BAB 3

ANALISIS DAN PERANCANGAN

3.1.Analisis Sistem

Tahap pertama dalam melakukan perancangan sistem adalah analisis sistem. Tujuan dari analisis sistem adalah untuk menganalisis persoalan-persoalan yang akan muncul dalam perancangan sistem dan sebagai gambaran apa yang harus dilakukan pada saat proses perancangan sistem. Analisis sistem menjadi fondasi yang menentukan keberhasilan sistem yang akan dibangun. Jadi, jika terjadi kesalahan pada tahap analisis ini maka akan menyebabkan kesalahan pada tahap-tahap selanjutnya. Terdapat beberapa tahapan dalam tahap analisis sistem yaitu analisis malasah dan analisis kebutuhan.

3.1.1. Analisis Masalah

Adapun tujuan analisis masalah yaitu untuk memahami kelayakan masalah dan untuk mengetahui penyebab masalah yang telah dirumuskan sebelumnya pada rumusan masalah dan pembahasan sebelumnya. Masalah yang akan diselesaikan oleh sistem ini adalah proses penyandian pesan teks oleh pengirim pesan kepada penerima pesan sehingga tidak semua orang dapat mengetahui isi pesan tersebut.

Masalah yang diperoleh dalam penelitian ini digambarkan dengan menggunakan diagram Ishikawa. Whitten & Bentley (2007) menyatakan Diagram Ishikawa atau biasa disebut dengan *Fishbone Diagram* (Diagram Tulang Ikan) bertujuan untuk mengidentifikasi, menganalisa dan menggambarkan semua penyebab-penyebab yang berhubungan dengan suatu masalah yang akan dipecahkan.

Adapun struktur diagram Ishikawa terdiri dari (Whitten & Bentley, 2007) :

1. Kepala ikan (*fish's head*), menunjukkan nama atau judul dari masalah yang diidentifikasi.
2. Tulang-tulang ikan (*fish's bones*), menunjukkan penyebab-penyebab masalah yang diperoleh.

Masalah dalam penelitian ini dapat dilihat pada Gambar 3.1 yang dirancang dalam bentuk diagram Ishikawa atau *Fishbone Diagram*.

Gambar 3.1 *Fishbone Diagram* Masalah Penelitian

Seperti yang terlihat pada gambar 3.1, masalah utama pada segi empat yang berposisi paling kanan (bagian kelapa ikan) yaitu Pengamanan TeksDengan *Hybrid Cryptosystem* Algoritma *Multi-Power RSA* dan Algoritma *Blowfish*. Kategori penyebab masalah dari penelitian ini masing-masing terdiri dari Material, Metode, Sistem dan *User*. Detail penyebab masalah ditunjukkan dengan tanda panah yang mengarah ke masing-masing kategori. Manfaat dari penggunaan metode *hybrid cryptosystem* pada kriptografi, teksakan dienkripsi dan didekripsi dengan menggunakan algoritma simetris dank unci dai algoritma simetris tersebut akan dienkripsi dan didekripsi dengan menggunakan algoritma asimetris.

3.1.2. Analisis Kebutuhan

Tahap kedua setelah mengidentifikasi penyebab masalah penelitian yaitu tahap analisis kebutuhan yang mempunyai tujuan yaitu mengumpulkan kebutuhan atau informasi yang harus dimiliki oleh sistem. Dalam analisis kebutuhan akan dilakukan analisis sistem secara fungsional dan analisis sistem secara nonfungsional (Pujianto, 2012)

a. Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan hal-hal apa saja yang akan dikerjakan oleh sistem untuk tercapainya tujuan dari sistem tersebut (Whitten & Bentley, 2007). Kebutuhan fungsional yang harus terpenuhi dari sistem yang mengimplementasikan algoritma *Multi-Power RSA* dan algoritma *Blowfish* dalam *hybrid cryptosystem* untuk pengamanan teks adalah sebagai berikut :

1. Sistem dapat membangkitkan kunci public (*private key*) dan kunci pribadi (*private key*) secara acak dengan menguji bilangan prima sebelumnya.
2. Sistem dapat mencari dan meng-*uploadfile* teks yang berekstensi .txt, .doc dan .docx yang tersimpan pada perangkat yang digunakan atau sistem dapat menerima input *plaintext* dari pengguna secara manual.
3. Sistem dapat mengenkripsiteks menggunakan algoritma simetris.
4. Sistem dapat mengenkripsi kunci algoritma simetris dengan menggunakan algoritma asimetris.
5. Sistem dapat menyimpan pesan dan kunci simetrisyang telah dienkripsi (*ciphertext* dan *cipherkey*). Hasil dari enkripsi dapat digunakan untuk mendekripsi pesan.
6. Sistem dapat mendekripsi kunci simetris (*cipherkey*) dengan menggunakan kunci pribadi (*private* key) algoritma asimetris. Hasil dari dekripsi tersebut adalah kunci algoritma simetris.
7. Sistem dapat mendekripsi pesan (*ciphertext*) menggunakan kunci algoritma simetris yang telah didapatkan sebelumnya.
8. Sistem dapat menyimpan pesan yang telah didekripsi dengan algoritma yang digunakan.

b. Kebutuhan Nonfungsional

Kebutuhan nonfungsional menggambarkan fitur, karakteristik, batasan sistem, performa, dokumentasi dan lainnya yang dibutuhkan oleh sistem (Whitten & Bentley, 2007). Untuk mendukung kinerja sistem, maka kebutuhan nonfungsionaldari sistem ini adalah sebagai berikut :

1. *User friendly*

   Sistem yang akan dibangun menggunakan desain yang responsive danmudah digunakan sehingga dapat dioperasikan oleh user.

2. Dokumentasi

   Sistem yang akan dibangun memiliki panduan penggunaan sistem yang akan mempermudah *user* dalam menggunakannya.

3. Performa

   Sistem yang akan dibangun memiliki waktu yang relatif singkat untuk proses enkripsi dan dekripsi.

4. Kemampuan

   Sistem yang akan dibangun dapat melakukan fungsi kriptografi yaitu enkripsi dan dekripsi.

5. *Control*

   Sistem yang dibangun akan menampilkan pesan *error* untuk setiap inputan yang tidak sesuai.

3.1.3. Arsitektur Umum Sistem

Arsitektur umum sistem dapat digunakan untuk menggambarkan jalannya sistem secara keseluruhan. Arsitektur umum sistem ini juga dapat menjadi acuan untuk pembuatan pemodelan sistem. Arsitektur umum sistem dapat dilihat pada gambar 3.2.

Gambar 3.2 Arsitekur Umum Sistem

Adapun penjelasan dari Gambar 3.2 yaitu :

Langkah pertama, yang menerima pesan yaitu Gwen membangkitkan *public key* dan *private key*, yang kemudian *public key* tersebut diberikan kepada pihak pengirim pesan, yaitu Mahadi. Setelah menerima *public key Multi-Power RSA*, pesan yang akan dikirim Mahadi akan dienkripsi dengan algoritma *Blowfish*menggunakan kunci simetris, maka didapatlah *ciphertext* dari pesan yang akan dikirim. Kemudian kunci simetris juga dienkripsi dengan algoritma *Multi-Power RSA*menggunakan *public key* yang sudah dibangkitkan sebelumnya, maka didapatlah *cipherkey* kunci simetris.Pada proses enkripsi, Mahadi akan mempunyai *public key Multi-Power RSA, ciphertext*, dan *cipherkey*. Selanjutnya pada proses dekripsi, Gwen memperoleh *ciphertext* dan*cipherkey* dari Mahadi. Selanjutnya, Gwen mendekripsi *cipherkey* dengan algoritma *Multi_power RSA* menggunakan *private key* yang sebelumnya telah dibangkitkan, maka didapatlah kunci simetrisnya.Setelah didapatkan kembali kunci simetrisnya, langkah selanjutnya akan *ciphertext* akan didekripsi dengan algoritma *Blowfish* menggunakan kunci simetris, maka didapatlah *plaintext* yang merupakan pesan yang dikirim mahadi ke Gwen.

3.1.4. Pemodelan Sistem

Pemodelan sistem bertujuan untuk memperoleh gambaran kerja dari sistem yang akan dibangun. Pemodelan sistem ini akan dibuat dengan menggunakan diagram UML (*Unified Modelling Language*). Diagram UML yang digunakan yaitu *Use Case Diagram, Activity Diagram* dan *Sequence Diagram*.

a. Use Case Diagram

Use Case Diagram bertujuan untuk menggambarkan kebutuhan sistem secara fungsional dengan mengidentifikasi siapa saja aktor yang berhubungan dengan sistem dan apa saja yang dapat dilakukan sistem. Pada penelitian ini *use case diagram* dapat ditunjukkan pada gambar 3.3.

Gambar 3.3 *Use Case Diagram* Penelitian

Gambar 3.3 menjelaskan bahwa sistem diakses oleh dua aktor yang dinamakan Pengirim dan Penerima. Aktor Penerima memiliki hak akses terhadap *use case* utama pada sistem. Penerima memiliki hak akses *use case* enkripsi untuk mengenkripsi pesan dan Pengirim memiliki hak akses *use case* pembangkit kunci algoritma asimetris dan dekripsi untuk mendapatkan pesan kembali.

b. Activity Diagram

Activity diagram adalah diagam yang menggambarkan aliran kerja atau aktifitas pada sistem yang sedang dibangun, menjelaskan dimana aktifitas mulai dan berhenti serta *decision* yang mungkin terjadi. *Activity diagram* juga menggambarkan interaksi aktifitas antara *user* dengan sistem secara berurutan. Terdapat tiga *activity diagram* pada sistem ini yaitu *activity diagram* pembangkit kunci, *activity diagram* proses enkripsi dan *acitivy diagram* proses dekripsi.

- *Activity Diagram* Pembangkit Kunci

Gambar 3.4 *Activity Diagram* Pembangkit Kunci

Pada gambar 3.4 kotak paling kiri menunjukkan aktifitas *user* sedangkan kotak paling kanan menunjukkan respon oleh sistem yang dikerjakan. Pertama, *user* harus menginputkan sebuah bilangan prima maksimal 8 bit, kemudian sistem akan memeriksa apakah *p* dan *q* adalah bilangan prima atau tidak. Jika bilangan tersebut komposit, *user* harus menginput ulang bilangan tersebut. Jika bilangan prima, maka sistem akan memproses nilai *N* dengan menekan *button* hitung dan menampilkannya. Lalu, *user* harus mengecek nilai *e* dengan syarat [gdc((p-1)*(q-1).e=1]. *User* dapat menghitung dan menampilkan nilai *d*, r_1, dan $r_2$dengan menekan *button* hitung.

- *Activity Diagram* Proses Enkripsi

Gambar 3.5 *Activity Diagram* Proses Enkripsi

Gambar 3.5 menunjukkan *activity diagram* proses enkripsi sistem, terlihat bahwa sistem dapat melakukan enkripsi pesan dengan *upload file* atau input pesan secara manual. Apabila proses enkripsi dilakukan dengan *upload file*, maka sistem akan menampilkan *open file dialog* untuk mencari *file* yang ingin dienkripsi. Lalu, sistem akan menampilkan isi dari *file* tersebut. Kemudian *user* harus menginputkan kunci simetris algoritma *Blowfish* untuk dapat melakukan proses enkripsi pesan. Setelah melakukan enkripsi pesan, sistem akan menampilkan *ciphertext.* Proses enkripsi pada sistem ini tidak hanya dilakukan pada pesan saja tetapi dilakukan juga proses enkripsi pada kunci.

Untuk melakukan proses enkripsi pada kunci, maka *user* harus menampilkan kembali kunci simetris yang telah diinputkan sebelumnya dengan menekan tombol tampil. Lalu, *user* harus menginputkan kunci public algoritma *Multi-Power RSA.* Proses enkripsi kunci dapat dilakukan setelah kunci simetris dan kunci public diinputkan. Hasil dari enkripsi kunci adalah *cipherkey*. Apabila *user*ingin menyimpan hasil enkripsi, sistem dapat menyimpannya untuk dapat digunakan pada proses dekripsi.

- *Activity Diagram* Proses Dekripsi

Gambar 3.6 *Activity Diagram* Proses Dekripsi

Proses dekripsi pesan ditunjukkan pada gambar 3.5. Proses dekripsi dilakukan dengan mencari *file* yang telah disimpan dan dienkripsi sebelumnya. *User* akan meng-*upload file* kemudian sistem akan menampilkan *open file dialog* untuk mencari *file* yang akan didekripsi. Lalu, sistem akan menampilkan isi *file* tersebut. Sebelum melakukan dekripsi pesan *user* harus melakukan dekripsi *cipherkey* terlebih dahulu. *User* harus menginputkan kunci pribadi atau *private key* algoritma *Multi-Power RSA*. Proses dekripsi kunci simetris dapat berjalan ketika *cipherkey* dan kunci pribadi telah terinput dengan menghasilkan kunci simetris.

Untuk melakukan proses dekripsi pesan atau *ciphertext*, *user* harus menampilkan kunci simetris yang telah didekripsi sebelumnya. Kemudian, proses dekripsi *ciphertext* dapat berjalan ketika *user* menakan tombol dekripsi. Hasil dari proses dekripsi tersebut adalah pesan awal atau *plaintext* awal dan sistem akan menampilkannya. *User* dapat melihat *plaintext* awal hasil dari proses dekripsi dan dapat menyimpannya.

c. Sequence Diagram

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar objek pada sistem dengan proses terkait dalam rangkaian waktu tertentu. *Sequence diagram* juga membantu untuk menggambarkan data yang masuk dan keluar dari sistem. *Sequence diagram* untuk sistem ini terdiri dari *sequence diagram* pembangkit kunci, *sequence diagram* proses enkripsi dan *sequence diagram* proses dekripsi.

- *Sequence Diagram* Pembangkit Kunci

Gambar 3.7 *Sequence Diagram* Pembangkit Kunci

Pada gambar 3.7 terlihat *sequence diagram* pembangkit kunci antara *user* sebagai aktor dengan sistem yang dibangun. Arti dari tanda panah garis penuh adalah aktifitas yang terjadi ketika *user* atau objek berinteraksi dengan objek lain, sedangkan arti dari garis putus-putus adalah respon sistem terhadap *user*.

- *Sequence Diagram* Proses Enkripsi

Gambar 3.8 *Sequence Diagram* Proses Enkripsi

Proses enkripsi terbagi menjadi enkripsi pesan teks dan enkripsi kunci simetris. Pada gambar 3.8 aktor atau objek saling berinteraksi dengan objek lain dalam kedua proses enkripsi tersebut, yaitu enkripsi teksmaupun enkripsi kunci simetris. Hasil dari proses enkripsi berupa *ciphertext* dan *cipherkey* yang dapat disimpan untuk digunakan pada proses selanjutnya yaitu proses dekripsi.

- *Sequence Diagram* Proses Dekripsi

Gambar 3.9 *Sequence Diagram* Proses Dekripsi

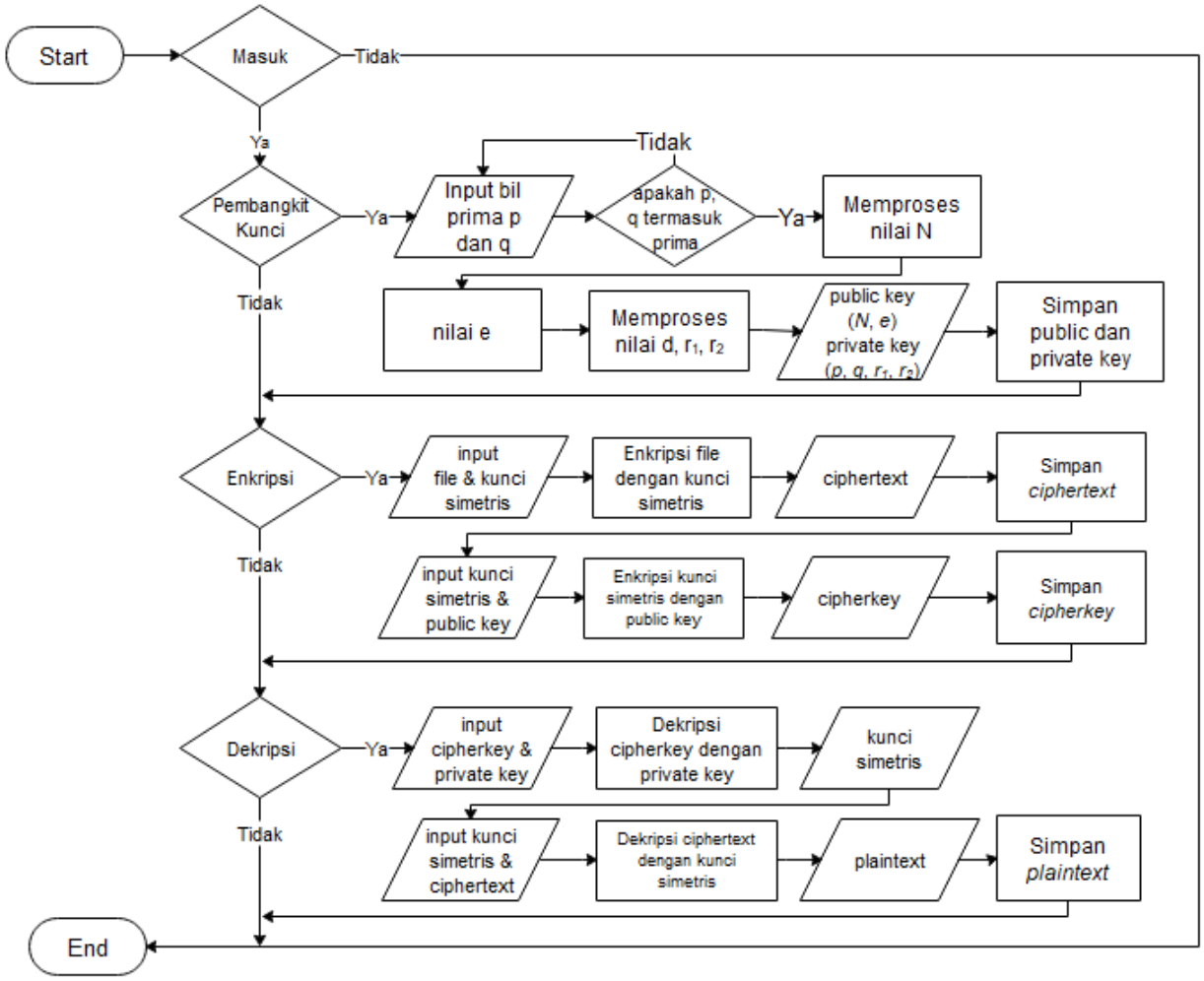
Proses dekripsi terbagi menjadi dekripsi kunci simetris dan dekripsi pesan teks. Pada gambar 3.9 aktor atau objek saling berinteraksi dengan objek lain dalam kedua proses dekripsi tersebut, yaitu dekripsi kunci simetris maupun dekripsi teks. Hasil dari proses dekripsi berupa kunci simetrisdan pesan awal atau *plaintext* awal kembaliyang juga dapat disimpan.

3.1.5. Flowchart

Flowchart merupakan diagram atau bagan yang menggambarkan aliran data atau urutan langkah-langkah di dalam sistem secara logika. Ada beberapa *flowchart* yang akan dibuat dalam perancangan sistem ini, yaitu: *flowchart sistem*, *flowchart* algoritma *Multi-Power RSA* dan *flowchart* algoritma *Blowfish*.

a. *Flowchart* Sistem

Flowchart sistem menunjukkan aliran data atau urutan langkah-langkah yang ada di dalam sistem secara sistematis yang ditampilkan pada gambar 3.10. Secara umum, sistem memiliki tiga halaman utama yang dapat dipilih oleh *user*, yaitu halaman pembangkit kunci, enkripsi, dan dekripsi.



Gambar 3.10 *Flowchart* Sistem

b. *Flowchart* Pengujian Bilangan Prima Algoritma *Agrawal Kayal Saxena (AKS)*

Flowchart algoritma *AKS* bertujuan untuk untuk menguji ataupun mengecek apakah bilang yang diinputkan termasuk bilangan prima atau bilangan komposit. *Flowchart* pengujian bilangan prima algoritma *AKS* digambarkan pada gambar 3.11.

Gambar 3.11*Flowchart* Pengujian Bilangan Prima Algoritma *AKS*

c. *Flowchart* Algoritma *Multi-Power RSA*

Flowchart algoritma Multi-Power RSA terdiri dari tiga buah *flowchart* yaitu: *flowchart* pembangkit kunci algoritma *Multi-Power RSA*, *flowchart* enkripsi algoritma *Multi-Power RSA* dan *flowchart* dekripsi algoritma *Multi-Power RSA*. *Flowchart* pembangkit kunci algoritma *Multi-Power RSA* digambarkan pada gambar 3.12.

Gambar 3.12*Flowchart* Pembangkit Kunci Algoritma *Multi-Power RSA*

Gambar 3.12 menunjukkan *flowchart* pembangkit kunci algoritma *Multi-Power* RSA. Pembangkitan kunci dimulai dari menginputkan 2 buah bilangan prima p dan q. Apabila bilangan tersebut prima maka nilai N akan diproses, begitu juga dengan nilai e. Nilai d, r_1 dan r_2 dapat ditampilkan setelah nilai sebelumnya didapatkan. Hasil akhirnya dari proses pembangkitan kunci adalah *public key* dan *private* key.

Gambar 3.13*Flowchart* Enkripsi Algoritma *Multi-Power RSA*

Gambar 3.13 menunjukkan *flowchart* enkripsi algoritma *Multi-Power RSA*. Proses enkripsi dimulai dengan menginputkan *plaintext* dan *public key*. Dimana dari proses enkripsi tersebut akan menghasilkan *ciphertext*.

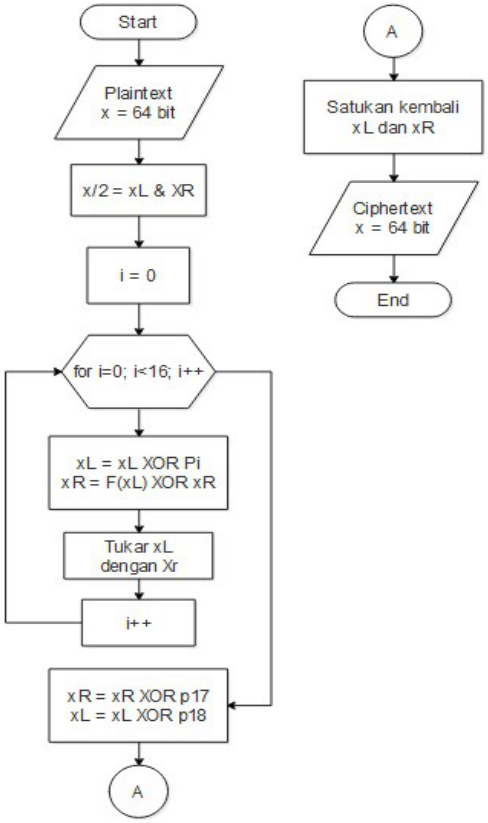
Untuk mendapatkan *plaintext* kembali maka dilakukan proses dekripsi *ciphertext*. *Flowchart* dekripsi algoritma *Multi-Power RSA* ditunjukkan pada gambar 3.14.

Gambar 3.14*Flowchart* Dekripsi Algoritma *Multi-Power RSA*

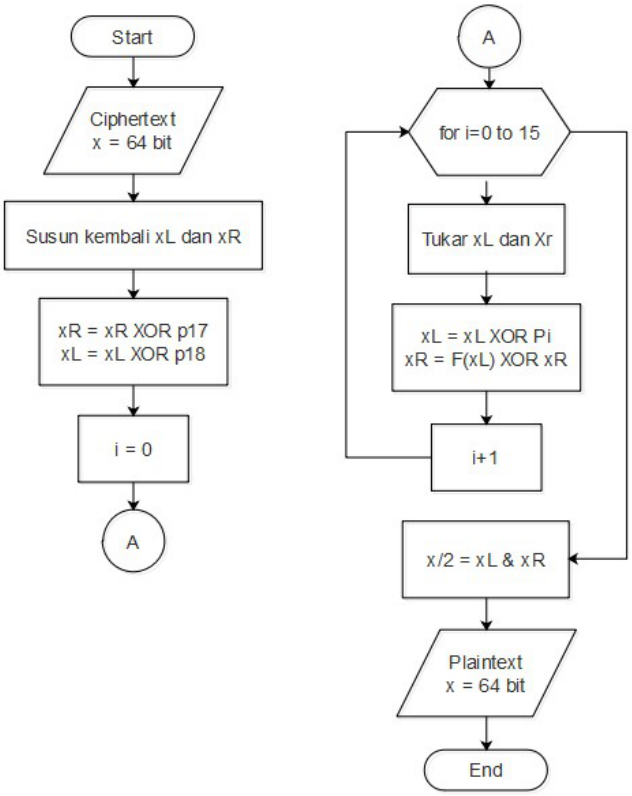
Gambar 3.14 menunjukkan *flowchart* dekripsi algoritma *Multi-Power RSA*. Proses dekripsi dimulai dengan input *ciphertext* dan *private key* yang hanya diketahui oleh penerima pesan. Hasil akhir dari proses dekripsi akan menghasilkan *plaintext* awal.

d. *Flowchart* Algoritma *Blowfish*

Flowchart algoritma *blowfish* terbagi menjadi dua yaitu *flowchart* enkripsi dan *flowchart* dekripsi. *Flowchart* enkripsi algoritma *blowfish* ditunjukkan pada gambar 3.15 dan *flowchart* dekripsi algoritma *blowfish* ditunjukkan pada gambar 3.16.



Gambar 3.15*Flowchart* Enkripsi Algoritma *Blowfish*



Gambar 3.16*Flowchart* Dekripsi Algoritma *Blowfish*

3.2. Perancangan*Interface*

Sistem akan dibangun dengan menggunakan bahasa pemrograman C# dengan menggunakan *software SharpDevelop*. Rancangan *interface* akan dibangun sesuai dengan kebutuhan dan *software* yang digunakan. Rancangan *interface* atau tampilan awal ini bertujuan untuk mempermudah dalam pembangunan sistem. Sistem ini menggunakan empay halaman rancangan *interface*, yaitu halaman utama atau halaman awal, halaman pembangkit kunci, halaman enkripsi dan halaman dekripsi.

a. *Interface* Halaman Utama

Pada halaman ini berisikan tentang judul sistem yang dibangun, logo universitas, identitas yang membangun sistem dan sebuah *button*. *User* yang ingin masuk dan memulai sistem harus melakukan aksi dengan menekan *button* tersebut. Rancangan *interface* halaman utama ditunjukkan pada gambar 3.17.

Gambar 3.17 Rancangan *Interface* Halaman Utama

Keterangan gambar:

1. *Label* (judul) digunakan untuk judul sistem yang dibangun.
2. *Picturebox* digunakan untuk logo Universitas Sumatera Utara.

3. *Label* (identitas) digunakan untuk identitas nama dan nim pembangun sistem.
4. *Button* digunakan untuk masuk dan memulai sistem.

b. *Interface* Halaman Pembangkit Kunci

Untuk memulai proses algoritma kriptografi maka dibangun halaman pembangkit kunci. Halaman ini berfungsi untuk membangkitkan kunci public (*public key*) dan kunci pribadi (*private key*) untuk melakukan proses enkripsi dan proses dekripsi. Rancangan *interface* halaman pembangkit kunci terlihat pada gambar 3.18.

Gambar 3.18 Rancangan *Interface* Halaman Pembangkit Kunci

Keterangan gambar:

1. *TabControl*berfungsi untuk memilih proses yang diinginkan, tab ini terdiri dari 4 proses yang dipilih yaitu ‘Pembangkit Kunci’, ‘Enkripsi’, ‘Dekripsi’, dan ‘Penggunaan Aplikasi’.
2. *GroupBox* untuk menggabungkan beberapa komponen proses pembangkit kunci.
3. *Label* digunakan untuk judul ‘Input Bilangan Prima (1-50)’.
4. *Label* digunakan untuk judul ‘Bilangan Prima (p)’.

5. *Textboxt* berfungsi untuk menginput nilai *p*.
6. *Button* berfungsi untuk mengecek apakah *p* bilangan prima atau komposit.
7. *Button* berfungsi untuk mengosong *textboxt.*
8. *Textboxt* berfungsi untuk menampilkam tulisan 'Bilangan Primana' atau 'Bilangan Komposit'.
9. *Label* digunakan untuk judul 'Bilangan Prima (q)'.
10. *Textboxt* berfungsi untuk menginputkan nilai *q*.
11. *Button* berfungsi untuk mengecek apakah *q* bilangan prima atau komposit
12. *Button* berfungsi untuk mengosong *textboxt.*
13. *Textboxt* berfungsi untuk menampilkam tulisan 'Bilangan Primana' atau 'Bilangan Komposit.
14. *Label* digunakan untuk judul 'Nilai N'.
15. *Textboxt* berfungsi untuk menampilkan nilai N.
16. *Button* berfungsi untuk menghitung nilai N.
17. *Label* digunakan untuk judul 'Input nilai e'.
18. *Textboxt* berfungsi untuk menampilkan nilai e.
19. *Button* digunakan untuk menampilkan nilai e dengan syarat gcd=1.
20. *Label* digunakan untuk informasi nilai e.
21. *Label* berfungsi untuk judul 'Nilai d'.
22. *Textboxt* digunakan untuk menampilkan nilai d.
23. *Button* digunakan untuk menghitung nilai d.
24. *Label* berfungsi untuk judul 'Nilai r1'.
25. *Textboxt* digunakan untuk menampilkan nilai r1.
26. *Button* digunakan untuk menghitung nilai r1.
27. *Label* berfungsi untuk judul 'Nilai r2'.
28. *Textboxt* digunakan untuk menampilkan nilai r2.
29. *Button* digunakan untuk menghitung nilai r2.
30. *GroupBox* untuk menggabungkan beberapa komponen kunci.
31. *Label* berfungsi untuk judul 'Public Key'.
32. *Textboxt* digunakan untuk menampilkan public key.
33. *Label* berfungsi untuk judul 'Private Key'.
34. *Textboxt* digunakan untuk menampilkan private key.
35. *Button* digunakan untuk meyimpan proses pembangkit kunci.

36. *Button* digunakan untuk menghapus isi dari *form* pada *groupbox* kunci.
37. *Button* digunakan untuk melanjutkan ke proses selanjutnya yaitu proses enkripsi.

c. *Interface* Halaman Enkripsi

Proses penyandian pesan dimulai pada halaman ini, yaitu halaman enkripsi. Proses enkripsi dimulai dengan enkripsi *plaintext* kemudian enkripsi kunci simetris. *Interface* halaman enkripsi dapat dilihat pada gambar digambarkan pada gambar 3.19.

Gambar 3.19 Rancangan *Interface* Halaman Enkripsi

Keterangan gambar:

1. *TabControl*berfungsi untuk memilih proses yang diinginkan, tab ini terdiri dari 4 proses yang dipilih yaitu ‘Pembangkit Kunci’, ‘Enkripsi’, ‘Dekripsi’, dan ‘Penggunaan Aplikasi’.
2. *GroupBox* untuk menggabungkan beberapa komponen proses enkripsi *plaintext*.

3. *Label* digunakan untuk judul ‘Plaintext’.
4. *Rich Text Box* digunakan untuk menginput *plaintext* atau membaca isi *file* yang akan dienkripsi.
5. *Label* digunakan untuk judul ‘Kunci Simetris’.
6. *Textboxt* berfungsi untuk menginput kunci simetris.
7. *Label* digunakan untuk judul ‘Ciphertext’.
8. *Rich Text Box* digunakan untuk menampung *ciphertext* hasil enkripsi algoritma *blowfish.*
9. *Button* digunakan untuk mencari *file* yang akan digunakan.
10. *Button* digunakan untuk melakukan proses enkripsi *plaintext*.
11. *GroupBox* untuk menggabungkan beberapa komponen proses enkripsi kunci simetris.
12. *Label* digunakan untuk judul ‘Kunci Simetris’.
13. *Textboxt* berfungsi untuk menampilkan kunci simetris.
14. *Label* digunakan untuk judul ‘Private Key’.
15. *Textboxt* berfungsi untuk menginputkan *private key*.
16. *Label* digunakan untuk judul ‘Cipherkey’.
17. *Textboxt* berfungsi untuk menampilkan *cipherkey*.
18. *Button* digunakan untuk menampilkan kunci simetris.
19. *Button* digunakan untuk melakukan proses enkripsi kunci simetris.
20. *GroupBox* untuk menggabungkan beberapa komponen *file* info.
21. *Label* digunakan untuk judul ‘File Loc’.
22. *Textboxt* berfungsi untuk menampilkan lokasi *file* yang akan dienkripsi.
23. *Label* digunakan untuk judul ‘Panjang Karakter’.
24. *Textboxt* berfungsi untuk menampilkan panjang karakteryang akan dienkripsi.
25. *Label* digunakan untuk judul ‘Running Time Teks’.
26. *Textboxt* berfungsi untuk menampilkan *running time file* yang akan dienkripsi.
27. *Label* digunakan untuk judul ‘Running Time Kunci’.
28. *Textboxt* berfungsi untuk menampilkan *running time* kunciyang akan dienkripsi.
29. *Button* digunakan untuk mengosongkan kembali halaman enkripsi.
30. *Button* digunakan untuk menyimpan hasil enkripsi.

d. *Interface* Halaman Dekripsi

Halaman dekripsi adalah halaman yang digunakan untuk mengembalikan penyandian pesan dengan proses dekripsi pesan sehingga pesan dapat dimengerti oleh penerima pesan. Gambar 3.20 merupakan rancangan *interface* halaman dekripsi.

Gambar 3.20 Rancangan *Interface* Halaman Dekripsi

Keterangan gambar:

1. *TabControl*berfungsi untuk memilih proses yang diinginkan, tab ini terdiri dari 4 proses yang dipilih yaitu ‘Pembangkit Kunci’, ‘Enkripsi’, ‘Dekripsi’, dan ‘Penggunaan Aplikasi’.
2. *GroupBox* untuk menggabungkan beberapa komponen proses dekripsi kunci simetris.
3. *Label* digunakan untuk judul ‘Cipherkey’.
4. *Textboxt* berfungsi untuk menampilkan *cipherkey*.
5. *Label* digunakan untuk judul ‘Private Key’.
6. *Textboxt* berfungsi untuk menginputkan *private key*.

7. *Label* digunakan untuk judul ‘Kunci Simetris’.
8. *Textboxt* berfungsi untuk menginputkan *private key*.
9. *Button* digunakan untuk mencari *file* yang akan digunakan.
10. *Button* digunakan untuk melakukan proses dekripsi kunci simetris.
11. *GroupBox* untuk menggabungkan beberapa komponen *file* info.
12. *Label* digunakan untuk judul ‘File Loc’.
13. *Textboxt* berfungsi untuk menampilkan lokasi *file* yang akan didekripsi.
14. *Label* digunakan untuk judul ‘Panjang Karakter’.
15. *Textboxt* berfungsi untuk menampilkan panjang karakteryang akan didekripsi.
16. *Label* digunakan untuk judul ‘Running Time Teks’.
17. *Textboxt* berfungsi untuk menampilkan *running time file* yang akan didekripsi.
18. *Label* digunakan untuk judul ‘Running Time Kunci’.
19. *Textboxt* berfungsi untuk menampilkan *running time* kunciyang akan didekripsi.
20. *GroupBox* untuk menggabungkan beberapa komponen proses dekripsi *plaintext*.
21. *Label* digunakan untuk judul ‘Ciphertext’.
22. *Rich Text Box* digunakan untuk menampilkan *ciphertext* yang akan didekripsi.
23. *Label* digunakan untuk judul ‘Kunci Simetris’.
24. *Textboxt* berfungsi untuk menginput kunci simetris.
25. *Label* digunakan untuk judul ‘Plaintext’.
26. *Rich Text Box* digunakan untuk menampung *plaintext* hasil dekripsi.
27. *Button* digunakan untuk menampilkan kunci simetris.
28. *Button* digunakan untuk melakukan proses dekripsi *ciphertext*.
29. *Button* digunakan untuk mengosongkan kembali halaman enkripsi.
30. *Button* digunakan untuk menyimpan hasil enkripsi.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1.Implementasi Sistem

Dalam penelitian ini, sistem dibangun dengan menggunakan bahasa pemrograman C# dan software yang digunakan adalah SharpDevelop versi 5.1. Terdapat empat bagian halaman yang terdapat dalam sistem yang dibangun, yaitu halaman utama untuk menampilkan judul dan identitas singkat pembuat sistem kepada *user*, halaman pembangkit kunci sebagai halaman yang berfungsi untuk proses pembangkitan kunci algoritma, halaman enkripsi sebagai halaman yang berfungsi untuk melakukan proses enkripsi pesan, dan halaman dekripsi sebagai halaman yang berfungsi untuk melakukan proses dekripsi pesan. Sistem pengamanan teks dengan *hybrid cryptosystem* algoritma *Multi-Power RSA* dan algoritma *Blowfish* ini diberi nama Blowfish_multipowerRSA.

4.1.1. Halaman Utama

Halaman utama merupakan halaman yang muncul pertama kali ketika *user* menjalankan sistem. Halaman utama dapat dilihat pada gambar 4.1.

Gambar 4.1 Halaman Utama

Pada gambar 4.1 terlihat halaman utama yang memiliki judul penelitian, logo Universitas Sumatera Utara, identitas pembuat sistem, dan sebuah *button* masuk yang diberikan aksi untuk masuk ke halaman selanjutnya.

4.1.2. Halaman Pembangkit Kunci

Halaman pembangkit kunci berfungsi untuk membangkitkan public key dan private key yang nantinya akan digunakan untuk proses enkripsi dan dekripsi. Pada pembangkitan kunci user diminta untuk memasukkan 2 bilangan prima. Halaman pembangkit kunci dapat dilihat pada gambar 4.2.

Gambar 4.2 Halaman Pembangkit Kunci

4.1.3. Halaman Enkripsi

Halaman enkripsi digunakan *user* untuk mengenkripsi pesan atau mengubah pesan menjadi kode-kode yang tidak dapat terbaca. Pesan yang ingin dienkripsi dapat diambil dari *file* yang telah tersimpan sebelumnya yang berekstensi .txt, .doc ataupun .docx. Selain itu, u*ser* juga dapat langsung menginputkan atau mengetik teks yang akan dienkripsi secara manual pada *text box plaintext*. Pesan yang dikirim akan dienkripsi dengan kunci simetris dank unci simetris akan dienkripsi dengan *public key*. Pada *file* info nantinya akan menunujukan lokasi *file*, panjang karakter, lama waktu enkripsi pesan, dan juga lama waktu enkripsi kunci simetris. Halaman Enkripsi dapat dilihat pada gambar 4.3.

Gambar 4.3 Halaman Enkripsi

4.1.4. Halaman Dekripsi

Halaman dekripsi digunakan *user* untuk mengembalikan *ciphertext* menjadi *plaintext* atau mengubah pesan yang telah dienkripsi kembali ke pesan awal. Pada halaman ini, ciphertext dan cipherkey akan didekripsi sehingga pesan yang dikirim dapat dibaca atau kembali seperti semula. Pada *file* info nantinya akan menunujukan lokasi *file*, panjang karakter, lama waktu enkripsi pesan, dan juga lama waktu enkripsi kunci simetris. Halaman dekripsi dapat dilihat pada gambar 4.4.

Gambar 4.4 Halaman Dekripsi

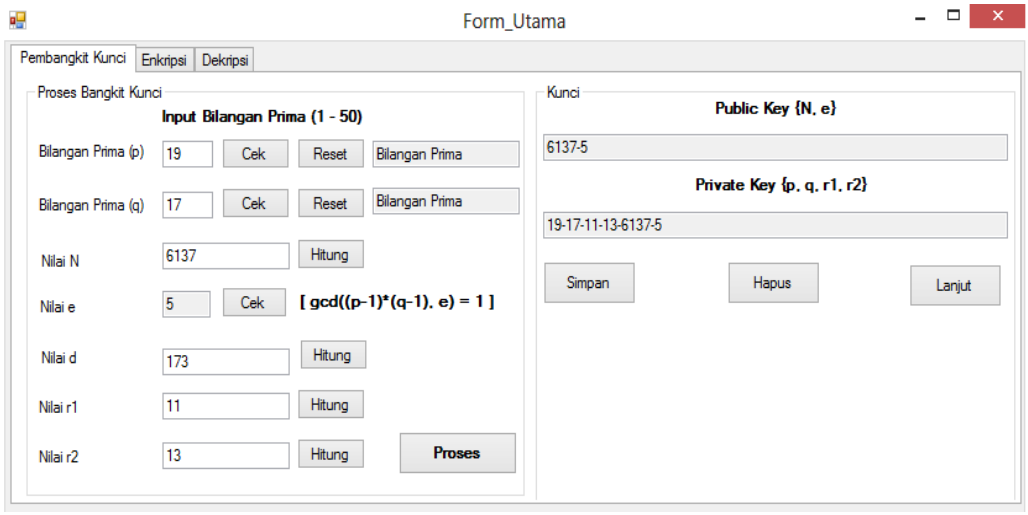
4.2.Pengujian Sistem

Pengujian dilakukan terhadap algoritma *Multi-Power RSA* dan algoritma *Blowfish* dalam *hybrid cryptosystem* untuk membuktikan bahwa sistem dapat berjalan dengan baik dalam melakukan pembangkitan kunci, enkripsi dan dekripsi. Pada pengujian ini parameter yang digunakan adalah *running time* teks dan *running time* kunci. Adapun kriteria pengujian sistem sebagai berikut :

1. *Plaintext* berupa *file* berekstensi .txt, .doc atau .docx dengan jumlah panjang 6 karakter. *Plaintext* juga dapat berupa kata atau kalimat yang dapat diinput manual oleh *user*.
2. *Ciphertext* yang disimpan dalam bentuk *file* yang berekstensi .mahadi dan *plaintext* yang disimpan dalam bentuk *file* berekstensi .doc.
3. *Public key* dan *private key* disimpan dalam bentuk file yang berekstensi .kunci.
4. Kunci simetris harus diinputkan secara manual oleh user dengan syarat, kunci simetris tersebut harus diinputkan dengan huruf ataupun angka.
5. Penghitungan *running time* teks dan *running time* kunci dilakukan secara manual dengan menggunakan bahasa pemrograman C#.
6. Sistem ini diuji dengan *Personal Computer* dengan spesifikasi *processor* intel (R) Core™ i5 CPU M430 @ 2.27 GHz, Memory 2 GB RAM.

4.2.1. Pengujian Pembangkit Kunci

Pada proses pembangkitan kunci, *user* haruslah memasukkan bilangan prima. Bilangan prima yang dimasukkan oleh *user* bisa dicek kebenarannya dengan cara menekan *button* cek. Selanjutnya, *user* harus menekan *button* hitung dan juga proses agar mendapatkan nilai dari *public key* dan *private key*. Maka, hasil proses dari pembangkitan kunci dapat dilihat pada gambar 4.5



Gambar 4.5 Pengujian Pembangkit Kunci

Adapun langkah-langkah yang dilakukan untuk proses pembangkitan kunci dengan algoritma *Multi-Power RSA* adalah, sebagai berikut:

a) Memasukkan bilangan pada p dan q, yang merupakan bilangan prima dengan batasan bilangan 1 – 50, bilangan prima yang digunakan $p = 19$ dan $q = 17$.

b) Bilangan yang diinput bisa di cek kebenarannya dengan cara menekan *button* cek, apabila bilangan tersebut prima maka perhitungannya dapa dilanjutkan, akan tetapi apabila bilangan yang dimasukkan merupakan bilangan komposit maka perhitungan tidak dapat dilanjutkan.

c) Selanjutnya, menghitung nilai N dengan rumus: $N = p^{b-1}.q$, $b = 3$.

$N = p^{b-1}.q$

$N = 19^2.17$

$N = 361.17 = 6137$

d) Mengecek nilai e dan mencari nilai e^{-1} dengan memenuhi syarat [gcd(p-1).(q-1).e) = 1]

Pilih nilai e $\{2 < e < (p-1).(q-1)\}$

$\{2 < e < (18).(16)\}$

$\{2 < e < 288\}$; nilai e yang dipilih adalah 5.

Mencari nilai e^{-1}, dengan rumus: $e^{-1} = e^{-1}.e \bmod (p-1).(q-1)$. Maka akan diperoleh nilai yang ditunjukkan pada tabel 4.1.

Tabel 4.1 Nilai e^{-1} Pada Pembangkit Kunci

e^{-1}	$e^{-1}.e$	$e^{-1}.e$ mod $(p\text{-}1).(q\text{-}1)$
1	5	5
2	10	10
…	…	
173	865	1

Maka nilai e^{-1} dari pembangkit kuncinya adalah 173, karena memenuhi syarat [gcd$(p-1).(q-1).e$) =1]

e) Selanjutnya, menghitung nilai d yang akan digunakan untuk mencari nilai r_1 dan r_2, dengan rumus: $d = e^{-1} \bmod (p-1).(q-1)$

$d = e^{-1} \bmod (p-1).(q-1)$

$d = 173 \bmod (18).(16)$

$d = 173 \bmod 288$

$d = 173$

f) Menghitung nilai r_1 dan r_2, sebagai berikut:

$r_1 = d \bmod (p-1)$ $\quad$ $r_2 = d \bmod (q-1)$

$r_1 = 173 \bmod (18)$ $\quad$ $r_2 = 173 \bmod (16)$

$r_1 = 11$ $\quad$ $r_2 = 13$

g) Maka, didapatlah *public key* {N, e}, yaitu {6137, 5} dan *private key* {p, q, r_1, r_2}, yaitu {19, 17, 11, 13} dan dapat disimpan.

4.2.2. Pengujian Enkripsi

Pada proses enkripsi, pesan yang akan dienkripsi dapat diambil dari *file* yang telah tersimpan sebelumnya yang berekstensi .doc. Selain itu, u*ser* juga dapat langsung menginputkan atau mengetik teks yang akan dienkripsi secara manual pada *textbox plaintext*. Ketika *user* menginputkan *plaintext* berupa *file*, maka isi dari *file* tersebut dapat ditampilkan pada *textbox* yang telah disediakan dengan menekan *button* cari. Kemudian, *user* harus melakukan *input* kunci simetris secara manual. Proses enkripsi *plaintext* akan dimulai ketika *user* menekan *button* enkripsi, lalu akan muncul *ciphertext* atau pesan yang sudah dienkripsi pada *textbox* yang telah disediakan dan juga akan muncul kunci simetris pada enkripsi kunci simetris serta akan muncul *file location*, panjang karakter dan *running time* teks pada *textbox* disebelah kanan bawah.

Setelah pesan sudah terenkripsi, user harus menyimpan *ciphertext* tersebut dengan cara menekan *button* simpan. Selanjutnya, *user* harus menginput *fileprivate key* berektensi .*kunci* dengan cara menekan *button* cari kunci. Proses enkripsi kunci simetris akan dimulai ketika *user* menekan *button* enkripsi, lalu akan muncul *cipherkey* atau kunci simetris yang sudah dienkripsi pada *textbox* yang telah disediakan dan juga akan muncul informasi *running time* kunci pada *textbox* disebelah kanan bawah. Setelah kunci simetris terenkripsi, *user* harus menyimpan *cipherkey* tersebut dengan cara menekan *button* simpan. Hasil dari pengujian proses enkripsi ditunjukkan pada gambar 4.6.

Gambar 4.6 Pengujian Enkripsi

4.2.2.1. Pengujian Enkripsi Algoritma Multi-Power RSA

Enkripsi Algoritma *Multi-Power RSA* dilakukan dengan mengubah masing-masing karakter dari kunci simetris menjadi kode ASCII. Kunci simetris yang akan dienkripsi dalam sistem ini adalah “Ilmukomputer” dan kunci simetris tersebut akan dienkripsi menggunakan *public key* yang sebelumnya telah dibangkitkan. Kemudian lakukan perhitungan untuk mendapatkan *cipherkey* dari masing-masing karakter. Karakter “-“ akan menjadi pemisah antara *cipherkey* satu dengan yang lainnya.

Contoh:

Enkripsilah karakter "I" yang memiliki kode ASCII 73 dengan menggunakan algoritma *Multi-Power RSA*!

1) *Public key* yaitu N dan *e* sudah dibangkitkan terlebih dahulu, yaitu N = 6137 dan *e* = 5.
2) Selanjutnya lakukan perhitungan dengan menggunakan rumus $C = m^e \bmod N$.

   C = me mod N

   $C = 73^5 \bmod 6137$

   C = 5267

   Jadi, *cipherkey* karakter "I" yang mempunyai kode ASCII 73 adalah 5267.
3) Ulangi langkah 1 dan 2 untuk mengenkripsi karakter berikutnya. Sehingga didapatkan hasil *cipherkey* kunci simetris secara lengkap, dapat dilihat pada tabel 4.2.

Tabel 4.2 Hasil *Cipherkey* Kunci Simetris "Ilmukomputer"

Karakter	Kode ASCII	$C = m^e \bmod N$
I	73	5267
l	108	1724
m	109	1150
u	117	4898
k	107	5233
o	111	5856
m	109	1150
p	112	2590
u	117	4898
t	116	488
e	101	3178
r	114	1805

4.2.3. Pengujian Dekripsi

Pada proses dekripsi, langkah pertama yang harus dilakukan adalah *user* harus memasukkan *file cipherkey* yang berekstensi .cipherkey dan juga menginput *file private key* yang berekstensi .kunci dari penyimpanan ke dalam sistem. Proses dekripsi *cipherkey* akan terjadi ketika *user* menekan *button* dekripsi, lalu akan muncul

kunci simetris dan juga *running time* kunci pada *textbox* disebelah kanan atas. Kemudian, *user* harus menginput *file ciphertext* yang berekstensi .mahadi dengan cara menekan *button* cari, maka akan muncul pesan yang sudah terenkripsi pada *textbox* yang telah disediakan dan juga akan muncul *file location* juga panjang karakter pada *texboxt* disebelah kanan atas. Proses dekripsi *ciphertext* akan dimulai ketika *user* menekan *button* dekripsi, lalu akan muncul *plaintext* dan juga *running time* teks. Hasil pengujian dekripsi ditunjukkan pada gambar 4.7.

Gambar 4.7 Pengujian Dekripsi

4.2.3.1. Pengujian Dekripsi Algoritma Multi-Power RSA

Pada proses dekripsi pada algoritma *Multi-Power RSA* digunakan *private key* {p, q, r_1, r_2} yang telah dibangkitkan sebelumnya. Dalam proses dekripsi nantinya akan menggunakan algoritma *Hensel Lifting* dalam mendekripsi *cipherkey* untuk mempercepat dan mempermudah proses dekripsi.

Contoh:

Dekripsilah *cipherkey* 5267, 1724, 1150, 4898, 5233, 5856, 1150, 2590, 4898, 488, 3178, 1805 dengan menggunakan algoritma *Multi-Power RSA*!

- Dekripsi *cipherkey* "5267":
  1) p^2 = p.p
     = 19.19
     = 361

2) $C_1 = C \bmod p^2$

$= 5267 \bmod 361$

$= 213$

3) $C_2 = C \bmod q$

$= 5267 \bmod 17$

$= 14$

4) HENSEL LIFTING

a. $M_1 = C_1^{r1-1} \bmod p$

$= 213^{11-1} \bmod 19$

$= 1296$

b. $K_0 = M_1.C_1 \bmod p$

$= 1296.213 \bmod 19$

$= 276048 \bmod 19$

$= 16$

c. $A = -K_0^{e} \bmod p^2$

$= -(16)^7 \bmod 361$

$= 173$

d. $A = A + C \bmod p^2$

$= 173 + 5267 \bmod 361$

$= 5440 \bmod 361$

$= 25$

e. $M_1 = M_1.A \bmod p^2$

$= 1296.25 \bmod 361$

$= 32400 \bmod 361$

$= 271$

f. $M_1 = M_1.(e^{-1} \bmod p) \bmod p^2$

$= 271.(e^{-1} \bmod 19) \bmod 361$

$= 271.4 \bmod 361$

$= 1$

g. $M_1 = M_1 + K_0 \bmod p^2$

$= 1 + 16 \bmod 361$

$= 17$

5) M_2 $= C_2^{r2} \bmod q$

$= 14^{13} \bmod 17$

$= 90$

6) V $= M_2 - M_1 \bmod q$

$= 90 - 17 \bmod 17$

$= 5$

7) V $= V.((p^2)^{-1} \bmod q) \bmod q)$

$= 5.((361^{-1} \bmod 17) \bmod 17)$

$= 5.13 \bmod 17$

$= 14$

8) M $= V.p^2 \bmod N$

$= 14.361 \bmod N$

$= 5054 \bmod 6137$

$= 5054$

9) M $= M + M_1 \bmod N$

$= 5054 + 17 \bmod 6137$

$= 5071$ (karakter I berhasil di dekripsi)

4.2.4. Waktu Proses (Real Running Time)

Pada penelitian ini salah satu tujuannya adalah menghitung ataupun menganalisa waktu proses enkripsi dan dekripsi dari masing-masing algoritma dengan percobaan panjang *plaintext* maupun kunci simetris yang berbeda-beda. Satuan waktu yang digunakan untuk menghitung kecepatan waktu proses enkripsi dan dekripsi adalah *millisecond* dan jenis *file* yang digunakan berekstensi .doc, .docx, .txt.

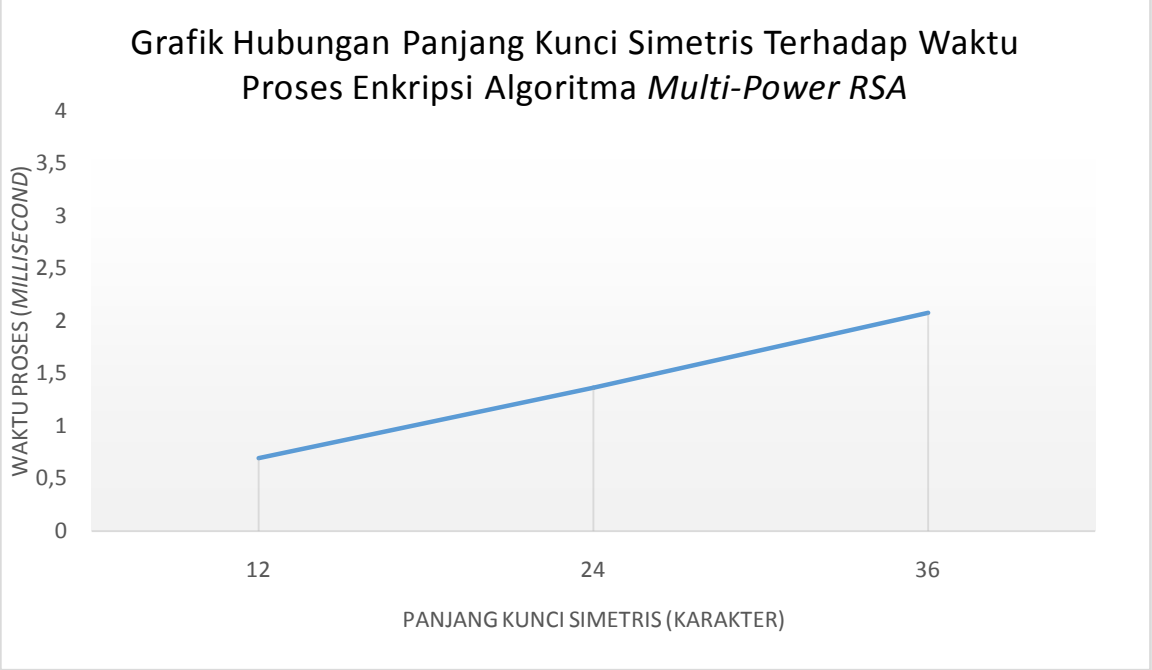
4.2.4.1. Waktu Proses Enkripsi Algoritma Multi-Power RSA

Pada enkripsi algoritma *Multi-Power RSA*, waktu proses dihitung berdasarkan panjang kunci simetrisnya, yaitu: 12 karakter, 24 karakter, dan 36 karakter. Hasil *running time* proses enkripsi algoritma *Multi-Power RSA* ditunjukkan pada tabel 4.3.

Tabel 4.3 Percobaan Enkripsi Algoritma *Multi-Power RSA*

Panjang Kunci Simetris	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
12	0.7776	0.6222	0.7223	0.6308	0.7287	0.69632
24	1.5529	1.5738	1.2436	1.2214	1.2228	1.3629
36	2.447	2.1331	1.831	2.1354	1.8374	2.07678

Tabel 4.3 menunjukkan waktu proses yang dihasilkan berdasarkan panjang kunci simetris 12 karakter, karakter 24, dan karakter 36 dilakukan sebanyak 5 kali percobaan dari masing-masing panjang kunci simetris. Selanjutnya, didapatkan nilai rata-rata seperti yang diilustrasikan pada grafik yang ditunjukan pada gambar 4.8.



Gambar 4.8 Grafik Hubungan Panjang Kunci Simetris terhadap Waktu Proses Enkripsi Algoritma *Multi-Power RSA*

4.2.4.2. Waktu Proses Enkripsi Algoritma Blowfish

Pada proses enkripsi algoritma *Blowfish*, waktu proses dihitung berdasarkan panjang *plaintext* yang akan dienkripsi dan panjang kunci simetrisnya. Pada percobaan yang dilakukan, ada 3 panjang *plaintext* yang digunakan, yaitu : 10, 100, dan 100 karakter. Sedangkan, panjang kunci simetris yang digunakan, yaitu : 12, 24, dan 36 karakter.

1. Enkripsi Blowfish dengan panjang plaintext 10, 100, dan 1000 karakter dengan panjang kunci simetris 12 karakter yangditunjukkan pada tabel 4.4.

Tabel 4.4 Percobaan Enkripsi *Blowfish* dengan Kunci Simetris 12 Karakter

Panjang *Plaintext*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
10	0.4583	0.4773	0.4311	0.4655	0.4556	0.45756
100	0.576	0.6105	0.6005	0.6018	0.5946	0.59668
1000	2.1521	2.1304	2.1852	2.1426	2.144	2.15086

2. Enkripsi Blowfish dengan panjang plaintext 10, 100, dan 1000 karakter dengan panjang kunci simetris 24 karakter yangditunjukkan pada tabel 4.5.

Tabel 4.5 Percobaan Enkripsi *Blowfish* dengan Kunci Simetris 24 Karakter

Panjang *Plaintext*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
10	0.466	0.4782	0.4877	0.432	0.4619	0.4619
100	0.5792	0.6127	0.6028	0.6227	0.5969	0.60286
1000	2.201	2.2703	2.1417	2.1616	2.1517	2.18526

3. Enkripsi Blowfish dengan panjang plaintext 10, 100, dan 1000 karakter dengan panjang kunci simetris 36 karakter yang ditunjukkan pada tabel 4.6.

Tabel 4.6 Percobaan Enkripsi *Blowfish* dengan Kunci Simetris 36 Karakter

Panjang *Plaintext*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
10	0.4673	0.4927	0.4746	0.475	0.4741	0.47674
100	0.5801	0.5991	0.5896	0.6331	0.6023	0.60084
1000	2.1489	2.1313	2.1499	2.1335	2.1349	2.1397

Hasil waktu proses yang diperoleh berdasarkan percobaan jumlah panjang *plaintext* 10 karakter, 100 karakter dan 1000 karakter dengan panjang kunci simetris 12 karakter, 24 karakter, 36 karakter ditunjukkan pada sebuah grafik pada gambar 4.9.

Gambar 4.9 Grafik Hubungan Panjang *Plaintext* terhadap Waktu Proses Enkripsi Algoritma *Blowfish*

4.2.4.3. Waktu Proses Dekripsi Algortima Multi-Power RSA

Pada dekripsi algoritma *Multi-Power RSA*, waktu proses dihitung berdasarkan *cipherkey* yang telah didapatkan dari hasil enkripsi kunci simetris sebelumnya dan ditunjukkan pada tabel 4.7.

Tabel 4.7 Percobaan Dekripsi Algoritma *Multi-Power RSA*

Panjang *Cipherkey*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
47	0.2432	0.2649	0.2504	0.2658	0.2499	0.25484
94	0.3872	0.3908	0.3962	0.3971	0.3931	0.39288
141	0.5326	0.5253	0.5163	0.5285	0.5176	0.52406

Hasil waktu proses dekripsi *ciphertext* dengan algoritma *Multi-Power RSA* diilustrasikan dalam grafik yang ditunjukkan pada gambar 4.10.

Gambar 4.10 Grafik Hubungan Panjang *Cipherkeyt* dengan Waktu Proses Dekripsi Algoritma *Multi-Power RSA*

4.2.4.4. Waktu Proses Dekripsi Algoritma Blowfish

Pada proses dekripsi algoritma *Blowfish*, waktu proses dihitung berdasarkan panjang *ciphertext* yang akan dienkripsi dan panjang *cipherkey*. Pada percobaan yang dilakukan, ada 3 panjang *ciphertext* yang digunakan, yaitu: 65, 433, dan 4033 karakter. Sedangkan, panjang *cipherkey* yang digunakan, yaitu: 47, 94, dan 141 karakter.

1. Dekripsi *Blowfish* dengan panjang *ciphertext* 65, 433, dan 4033 karakter dengan panjang *cipherkey* 47 karakter yang ditunjukkan pada tabel 4.8.

   Tabel 4.8 Percobaan Enkripsi *Blowfish* dengan *Cipherkey* 47 Karakter

Panjang *Ciphertext*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
65	0.4288	0.4374	0.4356	0.423	0.4252	0.43
433	0.4977	0.4995	0.4981	0.4741	0.5067	0.49522
4033	1.076	1.0665	1.0756	1.1009	1.1136	1.08652

2. Dekripsi *Blowfish* dengan panjang *ciphertext* 65, 433, dan 4033 karakter dengan panjang *cipherkey* 94 karakter yang ditunjukkan pada tabel 4.9.

   Tabel 4.9 Percobaan Enkripsi *Blowfish* dengan *Cipherkey* 94 Karakter

Panjang *Ciphertext*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
65	0.4625	0.4755	0.4433	0.4451	0.4193	0.44914
433	0.5244	0.5113	0.5013	0.5108	0.5144	0.51244
4033	1.0806	1.0688	1.0701	1.1245	1.0656	1.08192

3. Dekripsi *Blowfish* dengan panjang *ciphertext* 65, 433, dan 4033 karakter dengan panjang *cipherkey* 141 karakter yang ditunjukkan pada tabel 4.10.

Tabel 4.10 Percobaan Enkripsi *Blowfish* dengan *Cipherkey* 141 Karakter

Panjang *Ciphertext*	Waktu Proses (*milisecond*)					
	Percobaan 1	**Percobaan 2**	**Percobaan 3**	**Percobaan 4**	**Percobaan 5**	**Rata-rata**
65	0.4374	0.4832	0.4438	0.4411	0.4424	0.44958
433	0.5443	0.5778	0.5697	0.5833	0.5679	0.5686
4033	1.4474	1.6055	1.4528	1.4556	1.5004	1.49234

Hasil waktu proses yang diperoleh berdasarkan percobaan jumlah panjang *ciphertext* 10 karakter, 100 karakter dan 1000 karakter dengan panjang kunci simetris 47 karakter, 94 karakter, 141 karakter ditunjukkan pada sebuah grafik pada gambar 4.11.

Gambar 4.11 Grafik Hubungan Panjang *Ciphertext* dengan Waktu Proses Dekripsi Algoritma *Blowfish*

BAB 5

KESIMPULAN DAN SARAN

5.1.Kesimpulan

Berdasarkan analisis, perancangan dan hasil Pengamanan Teks dengan *Hybrid Cryptosystem* Algoritma *Multi-Power RSA*dan Algoritma *Blowfish*, maka dapat disimpulkan bahwa :

1. Pengamanan teks dengan metode *hybrid cryptosystem* dengan algoritma *Multi-Power RSA*dan algoritma *Blowfish*untuk merahasiakan teksberjalan dengan baik. Teksatau pesan berhasil dienkripsi, namun pada proses dekripsi kunci simetris ada sedikit perubahan karakter (huruf r) pada hasil dekripsinya yang mengakibatkan proses dekripsi *ciphertext*, hasil dekrispi tidak sama dengan teks semula.
2. Berdasarkan grafik panjang kuncisimetris dengan waktu proses algoritma *Multi-Power RSA* didapatkan hasil panjang kunciberbanding lurus dengan waktu membentuk grafik yang linear. Artinya, semakin panjang karakter panjang kuncimaka waktu yang dibutuhkan untuk melakukan enkripsi juga semakin lama.
3. Waktu proses yang dibutuhkan untuk proses dekripsi *cipherkey* lebih cepat dibandingkan dengan waktu proses yang dibutuhkan pada saat enkripsi *cipherkey* dengan menggunakan algoritma yang sama yaitu algoritma *Multi-Power RSA*.
4. Berdasarkan percobaan yang dilakukan pada algoritma *Blowfish*, waktu proses yang dihasilkan dekripsi *plaintext* lebih cepat dibandingkan hasil enkripsi *plaintext*.

5.2.Saran

Berikut ini beberapa saran yang dapat dipertimbangkan untuk memperbaiki penelitian ini dan untuk penelitian kedepannya, yaitu :

1. Pada penelitian ini terdapat 1 karakter (huruf r) yang tidak berhasil di dekripsi sebagaimana mestinya, untuk itu diharapkan pada penelitian berikutnya semua karakter dapat terdekripsi dengan benar.
2. Penelitian yang dibangun berbasis *desktop*, diharapkan pada penelitian kedepannya dapat membangun aplikasi pada perangkat lainnya, seperti Android, Windows Phone, IOS, maupun yang berbasis *mobile*.
3. Dalam peneltian ini *file* yang digunakan hanya 3 jenis *file*, yaitu .txt, .doc, .docx sebagai inputan, diharapkan kedepannya dapat menambah jenis *file* sebagai inputannya.