

Berdasarkan pembahasan dan hasil dari penelitian, maka diperoleh beberapa kesimpulan sebagai berikut:

1. Proses enkripsi dan dekripsi dari metode *hybrid* algoritma Rprime RSA dan algoritma RC4A memenuhi kriteria keutuhan data.
2. *Cipherkey* yang dienkripsi dengan Algoritma Rprime RSA menjadi lebih panjang dari sebelumnya, karena setiap karakter diubah menjadi hexa.
3. Banyaknya karakter pada data teks yang akan dienkripsi atau didekripsi mempengaruhi waktu proses, semakin banyak karakter yang akan diproses maka semakin banyak pula waktu yang dibutuhkan untuk enkripsi atau dekripsi.

5.2. SARAN

Berikut ini adalah saran yang dapat digunakan untuk tahap pengembangan penelitian sistem ini antara lain:

1. Dalam penelitian ini pesan yang dapat dienkripsi hanya data teks dengan format *.txt dan *.doc sehingga diharapkan untuk pengembangan kedepannya bisa mengenkripsi pesan teks dengan format yang lain seperti *.doc dan lain-lain.
2. Untuk pengembangan selanjutnya diharapkan menambahkan penjang digit bilangan prima yang lebih besar lagi.
3. Algortima Rprime RSA nilai p dan dp lebih banyak lagi diharapkan untuk pengembangan selanjutnya.

DAFTAR PUSTAKA

Ariyus, Dony. 2005. *Kriptografi Keamanan Data dan Komunikasi.* Yogyakarta: Penerbit GRAHA ILMU.

Ariyus, Dony. 2008. *Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi.* Yogyakarta: Penerbit ANDI.

Karahan, Mahmed. 2015. *New Attacks On Rc4a And Vmpc* . Thesis. Bilkent University.

Kromodimoeljo, Sentot. 2010. *Teori dan Aplikasi Kriptografi*. Jakarta: Penerbit SPK IT Consulting.

Lubis, Fata Ismail. 2015. *Implementasi Super Enkripsi Algoritma Elgamal Dengan Teknik Transposisi Segitiga*. Skripsi. Universitas Sumatera Utara.

Masya, Isnaini Rahma. 2016. *Pemanfaatan Affine Cipher Dalam Three-Pass Protocol Pada Pengamanan Citra Bitmap*. Skripsi. Universitas Sumatera Utara.

Mizfar, Al. 2016. *Pengamanan Text File Menggunakan Modifikasi Algoritma Playfair Cipher dab Algoritma Elgamal*. Skripsi. Universitas Sumatera Utara.

Mollin, R. A. 2007. *An Introduction to Cryptography Second Edition*. Chapman & Hall/CRC: Florida.

Nasution, Nurhaliza. 2016. *Implementasi Hybrid cryptosystem Algoritma RC4 dan Elgamal Dalam Pengamanan File Text*. Skripsi. Universitas Sumatera Utara.

Noman, A.A., Sidek, M.R., Ramli, R.A., Ali, L. 2008. *RC4A Strem Cipher for WLAN Security*. A Hardware Approach : 624-627.

Noman, A.A., Sidek, M.R., Ramli, R.A., Ali, L. 2009. *Hardware Implementation of RC4A Stream Cipher*. International Jurnal of Cryptology Research 1(2): 225-233.

Padhye, Sahadeo. 2002. *An Efficient Variant Of RSA Cryptosystem*. *CSIR (JRF) scheme* : 1-4.

Schneier, B. 1996. *Applied Cryptography: Protocols, Algorithms, and Sorce Code in C*. Wiley: New York.

Sujiono, Rizky M D. 2016. *Implementasi Three-pass Protocol Dengan Kombinasi Algoritma Beaufort Cipher dan One Time Pad*. Skripsi. Universitas Sumatera Utara.

Verma, S & Garg, D. 2011. *Improvement in RSA Cryptosystem*. J. *Jurnal Of Advances in Information Technology* 2(3): 146 – 151.

Verma, S., & Garg, D. March 2009. I*mprovement over Public Key Cryptographic Algorithm*. *IEEE International Advance Computing Conference* : 734 - 739.

Verma, S., & Garg, D. November 2015. *Improvement in Rebalanced CRT RSA*. *The International Arab Journal of Information Technology* 12 (6) : 524 - 531.

Whitthen, J. L., Bentley, L. D. 2007. *System Analysis and Design Methods Seventh Edition*. New York: McGraw-Hill.

Widarma, Adi. 2016. *Kombinasi Algoritma AES, RC4, dan Elgamal Dalam Skema Kriptografi Hybrid Untuk Keamanan Data*. Tesis. Universitas Sumatera Utara.

LAMPIRAN A
LISTING PROGRAM

1. MAIN PROGRAM

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace RprimeRSA_RC4A
{
    public partial class FrmMain : Form
    {
        public FrmMain()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            FrmPembangkit_Kunci kunci = new FrmPembangkit_Kunci();
            kunci.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form_enkripsi enkripsi = new Form_enkripsi();
            enkripsi.Show();
            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Dekripsi dekripsi = new Dekripsi();
            dekripsi.Show();
            this.Hide();
        }
```

```
    }
}
```

1. Fungsi

A-2

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ChineseRemainderProblem;
using System.Xml;
using System.IO;
using System.Numerics;

namespace ChineseRemainderProblem
{
    public class Fungsi
    {

        public static BigInteger in_totient(BigInteger a, BigInteger
n)
        {
            BigInteger i = n, v = 0, d = 1;
            while (a > 0)
            {
                BigInteger t = i / a, x = a;
                a = i % x;
                i = x;
                x = d;
                d = v - t * x;
                v = x;
            }
            v %= n;
            if (v < 0) v = (v + n) % n;
            return v;
        }

        public static Int64 ckGc(Int64 a, Int64 b)
        {
            while (a != 0 && b != 0)
            {
                if (a > b)
                    a %= b;
                else
                    b %= a;
            }
            return Math.Max(a, b);
        }

        public static BigInteger in_gcd(BigInteger a, BigInteger b)
        {

            if (a > b)
            {
                BigInteger t = b;
                b = a;
```

```
            a = t;
        }

        BigInteger r = b % a;
        while (r != 0)
        {
            b = a;
            a = r;
            r = b % a;
        }

        return a;
    }

    public static bool Coprime(Int64 a, Int64 b)
    {
        return ckGc(a, b) == 1;
    }

    public static BigInteger LCM(BigInteger a, BigInteger b)
    {
        BigInteger gcd = in_gcd(a, b);
        return a * b / gcd;
    }

    public Int64 inversi(Int64 a, Int64 n)
    {
        Int64 i = n, v = 0, d = 1;
        while (a > 0)
        {
            Int64 t = i / a, x = a;
            a = i % x;
            i = x;
            x = d;
            d = v - t * x;
            v = x;
        }
        v %= n;
        if (v < 0) v = (v + n) % n;
        return v;
    }


    public static void Solve(BigInteger a, BigInteger b, out
BigInteger x, out BigInteger y)
    {
        if (b == 1)
        {
            x = 1;
            y = a - 1;
        }
        else if (a == 1)
        {
            y = 1;
            x = b + 1;
        }
        else if (b > a)
        {
            BigInteger subx;
            Solve(a, b - a, out subx, out y);
            x = y + subx;
```

A-3

```
            }
            else if (a > b)
            {
                BigInteger suby;
                Solve(a - b, b, out x, out suby);
                y = x + suby;
            }
            else
            {
                throw new Exception(String.Format("The equation
{0}x={1}y+1 has no integer solution.", a, b));
            }
        }

        public static void diss()
        {
            var form = Form.ActiveForm as
RprimeRSA_RC4A.FrmPembangkit_Kunci;
            if (form != null)
            {
                form.TxtKosong.Text = "";
            }
        }

        public static void Solve(BigInteger a, BigInteger b,
BigInteger c, out BigInteger x1, out BigInteger y1)
        {

            BigInteger d = in_gcd(a, b);

            if (d > 1)
            {
                if (c % d != 0)
                {
                    diss();
                }
                a = a / d;
                b = b / d;
                c = c / d;
            }
            BigInteger x0, y0;
            Solve(a, b, out x0, out y0);

            x1 = (c * x0) % b;
            y1 = (c * y0) % a;
         }

        public static void Solve(BigInteger a, BigInteger b,
BigInteger r1, BigInteger r2, out BigInteger x1, out BigInteger y1)
        {
            if (r2 > r1)
            {

                Solve(a, b, r2 - r1, out x1, out y1);
            }
            else if (r1 > r2)
            {

                Solve(b, a, r1 - r2, out y1, out x1);
            }
```

A- 4

```
            else
            {
                x1 = b;
                y1 = a;

            }
        }

        public static void Solve(Int64 a, Int64 b, Int64 c, Int64
Int64 r2, Int64 r3, out BigInteger n0, out BigInteger t)
        {

            BigInteger x1, y1;
            Solve(a, b, r1, r2, out x1, out y1);

            BigInteger r4 = a * x1 + r1;

            BigInteger x2, y2;
            Solve(a * b, c, r4, r3, out x2, out y2);

            n0 = c * y2 + r3;
            t = LCM(LCM(a, b), c);

        }

        public static Int64 eksponensial(Int64 x, Int64 y, Int64 n)
        {
            Int64 z = 1;
            foreach (var i in Convert.ToString(y, 2))
            {
                if (i == '1')
                {
                    z = (x * (z * z)) % n;
                    continue;
                }
                z = (z * z) % n;
            }
            return z;
        }


        public static string RC4(string input, string key)
        {
            List<int> cip = new List<int>();
            StringBuilder result = new StringBuilder();

            int temp, y, j = 0, j2 = 0;
            int[] S1 = new int[256];
            int[] S2 = new int[256];
            //KSA
            for (int i = 0; i < 256; i++)
            {
                S1[i] = i;
            }
            //PRGA
            for (int i = 0; i < 256; i++)
            {
                j = (key[i % key.Length] + S1[i] + j) % 256;
                temp = S1[i];
                S1[i] = S1[j];
                S1[j] = temp;
```

A-5

```
            }

            for (int i = 0; i < input.Length; i++)
            {
                //y = i % 256;
                y = i % 256;
                j = (S1[y] + j) % 256;
                temp = S1[y];
                S1[y] = S1[j];
                S1[j] = temp;
                int z1 = S1[(S1[i] + S1[j]) % 256];
                int cipher1 = (int)input[i] ^ z1;

                j2 = (j2 + S2[i]) % 256;
                int temp1 = S2[i];
                S2[i] = S2[j2];
                S2[j2] = temp1;
                int z2 = S1[(S2[i] + S2[j2]) % 256];
                int cipher2 = cipher1 ^ z2;

                cip.Add(cipher2);
                result.Append(Convert.ToChar(cipher2));
            }
            return result.ToString();
        }


    }
}
```

A-6

2. FORM PEMBANGKIT KUNCI

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ChineseRemainderProblem;
using System.Xml;
using System.Numerics;

using System.IO;

namespace RprimeRSA_RC4A
{
    public partial class FrmPembangkit_Kunci : Form
    {
        public FrmPembangkit_Kunci()
        {
            InitializeComponent();

        }
            public enum NumberType
        {
            Composite,
            Prime
```

```
        }
        public bool IsPrimePseudo(Int64 integer)
        {
            NumberType type = PseudoPrime(integer);
            return type == NumberType.Prime;
        }

        // Pseudo primality testing with Fermat's theorem
        public NumberType PseudoPrime(BigInteger n)
        {
            BigInteger modularExponentiation =
                           ModularExponentiation(2,
BigInteger.Subtract(n, 1),
                                                 n);
            if (!modularExponentiation.IsOne)
            {
                return NumberType.Composite;
            }
            else
            {
                return NumberType.Prime;
            }
        }

        public   BigInteger   ModularExponentiation(BigInteger   a,
BigInteger b, BigInteger n)
        {
            // a^b mod n
            return BigInteger.ModPow(a, b, n);
        }

        private void btnKunci_Click(object sender, EventArgs e)
        {
            string TxtKey = TxtK.SelectedItem.ToString();
            int Keynya;
            if (int.Parse(TxtKey) == 1)
            {
                 Keynya = 99;
            }
            else {
                 Keynya = int.Parse(TxtKey) * 100;
            }
            int Mulai_batas = Keynya / 2;
            TxtN.Text = "";
            TxtxE.Text = "";
            txtpg.Text = "";
            TxtDP.Text = "";
            int K;
            if (int.TryParse(TxtK.Text, out K))
            {
                Random rnd = new Random();
                List<Int64> l_P = new List<Int64>();
                List<Int64> dp = new List<Int64>();
                List<Int64> pc = new List<Int64>();
            ulang:
                for (int i = 0; i < 3; i++)
                {
                lab:

                    Int64 i64 = rnd.Next(Mulai_batas, Keynya);
```

A-7

```
                    if (i64 % 2 != 0)
                    {
                        if (IsPrimePseudo(i64))
                        {

                            Int64 v = i64 - 1;
                            pc.Add(v);
                            v = 0;
                            l_P.Add(i64);

                        }
                        else
                        {
                            goto lab;
                        }
                    }
                    else
                    {
                        goto lab;
                    }

                }

            if(l_P[0] ==  l_P[1] || l_P[0] == l_P[2] || l_P[1] ==
l_P[2]){
                    l_P.Clear();
                    pc.Clear();
                    goto ulang;
                }

                if  (Fungsi.Coprime(l_P[0],  l_P[1])  !=  true  ||
Fungsi.Coprime(l_P[0],  l_P[2])  !=  true  ||  Fungsi.Coprime(l_P[1],
l_P[2]) != true)
                {

                    l_P.Clear();
                    pc.Clear();
                    goto ulang;
                }

                Int64 ax1 = l_P[0]-1;
                Int64 ax2 = l_P[1]-1;
                Int64 ax3 = l_P[2]-1;
                if   (Fungsi.Coprime(ax1,   ax2)   ==   true   &&
Fungsi.Coprime(ax2, ax3) == true && Fungsi.Coprime(ax1, ax3) == true)
                {
                    l_P.Clear();
                    pc.Clear();
                    goto ulang;
                }
            ulang2:
                for (int i = 0; i <= 2; i++)
                {
                lab2:
                    int dp1 = rnd.Next(Mulai_batas/2, Keynya/2);
                    if (dp1 % 2 != 0)
                    {
                        dp.Add(dp1);
                    }
                    else
```

A-8

```
                    {
                        goto lab2;
                    }

                }
            if (dp[0] == dp[1] || dp[0] == dp[2] || dp[1] == dp[2] ||
dp[0] == l_P[0] || dp[0] == l_P[1] || dp[1] == l_P[2]  )
            {
                dp.Clear();
                    goto ulang2;
            }
                if   (Fungsi.Coprime(dp[0],   dp[1])   !=   true   ||
Fungsi.Coprime(dp[0],   dp[2])   !=   true   ||   Fungsi.Coprime(dp[1],
dp[2]) != true)
                {

                    dp.Clear();
                    goto ulang2;
                }
                if   (Fungsi.Coprime(dp[0],   pc[0])   !=   true   ||
Fungsi.Coprime(dp[1],   pc[1])   !=   true   ||   Fungsi.Coprime(dp[2],
pc[2]) != true)
                {

                    dp.Clear();
                    goto ulang2;
                }
                BigInteger crtnya, t;


                    Fungsi.Solve(pc[0], pc[1], pc[2], dp[0], dp[1],
dp[2], out crtnya, out t);
                    if (TxtKosong.Text == "")
                    {
                        dp.Clear();
                        l_P.Clear();
                        pc.Clear();
                        TxtKosong.Text = "-";
                        goto ulang;
                    }
                    else {
                        BigInteger N = l_P.Aggregate((a, x) => a *
x);
                        TxtN.Text = N.ToString();


                        BigInteger tot = pc[0] * pc[1] * pc[2];
                        BigInteger invr = Fungsi.in_totient(crtnya,
tot); //waluswatlu 100 000 000 000 000
                        TxtxE.Text = invr.ToString();

                        foreach (Int64 pr in l_P)
                        {
                            txtpg.Text = txtpg.Text + pr.ToString()
+ " ";
                        }



                        foreach (int prud in dp)
                        {
```

A-9

```
                            TxtDP.Text = TxtDP.Text + prud.ToString()
+ " ";
                        }


                        TxtKosong.Text = "-";
                    }
            }
            else
            {
                MessageBox.Show("Maaf, K Harus Angka!");
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            TxtK.Text = "";
            TxtN.Text = "";
            TxtxE.Text = "";
            txtpg.Text = "";
            TxtDP.Text = "";

        }

        private void TxtSimpanK_Click(object sender, EventArgs e)
        {
            string K_N = TxtN.Text;
            string K_E = TxtxE.Text;
            string K_p = txtpg.Text;
            string K_DP = TxtDP.Text;
            if(K_N == "" || K_E == "" || K_p == "" || K_DP == "" ){
                MessageBox.Show("Maaf,    Tidak    ada    data    yang
disimpan.");}
            else{
                string       fileNames       =       "Kunci_"       +
DateTime.Now.ToString("yyyyMMddHHmmss") + ".xml";
                //XmlTextWriter               writer               =
RprimeRSA_RC4A.Simpan.GetWriterForFile(Encoding.UTF8);
                XmlTextWriter                writer                =
RprimeRSA_RC4A.Simpan.GetWriterForFolder(fileNames, Encoding.UTF8);
                if (writer == null)
                {
                    MessageBox.Show("Pilih       Folder       Untuk
Menyimpan!");
                }
                else
                {
                    writer.WriteStartDocument(true);
                    writer.Formatting = Formatting.Indented;
                    writer.Indentation = 2;
                    writer.WriteStartElement("Kunci_RPRIME");
                    createNode(K_N, K_E, K_p, K_DP, writer);
                    writer.WriteEndElement();
                    writer.WriteEndDocument();
                    writer.Close();
                    MessageBox.Show("Data  Kunci  Berhasil  di  simpan
");
                }
            }
        }
```

A-10

```
        private  void  createNode(string  Kunci_N,  string  Kunci_E,
string Kunci_P, string Kunci_DP, XmlTextWriter writer)
        {
            writer.WriteStartElement("Kunci");
            writer.WriteStartElement("Kunci_N");
            writer.WriteString(Kunci_N);
            writer.WriteEndElement();
            writer.WriteStartElement("Kunci_E");
            writer.WriteString(Kunci_E);
            writer.WriteEndElement();
            writer.WriteStartElement("Kunci_P");
            writer.WriteString(Kunci_P);
            writer.WriteEndElement();
            writer.WriteStartElement("Kunci_DP");
            writer.WriteString(Kunci_DP);
            writer.WriteEndElement();
            writer.WriteEndElement();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            FrmMain main = new FrmMain();
            main.Show();
            this.Hide();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)

    }
}
```


3. FORM ENKRIPSI

```
using System;
using Code7248.word_reader;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using ChineseRemainderProblem;
using Microsoft.Office.Interop.Word;
using System.Reflection;
using System.Threading;
using System.Diagnostics;

namespace RprimeRSA_RC4A
{
    public partial class Form_enkripsi : Form
    {
        public byte[] asciiBytes;
        public Form_enkripsi()
        {
            InitializeComponent();
        }

        private Stopwatch stopwatch = new Stopwatch();

        private void button1_Click(object sender, EventArgs e)
```

```
        {
            OpenFileDialog open = new OpenFileDialog();
            open.Filter     =     "Text     File(*.txt)|*.txt|Word
File(*.doc)|*.doc";
            if (open.ShowDialog() == DialogResult.OK)
            {
                string                    metode
open.FileName.Substring(open.FileName.Length - 3, 3);
                textBox1.Text = open.FileName.ToString();
                string temp = "";
                if (metode == "txt")
                {
                    FileStream         fstream         =         new
FileStream(open.FileName, FileMode.Open, FileAccess.ReadWrite);
                    StreamReader sreader = new StreamReader(fstream);
                    sreader.BaseStream.Seek(0, SeekOrigin.Begin);
                    while (sreader.Peek() >= 0)
                    {
                        temp += sreader.ReadLine();
                    }
                    sreader.Close();
                    TxtPl.Text = temp;

                }
                else if  (metode == "doc")
                {
                    TextExtractor       extractor       =       new
TextExtractor(open.FileName);
                    string contents = extractor.ExtractText();
                    contents = extractor.ExtractText();
                    TxtPl.Text = contents;
                }
            }

            LblPj.Text = "Panjang : " + TxtPl.TextLength;
        }


        private  void  button3_Click(object  sender,  EventArgs  e)  //
homr
        {
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {

            LblWaktu.Text = "";

            if (TxtRc.Text == "")
            {
                MessageBox.Show("Key RC4A Harus Di isi!");
                TxtRc.Focus();
            }

            else {
                if (TxtPl.Text == "")
                {
                    MessageBox.Show("Silahkan Masukkan File!");
                }
                else
```

A-12

```
                {
                    stopwatch.Start();

                    if (TxtPl.TextLength <= 256)
                    {
                        string cypheredText = Fungsi.RC4(TxtPl.Te
TxtRc.Text);                                                    A-13
                        asciiBytes
Encoding.ASCII.GetBytes(cypheredText);
                        string x = ConvertStringToHex(cypheredText,
System.Text.Encoding.Unicode);
                        string a = ConvertStringToHex(cypheredText,
System.Text.Encoding.Unicode);
                        TxtCp.Text = a.ToString();

//MessageBox.Show(ConvertHexToString(a.ToString()));
                    }
                    else {
                        MessageBox.Show("Maaf    Karakter    terlalu
panjang..");
                    }
                    stopwatch.Stop();
                    TimeSpan sp = stopwatch.Elapsed;
                    LblWaktu.Text    =    sp.TotalSeconds    +"/s    ,
"+sp.TotalMilliseconds+"/ms";
                    lbcptx.Text = TxtCp.TextLength.ToString();
                }
            };
        }

        public   static   string   ConvertStringToHex(String   input,
System.Text.Encoding encoding)
        {
            Byte[] stringBytes = encoding.GetBytes(input);
            StringBuilder          sbBytes          =          new
StringBuilder(stringBytes.Length * 2);
            foreach (byte b in stringBytes)
            {
                sbBytes.AppendFormat("{0:X2}", b);
            }
            return sbBytes.ToString();
        }

        public static string ConvertHexToString(string hexString)
        {
            var bytes = new byte[hexString.Length / 2];
            for (var i = 0; i < bytes.Length; i++)
            {
                bytes[i] = Convert.ToByte(hexString.Substring(i * 2,
2), 16);
            }

            return Encoding.Unicode.GetString(bytes);
        }


        private void TxtRc_KeyUp(object sender, KeyEventArgs e)
        {
            label7.Text = "Panjang : " +TxtRc.TextLength.ToString();
        }
```

```
        private void TxtRc_TextChanged(object sender, EventArgs e)
        {
            TxtRc2.Text = TxtRc.Text;
        }

        private void button9_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "XML Files (*.xml)|*.xml";
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                string ok = ofd.FileName;
                XmlTextReader reader = new XmlTextReader(ok);
                string sName="";

                while (reader.Read())
                {

                    switch (reader.NodeType)
                    {
                            case XmlNodeType.Element:
                                sName=reader.Name;
                                break;
                            case XmlNodeType.Text:
                            switch(sName)
                            {
                                case "Kunci_N":
                                    lblN.Text = reader.Value;
                                    break;
                                case "Kunci_E":
                                    LblE.Text = reader.Value;
                                    break;

                            }break;
                    }

                }
            }
        }

        private void button8_Click(object sender, EventArgs e)
        {
            if (TxtRc2.Text == "")
            {
                MessageBox.Show("Plain Key tidak boleh kosong!");
            }
            if (lblN.Text == "")
            {
                MessageBox.Show("Silahkan Import Kunci Public!");
            }
            if (TxtRc2.Text != "" && lblN.Text != "" && LblE.Text !=
"")
            {
                stopwatch.Start();
                asciiBytes = Encoding.ASCII.GetBytes(TxtRc2.Text);
                TxtCRc.Text = "";
                Int64 _N = Int64.Parse(lblN.Text);
                Int64 _E = Int64.Parse(LblE.Text);
                foreach (var ias in asciiBytes)
                {
```

A-14

```
                    TxtCRc.Text   +=   Fungsi.eksponensial(ias,   _E,
_N).ToString() + " ";
                }
                stopwatch.Stop();
                TimeSpan sp = stopwatch.Elapsed;
                LblWaktu.Text   =   sp.TotalSeconds   +   "/s   ,   "   +
sp.TotalMilliseconds + "/ms";
            }                                                          A-15
            label4.Text = TxtCRc.TextLength. ToString();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            if (TxtCRc.Text == "" || TxtRc2.Text == "")
            {
                MessageBox.Show("Maaf Data tidak boleh kosong!");
            }
            else
            {
                string Cipherkey = TxtCRc.Text;

                string      fileNames      =      "chiperkey"      +
DateTime.Now.ToString("yyyyMMddHHmmss") + ".xml";
                XmlTextWriter                writer                =
RprimeRSA_RC4A.Simpan.GetWriterForFolder(fileNames, Encoding.UTF8);
                if (writer == null)
                {
                    MessageBox.Show("Pilih       Folder       Untuk
Menyimpan!");
                }
                else
                {
                    writer.WriteStartDocument(true);
                    writer.Formatting = Formatting.Indented;
                    writer.Indentation = 2;
                    writer.WriteStartElement("Kunci_RPRIME");
                    createNode(Cipherkey, writer);
                    writer.WriteEndElement();
                    writer.WriteEndDocument();
                    writer.Close();
                    MessageBox.Show("Data  Kunci  Berhasil  di  simpan
");
                }
            }
        }

        private  void  createNode(string  Cipherkey,  XmlTextWriter
writer)
        {
            writer.WriteStartElement("cipher");
            writer.WriteStartElement("Cipherkey");
            writer.WriteString(Cipherkey);
            writer.WriteEndElement();
            writer.WriteEndElement();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            LblWaktu.Text = "";
            TxtRc2.Text = "";
            TxtCRc.Text = "";
```

```
            lblN.Text = "";
            LblE.Text = "";
        }

        private void button5_Click_1(object sender, EventArgs e)
        {
            if (TxtPl.Text == "" && TxtCp.Text == "")
            {
                MessageBox.Show("Maaf,  Data  tidak  ada  untuk  …
simpan!");
            }
            else
            {


                Microsoft.Office.Interop.Word.Application winword =
new Microsoft.Office.Interop.Word.Application();
                object missing = System.Reflection.Missing.Value;
                Microsoft.Office.Interop.Word.Document  document  =
winword.Documents.Add(ref  missing,  ref  missing,  ref  missing,  ref
missing);

                document.Content.SetRange(0, 0);
                document.Content.Text      =      TxtCp.Text      +
Environment.NewLine;
                winword.Visible = false;

                document.Close(ref   missing,   ref   missing,   ref
missing);
                MessageBox.Show("File Berhasil di simpan!");
            }
        }

        private    void    createNodeCipherText(string    ciphertext,
XmlTextWriter writer)
        {
            writer.WriteStartElement("cipher");
            writer.WriteStartElement("ciphertext");
            writer.WriteString(ciphertext);
            writer.WriteEndElement();
            writer.WriteEndElement();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            FrmMain fm = new FrmMain();
            fm.Show();
            this.Hide();
        }

        private void TxtPl_TextChanged(object sender, EventArgs e)
        {
            LblPj.Text = "Panjang : " + TxtPl.TextLength;
        }

        private void TxtCRc_TextChanged(object sender, EventArgs e)
        {
            label4.Text = TxtCRc.TextLength.ToString();
        }
```

```
    }
}
```

4. Form dekripsi

A-17

```
using System;
using Code7248.word_reader;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using ChineseRemainderProblem;
using Microsoft.Office.Interop.Word;
using System.Reflection;
using System.Threading;
using System.Diagnostics;
using System.Numerics;

namespace RprimeRSA_RC4A
{
    public partial class Dekripsi : Form
    {
        public Dekripsi()
        {
            InitializeComponent();
        }

        private Stopwatch stopwatch = new Stopwatch();

        private void button8_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "XML Files (*.xml)|*.xml";
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                string ok = ofd.FileName;
                XmlTextReader reader = new XmlTextReader(ok);
                string sName = "";

                while (reader.Read())
                {

                    switch (reader.NodeType)
                    {
                        case XmlNodeType.Element:
                            sName = reader.Name;
                            break;
                        case XmlNodeType.Text:
                            switch (sName)
                            {
                                case "Kunci_P":
                                    lblP.Text = reader.Value;
                                    break;
                                case "Kunci_DP":
                                    LblDP.Text = reader.Value;
                                    break;
                            } break;
                    }
```

```
        }
    }
}

private void button7_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.Filter = "XML Files (*.xml)|*.xml";
    if (ofd.ShowDialog() == DialogResult.OK)
    {
        string ok = ofd.FileName;
        XmlTextReader reader = new XmlTextReader(ok);
        string sName = "";

        while (reader.Read())
        {

            switch (reader.NodeType)
            {
                case XmlNodeType.Element:
                    sName = reader.Name;
                    break;
                case XmlNodeType.Text:
                    switch (sName)
                    {
                        case "Cipherkey":
                            TxtCK.Text = reader.Value;
                            break;

                    } break;
            }

        }
    }
}

private void button6_Click(object sender, EventArgs e)
{
    if (TxtCK.Text == "" || lblP.Text == "")
    {
        MessageBox.Show("Maaf Data tidak boleh Kosong");
    }
    else
    {
        stopwatch.Start();
        List<Int64> _P = new List<Int64>();
        List<Int64> _DP = new List<Int64>();
        List<Int64> _C = new List<Int64>();
        Int64 M1, M2, M3;

        string s_p = lblP.Text.Trim();
        string s_DP = LblDP.Text.Trim();
        string[] t_P = s_p.Split(' ');
        string[] t_DP = s_DP.Split(' ');

        string s_CPK = TxtCK.Text.Trim();
        string[] t_CPK = s_CPK.Split(' ');



        for (int x = 0; x < t_P.Length; x++)
```

A-18

```
                {
                    _P.Add(Int64.Parse(t_P[x]));
                }
                for (int x = 0; x < t_DP.Length; x++)
                {
                    _DP.Add(Int64.Parse(t_DP[x]));
                }
                for (int x = 0; x < t_CPK.Length; x++)
                {
                    _C.Add(Int64.Parse(t_CPK[x]));
                }
                BigInteger CRT, T;
                foreach (var a in _C)
                {
                    M1 = Fungsi.eksponensial(a, _DP[0], _P[0]);
                    M2 = Fungsi.eksponensial(a, _DP[1], _P[1]);
                    M3 = Fungsi.eksponensial(a, _DP[2], _P[2]);

                    Fungsi.Solve(_P[0], _P[1], _P[2], M1, M2, M3, out
CRT, out T);
                    char character = (char)CRT;
                    TxtPlainKey.Text = TxtPlainKey.Text + character;
                }
            }
            stopwatch.Stop();
            TimeSpan sp = stopwatch.Elapsed;
            LblWaktu.Text   =   sp.TotalSeconds   +   "/s   ,   "   +
sp.TotalMilliseconds + "/ms";
        }

        private void button4_Click(object sender, EventArgs e)
        {
            if (TxtPlainKey.Text == "" && TxtCK.Text == "")
            {
                MessageBox.Show("Maaf Data tidak boleh kosong!");
            }
            else
            {
                string Pl = TxtPlainKey.Text;
                string      fileNames      =      "PlainKey_"      +
DateTime.Now.ToString("yyyyMMddHHmmss") + ".xml";
                XmlTextWriter                writer                =
RprimeRSA_RC4A.Simpan.GetWriterForFolder(fileNames, Encoding.UTF8);
                if (writer == null)
                {
                    MessageBox.Show("Pilih       Folder       Untuk
Menyimpan!");
                }
                else
                {
                    writer.WriteStartDocument(true);
                    writer.Formatting = Formatting.Indented;
                    writer.Indentation = 2;
                    writer.WriteStartElement("PlainKey_Kunci");
                    createNode(Pl, writer);
                    writer.WriteEndElement();
                    writer.WriteEndDocument();
                    writer.Close();
                    MessageBox.Show("Data  Kunci  Berhasil  di  simpan
");
                }
```

A-19

```
        }
    }

    private void createNode(string P_Key, XmlTextWriter writer)
    {
        writer.WriteStartElement("plain");
        writer.WriteStartElement("plainKey");
        writer.WriteString(P_Key);
        writer.WriteEndElement();
        writer.WriteEndElement();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.Filter = "XML Files (*.xml)|*.xml";
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            string ok = ofd.FileName;
            XmlTextReader reader = new XmlTextReader(ok);
            string sName = "";

            while (reader.Read())
            {

                switch (reader.NodeType)
                {
                    case XmlNodeType.Element:
                        sName = reader.Name;
                        break;
                    case XmlNodeType.Text:
                        switch (sName)
                        {
                            case "plainKey":
                                TxtKunci.Text = reader.Value;
                                break;

                        } break;
                }

            }
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
         OpenFileDialog open = new OpenFileDialog();
        open.Filter      =      "Text      File(*.txt)|*.txt|Word
File(*.doc)|*.doc";
        if (open.ShowDialog() == DialogResult.OK)
        {
            string                    metode                    =
open.FileName.Substring(open.FileName.Length - 3, 3);

            string temp = "";
            if (metode == "txt")
            {
                FileStream         fstream         =         new
FileStream(open.FileName, FileMode.Open, FileAccess.ReadWrite);
                StreamReader sreader = new StreamReader(fstream);
                sreader.BaseStream.Seek(0, SeekOrigin.Begin);
```

A-20

```
                    while (sreader.Peek() >= 0)
                    {
                        temp += sreader.ReadLine();
                    }
                    sreader.Close();
                    TxtCipher.Text = temp;
```

A-21

```
                }
                else if (metode == "doc")
                {
                    TextExtractor       extractor       =       new
TextExtractor(open.FileName);
                    string contents = extractor.ExtractText();
                    contents = extractor.ExtractText();
                    TxtCipher.Text = contents;
                }
            }
            label5.Text = TxtCipher.TextLength.ToString();

        }

        public static string ConvertHexToString(string hexString)
        {
            var bytes = new byte[hexString.Length / 2];
            for (var i = 0; i < bytes.Length; i++)
            {
                bytes[i] = Convert.ToByte(hexString.Substring(i * 2,
2), 16);
            }

            return Encoding.Unicode.GetString(bytes);
        }

        private void btndekripsi_Click(object sender, EventArgs e)
        {
            stopwatch.Start();
            if (TxtKunci.Text == "" || TxtCipher.Text == "")
            {
                MessageBox.Show("Maaf Data tidak boleh Kosong!");
            }
            else
            {
                string testString = TxtCipher.Text.Trim();
                string hasil = ConvertHexToString(testString);
                string PlainText = Fungsi.RC4(hasil, TxtKunci.Text);
                textBox4.Text = PlainText;

            }
            stopwatch.Stop();
            TimeSpan sp = stopwatch.Elapsed;
            LblWaktu.Text   =   sp.TotalSeconds   +   "/s   ,   "   +
sp.TotalMilliseconds + "/ms";

        }

        private void button3_Click(object sender, EventArgs e)
        {
            FrmMain f = new FrmMain();
            f.Show();
            this.Hide();
        }
```

A-22

```
        private void button5_Click(object sender, EventArgs e)
        {
            LblWaktu.Text = "";
            TxtCK.Text = "";
            TxtPlainKey.Text = "";
        }

        private void TxtPlainKey_TextChanged(object sender, EventArgs
e)
        {
            LblPanjang.Text = TxtPlainKey.TextLength.ToString();
        }

        private void TxtCipher_TextChanged(object sender, EventArgs
e)
        {
            LblPanjangCp.Text = TxtCipher.TextLength.ToString();
        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {
            LblPanjangPT.Text = textBox4.TextLength.ToString();
        }
    }
}
```

B-1

CURRICULUM VITAE

(DAFTAR RIWAYAT HIDUP)

==========

Data Pribadi

Nama	: Nikmah Hanum
Tempat, Tanggal Lahir	: Medan, 23 November 1995
Jenis Kelamin	: Perempuan
Umur	: 21 Tahun
Alamat	: Jl. Jamin Ginting Gg. Haji Arif No. 10 Padang Bulan Medan
Agama	: Islam
No. HP	: 081262756500
Email	: Nikmah23hanum@gmail.com

Pendidikan Formal

2001 – 2007 : SD Negeri 149561 Hutabaringin
2007 – 2010 : MTsN Panyabungan
2010 – 2013 : MAN Panyabungan

Keahlian

Pemrograman	: C#, C++.
Database	: MySql
Design	: Corel Draw

Pendidikan Non-Formal dan Seminar

2014 : Seminar Nasional Literasi Informasi (SENARAI)
2016 : Global Inspiring of Indonesia Next Program
2017 : Private TOEFL

Pengalaman Organisasi

2014 : Anggota Kaderisasi UKMI AL-KHUWARIZMI
2014 : Anggota Konsumsi IMILKOM GOTO SCHOOL
2015 : Anggota LSO-IT UKMI AL-KHUWARIZMI
2015 : Anggota Kesehatan PORSENI IMILKOM