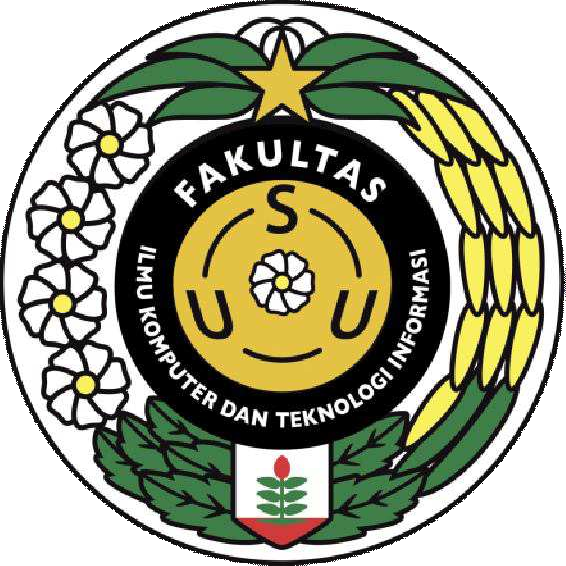
PERANCANGAN APLIKASI PENGAMANAN DATA FILE MENGGUNAKAN KOMBINASI METODE AFFINE CIPHER DAN RSA

SKRIPSI

FITRA MAULIDA

101402101



PROGRAM STUDI S-1 TEKNOLOGI INFORMASI FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

MEDAN

2015

PERANCANGAN APLIKASI PENGAMANAN DATA FILE MENGGUNAKAN KOMBINASI METODE AFFINE CIPHER DAN RSA

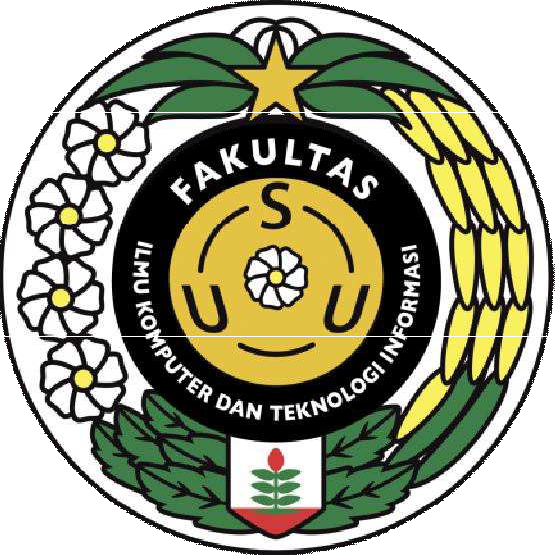
SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah

Sarjana Teknologi Informasi

FITRA MAULIDA

101402101



PROGRAM STUDI S-1 TEKNOLOGI INFORMASI FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

MEDAN

2015

**PERSETUJUAN**

Judul : PERANCANGAN APLIKASI PENGAMANAN DATA FILE MENGGUNAKAN KOMBINASI METODE AFFINE CIPHER DAN RSA

Kategori : SKRIPSI

Nama : FITRA MAULIDA Nomor Induk Mahasiswa : 101402101

Program Studi : S1 TEKNOLOGI INFORMASI Departemen : TEKNOLOGI INFORMASI

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Komisi Pembimbing :

Pembimbing 2 Pembimbing 1

Dedy Arisandi, ST.M.Kom M. Andri Budiman, ST., M.Comp.Sc., M.E.M. NIP. 19790831 200912 1 002 NIP. 19751008 200801 1 011

Diketahui/disetujui oleh

Program Studi S1 Teknologi Informasi

Ketua,

Muhammad Anggia Muchtar, ST., MM.IT NIP. 198001102008011010

**PERNYATAAN**

PERANCANGAN APLIKASI PENGAMANAN DATA FILE MENGGUNAKAN KOMBINASI METODE AFFINE CIPHER DAN RSA

SKRIPSI

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

Medan,

Fitra Maulida

101402101

**UCAPAN TERIMA KASIH**

Puji dan syukur penulis panjatkan kepada Allah SWT Yang Maha Pengasih dan Maha Penyayang, dengan limpahan rahmat dan karunia-Nya penulisan tugas akhir ini berhasil diselesaikan dalam waktu yang telah ditetapkan.

Terima kasih kepada Bapak Muhammad Anggia Muchtar, ST., MM.IT selaku ketua Program Studi Teknologi Informasi Universitas Sumatera Utara.

Terima kasih kepada Bapak M. Andri Budiman ST., M.Comp.Sc., M.E.M. dan Bapak Dedy Arisandi, ST., M.Kom selaku pembimbing atas segala ilmu, motivasi, nasehat, dan bantuan yang telah diberikan sehingga penulis dapat menyelesaikan penelitian tugas akhir hingga penyelesaian penulisan skripsi ini. Untuk seluruh staf pengajar Program Studi Teknologi Informasi, terutama Bapak Dani Gunawan, ST., M.T dan Ibu Dr. Erna Budhiarti Nababan, M.IT selaku penguji.

Ucapan terima kasih yang tiada tara untuk kedua orang tua penulis. Untuk Mamak dan Ayah yang telah menjadi orang tua terhebat sejagad raya, yang selalu memberikan motivasi, nasehat, cinta, perhatian, dan kasih sayang serta doa yang tentu takkan bisa penulis balas.

Untuk ke empat kakak dan juga adik penulis, terima kasih atas segala perhatian, kasih sayang, dan motivasi serta doanya. Terima kasih banyak telah menjadi motivator yang luar biasa sehingga penulis dapat menyelesaikan penelitian ini.

Terima kasih untuk Rabiatul Adawiyah D dan Andreas TSM yang telah banyak membantu dan mengajarkan penulis, Serta terima kasih kepada sahabat - sahabat Nency, Ilda, Epin dan Fhilien yang telah memberikan motivasi, dukungan, bantuan, serta perhatian sehingga skripsi ini bisa selesai tepat waktu.

Dan terima kasih untuk teman-teman seperjuangan Teknologi Informasi USU Angkatan 2010 serta abang dan kakak senior untuk bantuan dan kebersamaan selama perkuliahan di Teknologi Informasi USU.

Semoga segala kebaikan dan bantuannya dibalas oleh Allah SWT dan semoga tugas akhir ini dapat bermanfaat bagi semua pihak yang memerlukannya.

**ABSTRAK**

Keamanan data merupakan suatu kegiatan yang bertujuan untuk mengamankan data terutama data yang bersifat rahasia dan penting. Apabila terjadi suatu pembobolan dan pencurian informasi suatu data penting dalam sebuah file, maka akan merugikan pihak yang berkepentingan. Oleh karena itu bibutuhkan suatu sistem aplikasi untuk dapat mengamankan data file tersebut. Kriptografi merupakan suatu teknik pengamanan yang dapat digunakan. Pada penelitian ini, penulis membangun suatu sistem keamanan data file menggunakan metode affine cipher dan dikombinasikan dengan RSA yang menghasilkan sebuah kriptografi hybrid. Affine cipher merupakan kriptografi simetrik yang lebih cepat dalam proses enkripsi dan dekripsi dibandingankan dengan kriptografi asimetrik. Dan RSA merupakan algoritma asimetrik yang teruji sebagai sistem kriptografi yang aman karena kesulitan dalam proses memfaktorkan bilangan yang sangat besar. Pada penelitian ini enkripsi dilakukan pada bit data file yang menyebabkan struktur file teracak sehingga file tidak dapat dibuka dan dibaca tanpa adanya proses dekripsi terlebih dahulu. rata-rata waktu enkripsi adalah 113 *millisecond*, Sedangkan rata-rata waktu dekripsi adalah 150 *millisecond.* Enkripsi dapat dilakukan pada semua jenis file.

Kata kunci:

Enkripsi, Dekripsi, Affine Cipher, RSA, Kriptografi Hybrid, Pengamanan Data.

**DESIGN APPLICATION OF DATA FILE SECURITY WITH AFFINE CIPHER AND RSA METHOD COMBINATION**

**ABSTRACT**

Data security is an activity that aims to secure of data, especially the secret and important one. When pierching and robbing occurs to the to the information of important data file, it will be harmfull to the parties having interest on it. Therefore, an application system to secure the data file is heavily needed. Cryptography is a security techniques that can be used. In this research, the authors builds a data file securit y system using affine cipher method combined with RSA that produces a hybrid cryptography. Affine cipher is a symmetric cryptographic with faster encryption and decryption process than asymmetric cryptography. And RSA is a asymmetric algorithm that has been guaranteed as a secure cryptographic because of difficulties to process of factoring the large numbers. In this research, encryption does on the bit data of file that causes the file structure is to be messed up. so that files can’t be opened and can’t be read without decryption process before. The average duration of encryption is 113 millisecond whereas average duration of decryption is 150 millisecond. Encryption can be performed on all types of files.

Keyword:

Encryption, Decryption, Affine Cipher, RSA, Hybrid Cryptography, Data Security.

**DAFTAR ISI**

Hal. Persetujuan ii Pernyataan iii

Ucapan Terima Kasih iv

Abstrak v Abstract vi Daftar Isi vii Daftar Tabel ix Daftar Gambar x

BAB 1 Pendahuluan 1

1.1. Latar Belakang 1

1.2. Rumusan Masalah 3

1.3. Tujuan Penelitian 3

1.4. Batasan Masalah 4

1.5. Manfaat Penelitian 4

1.6. Metodologi Penelitian 4

1.7. Sistematika Penulisan 5

BAB 2 Landasan Teori 6

2.1. Keamanan dan Kerahasiaan Data 6

2.2. Aspek Keamanan Komputer 7

2.3. Kriptografi 7

*2.3.1. Fungsi Hash* 10

*2.3.2. Algoritma Simetris* 10

2.3.3. *Algoritma Asimetris* 10

2.3.4. *Algoritma Hibryd* 11

*2.3.4.1. Proses Algoritma Hybrid* 12

2.4. Jenis Serangan Dalam Kriptografi 13

2.5. Algoritma RSA (Rivest, Shamir, Adleman) 15

|  |  |
| --- | --- |
| 2.6. Algoritma Affine Cipher | 18 |
| 2.7. Kode ASCII | 21 |
| 2.8. Penelitian Terdahulu | 21 |

|  |  |
| --- | --- |
| BAB 3 Analisis dan Perancangan | 23 |
| 3.1. Arsitektur Umum | 23 |
| 3.2. Analisis Sistem | 24 |
| 3.2.1. Proses Enkripsi dan Dekripsi *Hybrid Affine Cipher dan RSA* | 25 |
| 3.2.2. Implementasi Algoritma Terhadap *File* | 28 |
| 3.2.3. *Flowchart* Sistem | 29 |
| 3.3. Perancangan Sistem | 32 |
| 3.3.1. Kebutuhan Perangkat Lunak | 32 |
| 3.3.2. Perancangan Antarmuka Sistem | 32 |
| 3.4. Perancangan Database | 35 |
| 3.4.1. Perancangan Tabel | 35 |
| 3.4.2. Relasi Antar Tabel | 36 |
| BAB 4 Implementasi dan Pengujian | 37 |
| 4.1. Spesifikasi Perangkat Lunak dan Perangkat Keras | 37 |
| 4.2. Tampilan Sistem Secara Menyeluruh | 38 |
| 4.3. Pengujian Sistem Secara Menyeluruh | 41 |
| 4.4. Analisis Kriptografi Terhadap Sistem | 46 |
| 4.5. Analisiss Waktu Proses Enkripsi dan Dekripsi *File* | 52 |
| BAB 5 Kesimpulan dan Saran | 54 |
| 5.1. Kesimpulan | 54 |
| 5.2. Saran | 55 |
| Daftar Pustaka | 56 |
| Lampiran: | 58 |

**DAFTAR TABEL**

Hal. Tabel 2.1. perhitungan enkripsi metode affine cipher 20

Tabel 2.2. perhitungan m-1 affine cipher 20

Tabel 2.3. Perhitungan dekripsi affine cipher 20

Tabel 3.1. Perhitungan kunci privat “d” RSA 26

Tabel 3.2. perhitungan m-1 affine cipher pada contoh hybrid 27

Tabel 4.1. Pengujian waktu proses enkripsi dan dekripsi 52

**DAFTAR GAMBAR**

|  |  |
| --- | --- |
| Gambar 2.1. Proses Enkripsi | Hal.  8 |
| Gambar 2.2. Proses Dekripsi | 8 |
| Gambar 2.3. Proses Enkripsi dan Dekripsi Kriptografi Simetris | 10 |
| Gambar 2.4. Proses Enkripsi dan Dekripsi Kriptografi Asimetris | 11 |
| Gambar 3.1.1.Arsitektur Umum Enkripsi yang diajukan | 23 |
| Gambar 3.1.2.Arsitektur Umum Dekripsi yang diajukan | 24 |
| Gambar 3.1.3 Proses Enkripsi dan Dekripsi pada *file* | 28 |
| Gambar 3.2.1 Proses enkripsi pada sistem | 30 |
| Gambar 3.2.2 Proses dekripsi pada sistem | 31 |
| Gambar 3.4.2.1 Rancangan Tampilan Menu *Login* | 32 |
| Gambar 3.4.2.1 Rancangan Tampilan Menu *Register* | 33 |
| Gambar 3.4.2.1 Rancangan Tampilan Menu Enkripsi | 34 |
| Gambar 3.4.2.2 Rancangan Tampilan Menu Dekripsi | 34 |
| Gambar 3.5.1 Tabel Data *user* | 35 |
| Gambar 3.5.2. Tabel data *files* | 36 |
| Gambar 3.5.3. *Database Relationship* | 36 |
| Gambar 4.2.1. Tampilan Menu *Login* | 38 |
| Gambar 4.2.2. Tampilan Form *Register Form* | 38 |
| Gambar 4.2.3. Tampilan *Database data\_user* | 39 |
| Gambar 4.2.4. Tampilan *Database data\_file* | 39 |
| Gambar 4.2.5. Tampilan Menu Enkripsi | 40 |
| Gambar 4.2.6. Tampilan Menu Dekripsi | 40 |
| Gambar 4.2.7. Tampilan Menu *backup* | 41 |

|  |  |  |
| --- | --- | --- |
| Gambar 4.3.1.1 | Login *User* | 42 |
| Gambar 4.3.1.2 | Melakukan enkripsi terhadap sebuah *file* | 42 |
| Gambar 4.3.1.3 | Proses enkripsi selesai | 43 |
| Gambar 4.3.1.4 | *File* asli dan *file* enkripsi | 43 |
| Gambar 4.3.1.5 | *File* yang telah dienkripsi dan tidak dapat dibuka/dibaca | 44 |
| Gambar 4.3.1.6 | Backup *file* enkripsi | 44 |
| Gambar 4.3.1.7 | Melakukan dekripsi | 45 |
| Gambar 4.3.1.8 | Proses akhir dekripsi | 45 |
| Gambar 4.3.1.9 | *File* yang telah didekripsi kembali | 46 |
| Gambar 4.4.1. | *File* “BAB 1.docx” dalam heksadesimal | 47 |
| Gambar 4.4.2. | *File* “BAB 1\_SECURE.docx” dalam heksadesimal | 48 |
| Gambar 4.4.3. | *File* “BAB 1.docx” hasil dekripsi dalam heksadesimal | 49 |
| Gambar 4.4.4. | *File* tidak dapat di enkripsi lebih dari sekali | 49 |
| Gambar 4.4.5. | menggunakan *user* yang salah | 50 |
| Gambar 4.4.6. | Menampilkan *backup file* | 51 |

**ABSTRAK**

Keamanan data merupakan suatu kegiatan yang bertujuan untuk mengamankan data terutama data yang bersifat rahasia dan penting. Apabila terjadi suatu pembobolan dan pencurian informasi suatu data penting dalam sebuah file, maka akan merugikan pihak yang berkepentingan. Oleh karena itu bibutuhkan suatu sistem aplikasi untuk dapat mengamankan data file tersebut. Kriptografi merupakan suatu teknik pengamanan yang dapat digunakan. Pada penelitian ini, penulis membangun suatu sistem keamanan data file menggunakan metode affine cipher dan dikombinasikan dengan RSA yang menghasilkan sebuah kriptografi hybrid. Affine cipher merupakan kriptografi simetrik yang lebih cepat dalam proses enkripsi dan dekripsi dibandingankan dengan kriptografi asimetrik. Dan RSA merupakan algoritma asimetrik yang teruji sebagai sistem kriptografi yang aman karena kesulitan dalam proses memfaktorkan bilangan yang sangat besar. Pada penelitian ini enkripsi dilakukan pada bit data file yang menyebabkan struktur file teracak sehingga file tidak dapat dibuka dan dibaca tanpa adanya proses dekripsi terlebih dahulu. rata-rata waktu enkripsi adalah 113 *millisecond*, Sedangkan rata-rata waktu dekripsi adalah 150 *millisecond.* Enkripsi dapat dilakukan pada semua jenis file.

Kata kunci:

Enkripsi, Dekripsi, Affine Cipher, RSA, Kriptografi Hybrid, Pengamanan Data.

**DESIGN APPLICATION OF DATA FILE SECURITY WITH AFFINE CIPHER AND RSA METHOD COMBINATION**

**ABSTRACT**

Data security is an activity that aims to secure of data, especially the secret and important one. When pierching and robbing occurs to the to the information of important data file, it will be harmfull to the parties having interest on it. Therefore, an application system to secure the data file is heavily needed. Cryptography is a security techniques that can be used. In this research, the authors builds a data file securit y system using affine cipher method combined with RSA that produces a hybrid cryptography. Affine cipher is a symmetric cryptographic with faster encryption and decryption process than asymmetric cryptography. And RSA is a asymmetric algorithm that has been guaranteed as a secure cryptographic because of difficulties to process of factoring the large numbers. In this research, encryption does on the bit data of file that causes the file structure is to be messed up. so that files can’t be opened and can’t be read without decryption process before. The average duration of encryption is 113 millisecond whereas average duration of decryption is 150 millisecond. Encryption can be performed on all types of files.

Keyword:

Encryption, Decryption, Affine Cipher, RSA, Hybrid Cryptography, Data Security.

**BAB 1**

**PENDAHULUAN**

Bab ini membahas tentang hal-hal yang menjadi latar belakang pembuatan tugas akhir, rumusan masalah, tujuan, batasan masalah, manfaat, metodologi penelitian serta sistematika penulisan tugas akhir.

**1.1. Latar Belakang**

Dewasa ini, keamanan suatu data atau informasi merupakan hal yang sangat penting, seiring dengan semakin majunya dunia teknologi informasi, semakin banyak pula orang yang sengaja mencoba mengacak-acak bahkan mengubah dan mengambil informasi dari suatu data *file* tanpa seizin pemiliknya. Setiap individu maupun kelompok kerja biasanya pasti memiliki suatu informasi atau data penting yang bersifat rahasia, tidak boleh diketahui oleh orang lain. sehingga perlunya suatu pengamanan agar data suatu *file* tidak disalahgunakan oleh orang lain. Oleh karena itu dikembangkan suatu pengamanan data yang dikenal dengan kriptografi.

Beberapa metode telah banyak digunakan dalam mengamankan data di antaranya Hamzah (2011) menggunakan metode RSA dan *blowfish* untuk enkripsi dan dekripsi data menghasilkan algoritma RSA lebih cepat dalam prosesnya dibandingkan *blowfish*. RSA merupakan algoritma yang cocok dalam pengamanan data, karena algoritma ini melakukan pemfaktoran bilangan yang sangat besar, RSA dianggap aman karena susah dalam pemecahan sandinya dan *lock foldernya* belum mengimplementasikan RSA sendiri karena tidak mempunyai *public key* dan *private key* (Lestari.2013). model hibrida memberikan *linearitas* yang lebih baik seperti AES dan yang digabung dengan DES, ada *difusi* yang lebih baik. Oleh karena itu kemungkinan serangan aljabar pada model *hybrid* berkurang. Metode *hybrid*

melibatkan lebih banyak perhitungan dibandingkan dengan AES atau DES saja. (Chauhan et al, 2013). Menggunakan algoritma enkripsi Rivest Code 5 dalam mengamankan data *file*, pada penelitian ini enkripsi dilakukan pada struktur bit data pada *file*. Sehingga data pada *file* tersebut tidak bisa dibuka dan dibaca orang lain sebelum melakukan dekripsi terlebih dahulu, Karena struktur bit pada *file* tersebut telah terenkripsi. Hal ini dibuktikan dengan tampilan struktur *file* asli dan *file* terenkripsi berbeda dalam heksadesimal (Suryawan, S.H & Hamdani, 2013)

Keunggulan dari *affine cipher* sebagai sandi subtitusi adalah dilakukannya pertukaran huruf-huruf yang akan disandikan dengan huruf-huruf itu sendiri, sehingga keteraturan huruf-huruf tersebut menjadi acak. Kekuatan cipher ini terletak pada kunci yaitu nilai integer yang menunjukkan pergeseran karakter-karakter, kekuatan kedua terletak pada barisan bilangan-bilangan yang berfungsi sebagai pengali dan kunci. Barisan bilangan tersebut dapat berupa bilangan tertentu seperti deret bilangan ganjil, deret bilangan genap, deret Fibonacci, deret bilangan prima, serta deret bilangan yang dapat dibuat sendiri. Namun, yang paling utama *affine cipher* sebagai sandi subtitusi juga memiliki kekuatan *konfusi*, yaitu memiliki kemampuan mengaburkan hubungan antara *plaintext* dan *ciphertext*, hal ini menimbulkan kesulitan dalam usaha *kriptanalis* mencari keteraturan. Sedangkan kelemahannya sendiri adalah *ciphertext-only attack,* mudah di serang dengan hanya melihat *ciphertext-*nya. Dan *Exhaustive key search*, yaitu serangan terhadap kuncinya.

Algoritma Rivest Shamir Adleman dibuat oleh 3 orang peneliti yaitu Ron Rivest, Adi Shamir dan Leonard Adleman. Nama RSA juga di ambil dari ketiga peneliti ini. RSA merupakan algoritma kriptografi yang paling sering digunakan karena kehandalannya dan memiliki tingkat keamanan yang cukup tinggi. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci pribadi. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang tepat, maka selama itu pula keamanan algoritma RSA tetap terjamin. Tetapi seperti algoritma asimetris lainnya, RSA lebih lambat dalam melakukan enkripsi daripada algoritma simetris (Benz, 2001). Sehingga, dalam penelitian ini penulis menggunakan RSA hanya untuk memperkuat kunci dari *affine cipher* itu sendiri. Metode *affine cipher* yang penulis lakukan akan menggunakan ukuran bilangan ASCII

yaitu dari 0-255, agar lebih sesuai dipadukan dengan metode modern seperti RSA. Dibandingan dengan metode *affine* klasik yang hanya menggunakan ukuran dari alphabet yaitu 26. Oleh karena itu kombinasi antara *affine cipher* dengan *RSA* akan memberikan penyandian baru dengan cara menggabungkan dua metode yaitu a*ffine cipher* dengan *RSA*, sehingga informasi lebih sulit untuk dipecahkan. RSA digunakan untuk enkripsi kunci agar lebih sulit dipecahkan, dan *affine cipher* dapat mempercepat prosesnya.

Setiap metode kriptografi mempunyai kelebihan dan kelemahannya masing- masing baik dari segi kecepatan *enkripsi* dan *dekripsi* maupun dari segi keamanannya. Kelebihan kriptografi simetris adalah pada segi kecepatan untuk proses enkripsi dan dekripsi yang tinggi, namun memiliki kelemahan dalam segi pendistribusian kuncinya. Sedangkan kelebihan kriptografi asimetris adalah kemudahan dalam pertukaran kunci, namun lemah dalam segi kecepatannya. Untuk mengatasi kelemahan masing-masing algoritma kriptografi tersebut, maka dipadukan kedua sistem algoritma kriptografi tersebut. Perpaduan atau penggabungan sistem ini disebut kriptografi *hybrid*. Oleh karena itu, penulis mencoba untuk menggabungan antara metode affine cipher dan RSA agar satu sama lain saling memperkuat dan menutupi kekurangannya dalam suatu pengamanan data, agar dapat meghindari disalahgunakannya data oleh orang- orang yang tidak berhak.

**1.2. Rumusan Masalah**

Keamanan data dalam suatu *file* merupakan suatu hal yang sangat penting bagi seseorang maupun kelompok. Data tersebut ada yang bersifat pribadi, rahasia atau tidak boleh diketahui oleh orang lain. Untuk menghindari adanya pembobolan dan pencurian data, dibutuhkan suatu pengamanan agar *file* tidak dapat diakses dan disalahgunakan oleh orang lain yang tidak berhak.

**1.3. Tujuan Penelitian**

Adapun tujuan dari penelitian ini adalah untuk mengunci dan menjaga keamanan data file menggunakan kombinasi metode *affine cipher* dan RSA.

**1.4. Batasan Masalah**

Permasalahan dibatasi mencakup hal-hal berikut:

1. Besar *file* maksimum sampai dengan 65 MB.

2. Pengamanan tidak dilakukan pada folder

3. Enkripsi dan dekripsi dilakukan pada struktur *bit* masing-masing *file*.

**1.5. Manfaat Penelitian**

Manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Membantu mengamankan *file*

2. Menjadi referensi dalam bidang kriptografi khususnya pengamanan data file menggunakan penggabungan antara *affine cipher* dan RSA.

**1.6. Metodologi Penelitian**

Tahapan-tahapan yang dilakukan pada pelaksanaan penelitian ini adalah sebagai berikut.

1. Studi Literatur

Studi literatur dilakukan dalam rangka pengumpulan bahan referensi mengenai

Pengamanan data, enkripsi, dekripsi, *affine cipher*, dan algoritma RSA.

2. Analisis Permasalahan

Pada tahap ini dilakukan analisis terhadap bahan referensi yang telah dikumpulkan pada tahap sebelumnya untuk mendapatkan pemahaman mengenai metode yang diterapkan yakni algoritma *hybrid affine cipher dan RSA*, serta domain masalah yang akan diselesaikan yakni masalah enkripsi dan pengamanan data.

3. Pengumpulan Data

Pada tahap ini dilakukan beberapa macam data *file* baik itu berupa data document word doc, docx, pdf, txt, ppt, gambar, mp3, video dan lainnya yang nantinya akan dienkripsi.

4. Implementasi

Pada tahap ini dilakukan implementasi algoritma *affine cipher dan RSA* dalam penyelesaian masalah pengamanan data *file* menggunakan data yang telah dikumpulkan sebelumnya.

5. Evaluasi dan Analisis Hasil

Pada tahap ini dilakukan evaluasi serta analisis terhadap hasil yang didapatkan melalui implementasi algoritma *affine cipher dan RSA* dalam penyelesaian masalah pengamanan data *file*.

6. Dokumentasi dan Pelaporan

Pada tahap ini dilakukan dokumentasi dan penyusunan laporan hasil evaluasi dan analisis serta implementasi algoritma *affine cipher dan RSA* dalam pengamanan data.

**1.7. Sistematika Penulisan**

Tugas akhir ini disusun dalam lima bab dengan sistematika penulisan sebagai berikut: Bab I Pendahuluan

Bab pendahuluan ini berisi tentang hal-hal yang mendasari dilakukannya penelitian serta pengidentifikasian masalah penelitian. Bagian-bagian yang terdapat dalam bab pendahuluan ini meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, dan manfaat penelitian.

Bab II Tinjauan Pustaka

Pada bab tinjauan pustaka menguraikan landasan teori, penelitian terdahulu, kerangka pikir dan hipotesis yang diperoleh dari acuan yang mendasari dalam melakukan kegiatan penelitian pada tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini membahas analisis dan penerapan algoritma *affine cipher dan RSA* untuk melakukan pengamanan pada data. Pada bab ini dijabarkan arsitektur umum, proses yang dilakukan serta cara kerja algoritma yang digunakan.

Bab IV Implemetasi dan Pengujian

Pada bab hasil dan pembahasan akan memaparkan implementasi sistem dan hasil terhadap uji coba algoritma yang telah dilakukan dalam meyelesaikan permasalahan pengamanan data.

Bab V Kesimpulan dan Saran

Bab ini berisi tentang kesimpulan hasil penelitian dan saran-saran yang berkaitan dengan penelitian selanjutnya.

**BAB 2**

**LANDASAN TEORI**

Pada bab ini, akan dibahas landasan teori, penelitian terdahulu, kerangka pikir dan hipotesis yang mendasari penyelesaian permasalahan pengamanan data *file* dengan kombinasi algoritma *affine cipher* dan RSA.

**2.1. Keamanan dan Kerahasiaan Data**

Masalah keamanan dan kerahasiaan data merupakan salah satu aspek penting dari suatu sistem informasi. Dalam hal ini, sangat terkait dengan betapa pentingnya informasi tersebut diterima oleh orang yang berkepentingan. Informasi akan menjadi tidak *valid* lagi jika informasi tersebut di ketahui atau dibajak oleh orang yang tidak berhak .

Informasi atau data saat ini sudah menjadi sesuatu yang sangat penting bagi sebuah organisasi, perguruan tinggi, lembaga pemerintahan maupun individual, termasuk kemampuan dalam mengakses dan menyediakan informasi secara cepat serta akurat. Karena pentingnya sebuah informasi, seringkali informasi yang diinginkan hanya dapat diakses oleh orang tertentu misalnya pihak penerima yang diinginkan, dan jika informasi ini sampai diterima oleh pihak yang tidak diinginkan akan berdampak kerugian pada pihak pengirim. Keamanan informasi adalah bagaimana kita dapat mencegah penipuan, atau paling tidak mendeteksi adanya penipuan di sebuah sistem yang berbasis informasi, untuk melindungi informasi dari pengaksesan, penggunaan, penyebaran, perusakan, perubahan, dan penghancuran tanpa otorisasi yang sah. Untuk itu diperlukan sebuah pendekatan dalam melakukan pengamanan pada informasi, seperti melakukan enkripsi, *steganografi*, *cipher* dan *hashing* terhadap informasi tersebut.

Identifikasi dan otentikasi pengguna adalah kebutuhan dasar untuk menjamin keamanan karena metode identifikasi mendefinisikan orang-orang yang diizinkan

untuk mengakses data dan menyediakan mekanisme lengkap dalam aksesibilitas. Untuk menjamin keamanan, identitas harus dikonfirmasi dan itu membuat data sensitif lebih aman dari modifikasi oleh pengguna biasa. (Basharat et al, 2012)

**2.2. Aspek Keamanan Komputer**

Keamanan informasi adalah bagaimana kita dapat mencegah penipuan, atau paling tidak mendeteksi adanya penipuan di sebuah sistem yang berbasis informasi, untuk melindungi informasi dari pengaksesan, penggunaan, penyebaran, perusakan, perubahan, dan penghancuran tanpa otorisasi yang sah. Untuk itu diperlukanlah sebuah pendekatan dalam melakukan pengamanan pada informasi, seperti melakukan enkripsi, *steganografi*, *cipher* dan *hashing* terhadap informasi tersebut.

Aspek keamanan komputer adalah bentuk pertimbangan yang menyatakan sebuah komputer bisa dinyatakan aman. Aspek-aspek keamanan di dalam kriptografi adalah sebagai berikut (Munir, 2006):

1. *Confidentiality* (kerahasiaan) yaitu merupakan usaha untuk menjaga informasi dari orang yang tidak berhak mengakses.

2. *Authentication* (otentikasi) yaitu agar penerima informasi dapat memastikan keaslian pesan tersebut datang dari orang yang dimintai informasi

3. *Non-repudiation* (nir penyangkalan) yaitu merupakan hal yang yang bersangkutan dengan si pengirim, si pengirim tidak dapat mengelak bahwa dia lah yang mengirim informasi tersebut.

4. *Integrity* (data Integritas) yaitu keaslian pesan yang dikirim melalui sebuah jaringan dan dapat di pastikan bahwa informasi yang dikirim tidak di modifikasi oleh orang yang tidak berhak dalam perjalanan informasi tersebut.

**2.3. Kriptografi**

Menurut Kromodimoeljo (2010), “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”. Dekripsi menggunakan kunci dekripsi untuk mendapatkan kembali data asli. Proses enkripsi dilakukan

menggunakan suatu algoritma dengan beberapa parameter. Secara umum kriptografi

terdiri dari proses enkripsi dan dekripsi dan terlihat pada gambar 2.1 dan 2.2 berikut:

Plaintext

Enkripsi

Ciphertext

**Gambar 2.1 Proses Enkripsi**



**Gambar 2.2 Proses Dekripsi**

Pada gambar 2.1 proses enkripsi dinyatakan dengan *plaintext* di *enkripsi* menghasilkan *ciphertext*, kemudian pada gambar 2.2 proses *dekripsi* dinyatakan dengan *ciphertext* di *dekripsi* menghasilkan *plaintext*. Berikut penjelasan istilah-istilah yang terdapat dalam proses tersebut:

*Plaintext* merupakan pesan yang akan dirahasiakan, dinotasikan dengan **m** (message), yang dapat berupa bit stream, text, file text, image atau dapat dikatakan **m** merupakan data *binary*.

*Enkripsi* merupakan proses pengamanan data agar data atau informasi tidak dapat diketahui dan dibaca oleh orang-orang yang tidak berhak. Enkripsi dinotasikan dengan **E**, yang berfungsi untuk mengubah m menjadi c. Dalam matematika di rumuskan

sebagai berikut (Mollin, 2007) :

Keterangan rumus:

**E** = enkripsi

**e** = kunci enkripsi

**m** = message

**c** = ciphertext

**Ee(m) = c** (1)

*Ciphertext* merupakan hasil dari pengkodean atau enkripsi, di notasikan dengan **c**, juga berupa data binary yang kadang-kadang mempunyai ukuran yang sama dengan **m**, lebih kecil dari m atau lebih besar dari **m**.

*Dekripsi* merupakan kebalikan dari proses enkripsi yaitu proses mengubah data menjadi dapat dibaca dan diketahui informasinya. Fungsi dekripsi **D**, berfungsi untuk mengubah **c** menjadi **m**. Dalam matematika dirumuskan sebagai berikut:

**Dd(c) = m** (2)

Keterangan rumus:

**D** = dekripsi

**c** = ciphertext

**m** = message

**d** = kunci dekripsi

“Kriptografi terbagi atas dua kategori yaitu kriptografi klasik dan kriptografi modern. Kriptografi klasik adalah sistem kriptografi yang menggunakan penyandian kunci simetris dan menggunakan metode subtitusi (pergantian huruf) atau transposisi (pertukaran tempat). Dan kriptografi modern adalah sistem kriptografi yang menggunakan penyandian kunci asimetris” (Sadikin, 2012).

Dalam kriptografi klasik, teknik enkripsi yang digunakan adalah enkripsi simetris dimana kunci dekripsi sama dengan kunci enkripsi. Untuk *public key cryptography*, diperlukan teknik enkripsi asimetris dimana kunci dekripsi tidak sama dengan kunci enkripsi. Enkripsi, dekripsi dan pembuatan kunci untuk teknik enkripsi asimetris memerlukan komputasi yang lebih intensif di bandingkan enkripsi simetris, karena enkripsi asimetris menggunakan bilangan-bilangan yang sangat besar. Namun, walaupun enkripsi asimetris lebih “mahal” dibandingkan enkripsi simetris, *public key cryptography* sangat berguna untuk *key management* dan *digital signature*. (Kromodimoeljo, 2010).

Berikut ini adalah beberapa mekanisme yang berkembang pada kriptografi:

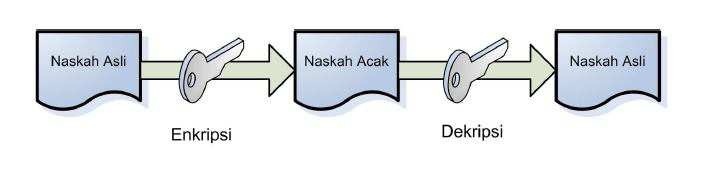
**2.3.1. Fungsi *hash***

Fungsi *hash* adalah fungsi yang melakukan pemetaan pesan dengan panjang sembarang ke sebuah teks khusus yang disebut *message digest* dengan panjang yang tetap. Fungsi *hash* pada umumnya digunakan untuk menguji keutuhan pada sebuah data.

**2.3.2. Algoritma simetris**

Penyandian dengan kunci simetris adalah penyandian yang pada proses enkripsi dan dekripsinya menggunakan sebuah kunci yang sama. Kunci yang digunakan pada proses enkripsi dan dekripsi bersifat rahasia. Beberapa jenis kriptografi dengan penyandian kunci simetris antara lain Data Encryption Standard (DES), 3DES, Advanced Encryption Standard (AES), Blowfish dan International Data Encryption Algorithm (IDEA). Adapun proses enkripsi dan dekripsi algoritma simetris adalah sebagai berikut :

Pada gambar 2.3 dinyatakan bahwa pada kriptografi simetris, pesan/naskah asli di enkripsi dan di dekripsi menggunakan kunci yang sama.



Gambar 2.3:

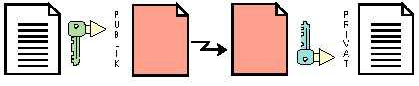
Proses Enkripsi dan Dekripsi Kriptografi Simetris (Kromodimoeljo, 2010)

**2.3.3. Algoritma Asimetris**

Penyandian dengan kunci asimetris atau sering juga disebut dengan kunci publik adalah penyandian yang pada proses enkripsi dan dekripsinya memiliki kunci yang berbeda. Pada kunci yang digunakan untuk enkripsi bersifat publik (*public key*). Sedangkan kunci yang digunakan untuk dekripsi bersifat rahasia (*private key*). Beberapa jenis kriptografi dengan penyandian kunci asimetris antara lain RSA, Diffie-

Helman, ELGamal, Knapsack, dan beberapa penerapan teknik Elliptic Curve. “Dengan cara seperti ini, jika Anto mengirim pesan untuk Badu, Anto dapat merasa yakin bahwa pesan tersebut hanya dapat dibaca oleh Badu, karena hanya Badu yang bisa melakukan dekripsi dengan kunci privatnya. Tentunya Anto harus memiliki kunci publik Badu untuk melakukan enkripsi. Anto bisa mendapatkannya dari Badu, ataupun dari pihak ketiga seperti Tari” (Wahyuni, 2011).

Teknik enkripsi asimetris ini jauh lebih lambat daripada enkripsi dengan kunci simetris. Oleh karena itu, biasanya bukanlah pesan itu sendiri yang disandikan dengan kunci asimetris, namun hanya kunci simetrislah yang disandikan dengan kunci asimetris. Sedangkan pesannya dikirim setelah disandikan dengan kunci simetris tadi. “Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA (Rivest Shamir Adleman). Adapun proses enkripsi dan dekripsi algoritma asimetris dapat dilihat pada gambar 2.4. sebagai berikut:



Gambar 2.4:

Proses Enkripsi dan Dekripsi Kriptografi Asimetris (Wahyuni, 2011)

Pada gambar 2.4 dinyatakan bahwa pada kriptografi asimetris, pesan di enkripsi menggunakan kunci publik, kemudian untuk dapat membuka pesan harus di dekripsi menggunakan kunci privat.

**2.3.4. Algoritma *Hybrid***

Metode kriptografi dibagi menjadi kriptografi simetris dan kriptografi asimetris. Perbedaan yang paling mendasar pada algoritma kriptografi ini terletak pada kunci yang digunakan untuk proses pengenkripsian dan pendekripsian. Kriptografi simetris efisien dalam proses enkripsi dekripsi tetapi memiliki kelemahan pada proses pendistribusian kunci sedangkan kriptografi asimetris kurang efisien pada saat proses

enkripsi dekripsi dikarenakan membutuhkan waktu yang lebih lama dibandingkan pada kriptografi simetris*.*

Namun demikian, kriptografi asimetris memiliki keuntungan dimana tidak diperlukan proses distribusi kunci dikarenakan kunci yang digunakan untuk proses enkripsi-dekripsi ditempatkan pada jaringan umum. oleh karena itu pada saat ini untuk meningkatkan mekanisme keamanan, maka kedua sistem kriptografi tersebut digabungkan, perpaduan antar kedua sistem kriptografi ini disebut metode kriptografi

*hybrid*.

kriptografi *hibryd* adalah suatu penggabungan antara kriptografi simetris dan kriptografi asimetris. Masing-masing algoritma memiliki kelebihan dan kelemahan. Kelebihan kriptografi simetris adalah pada segi kecepatan untuk proses enkripsi dan dekripsi yang tinggi, namun memiliki kelemahan dalam segi pendistribusian kuncinya. Sedangkan kelebihan kriptografi asimetris adalah kemudahan dalam pertukaran kunci, namun lemah dalam segi kecepatannya. Untuk mengatasi kelemahan masing-masing algoritma kriptografi tersebut, maka dipadukan kedua sistem algoritma kriptografi tersebut. Perpaduan atau penggabungan sistem ini disebut kriptografi *hibryd*.

Saat ini sangat penting untuk merancang algoritma enkripsi yang kuat sebagai kekuatan komputer berkembang dari hari ke hari. Dengan demikian model *hibrida* memberikan *linearitas* yang lebih baik contohnya seperti AES yang digabung dengan DES, ada *difusi* yang lebih baik. Oleh karena itu kemungkinan serangan aljabar pada model *hybrid* berkurang. Metode *hybrid* melibatkan lebih banyak perhitungan dibandingkan dengan menggunakan satu algoritma saja. (Chauhan et al, 2013)

Menggabungkan kedua *symmetric-key* dan algoritma kunci publik menyediakan lebih besar fitur keamanan dan keunikan yang hanya mungkin terjadi pada sistem *hybrid.* (Praphul, 2013)

**2.3.4.1 Proses Kriptografi *Hybrid***

Proses kriptografi *hybrid* yaitu melakukan proses enkripsi pada *plainteks* terlebih dahulu agar dapat mempertinggi daya tahan terhadap serangan penyerang (*attacker*) yang pada umumnya memanfaatkan pola-pola yang ada pada *plainteks* untuk memecahkan *chiperteks* menggunakan algoritma simetris. Setelah *plainteks* menjadi teks terenkripsi kemudian kunci privat algoritma simetris dienkripsi menggunakan

pasangan kunci publik algoritma asimetris kemudian dikirimkan. Pengenkripsian kunci ini disebut *session key* yaitu merupakan kunci privat yang terenkripsi yang bersifat tercipta hanya pada saat itu juga (*one-time only*). Kunci yang terenkripsi bersama dengan chiperteks kemudian ditransmisikan kepada penerima. Untuk proses dekripsi, penerima menerima paket tersebut menggunakan pasangan kunci privat algoritma asimetris untuk mendekripsi *session key* terlebih dahulu. Lalu dengan *session key* tersebut, kunci privat algoritma simetris dapat dibuka dan penerima dapat mendekripsi *chiperteks* tersebut menjadi *plainteks* kembali. Algoritma yang akan digunakan untuk pengamanan data ini adalah algoritma *Affine Cipher* dan RSA dimana metode ini menggabungkan kelebihan masing-masing algoritma tersebut.

**2.4. Jenis Serangan pada Kriptografi**

Dalam algoritma kriptografi ada beberapa jenis serangan yang terdefenisi, antara lain

(Rudianto, 2007) :

1. *Exhaustive Key Search*

Penyerang mencoba semua kemungkinan kunci satu persatu dan mengecek apakah plainteks memiliki kecocokan dengan chiperteks yang menjadi sampel.

2. *The Matching Chipertext Attack*

Serangan ini didasarkan pada fakta bahwa setidaknya ada sebuah blok chipper berukuran m-bit yang muncul dari hasil enkripsi yang berasal dari 2m/2 blok plainteks sehingga dapat diketahui sedikit informasi mengenai plainteksnya.

3. *Differential Cryptanalysis*

Cara ini merupakan salah satu metode kriptanalisis konvensional yang paling umum dan sering digunakan, yang dipublikasikan oleh Bilham dan Shamir pada tahun 1990. Kriptanalisis ini biasa digunakan untuk melawan metodemetode kriptografi yang dibangun dari perulangan fungsi yang tetap. Ide dibalik metode ini adalah selisih dari plainteks dan chiperteks, yang didapatkan dari hasil kombinasi dengan kunci, selalu sama besarnya.

4. *Truncated Differentials*

Untuk beberapa chiperteks, dimungkinkan dan sangat bermanfaat, memprediksi hanya sebagian nilai saja dengan menggunakan *differential cryptanalysis* untuk setiap putarannya.

5. *Impossible Differential*

Ide utamanya adalah menspesifikkan bahwa ketidak-mungkinan terhadap beberapa putaran terhadap chiper serangan. Kemudian dengan menebak beberapa kunci dalam putaran yang tidak tercakup dalam fungsi, dapat dilakukan pembuangan terhadap beberapa nilai kunci yang salah.

6. *Higher*-*order Differentials*

Sebuah *sth-order differentials* didefenisikan secara rekursif sebagai sebuah fungsi

(*s-1*) *th-order differentials, dimana sth-order differentials* berisi kumpulan 2n teks yang mengandung *predetermined differentials*.

7. *Linear Cryptanalysis*

*Linear cryptanalysis* ditemukan oleh Matsui pada tahun 1993. *Linear cryptanalysis* merupakan serangan *knowplainteks* dimana penyerang mengeksploitasi pendekatan persamaan linier dari beberapa bit plainteks, beberapa bit chiperteks, dan beberapa bit kunci.

8. Kriptanalisis Mod n

Serangan ini merupakan generalisasi dari serangan linier. Serangan ini dapat digunakan untuk chiperteks dimana beberapa kata terbiaskan dalam persamaan modulo n, dimana n adalah integer yang bernilai kecil. Telah terbukti bahwa algoritma kriptografi yang menggunakan hanya rotasi bit dan menambahkan modulo dari 232 sangat rapuh terhadap serangan ini.

9. *Related*-*key attacks*

Knudsen telah menggunakan metode mendapatkan hasil enkripsi menggunakan satu kunci terhadap sebuah plainteks yang terpilih dan berhasil mengurangi kunci secara *exhaustive search* sampai dengan empat (4) kali lebih cepat. Serangan ini membutuhkan hasil enkripsi dari beberapa kunci yang berbeda.

**2.5. Algoritma RSA**

RSA merupakan algoritma kriptografi asimetris. Ditemukan pertama kali pada tahun

1977 oleh Ron Rivest, Adi Shamir, dan Leonard Adleman. Nama RSA sendiri diambil dari inisial nama depan ketiga penemunya tersebut.

RSA adalah salah satu model dan metode enkripsi. Keamanan enkripsi dari RSA cukup baik, hal ini terjadi karena sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. RSA banyak diaplikasikan untuk mengenkripsi teks. Teks merupakan data yang penting dan paling sering digunakan, apalagi jika teks tersebut berisi rahasia penting suatu perusahaan maupun rahasia pribadi seseorang. (Syaputra et al, 2012).

**a. Enkripsi dan Dekripsi Menggunakan RSA**

Kunci pada RSA mencakup dua buah kunci, yaitu *public key* dan *private key. Public key* digunakan untuk melakukan enkripsi, dan dapat diketahui oleh orang lain. Sedangkan *private key* tetap dirahasiakan dan digunakan untuk melakukan dekripsi. Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar. Sampai saat ini RSA masih dipercaya dan digunakan secara luas di internet.

Tingkat keamanan algoritma penyandian RSA sangat bergantung pada ukuran

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

Rumus dasar enkripsi dan dekripsi RSA adalah sbagai berikut (Mollin, 2007) :

ܿ ≡ ݉௘ ( ݉݋݀ ݊) (3)

Dan

݉ ≡ ܿௗ ( ݉݋݀ ݊) (4)

Langkah-langkah pembangkitan kunci pada RSA adalah sebagai berikut:

a. Pilih dua buah bilangan prima sembarang *p* dan *q*. Jaga kerahasiaan *p* dan *q* ini. b. Hitung *n = p \* q*. Besaran *n* ini tidak perlu dirahasiakan.

c. Hitung ɸ ( ݊) *= (p-1) \* (q-1)*. Sekali ɸ ( ݊) telah dihitung, *p* dan *q* dapat

dihapus untuk mencegah diketahuinya oleh pihak lain.

d. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya *e,* yang relatif

prima terhadap ɸ ( ݊) (relatif prima berarti GCD(e, ɸ ( ݊) ) = 1) dengan syarat *e*

≠ *(p-1), e* ≠ *(q-1),* dan *e < n*

e. Hitung kunci dekripsi, *d*, dengan kekongruenan *ed ≡1 (mod* ɸ ( ݊) *)*.

f. Sehingga di dapat pasangan kunci publik (e,n) dan kunci privat (d,n)

Algoritma RSAmemiliki besaran-besaran sbb :

*p* dan *q* bilangan prima (rahasia)

*n = p \* q* (tidak rahasia)

ɸ ( ݊) *= (p-1) \* (q-1)* (rahasia)

*e* = kunci enkripsi (tidak rahasia)

*d* = kunci dekripsi (rahasia)

m = *message* (rahasia)

c = *ciphertext* (tidak rahasia)

Contoh RSA:

 Pilih bilangan prima ݌ dan ݍ misalnya ݌ = 7 dan ݍ = 17



 cari ݊ = ݌ ݍ = 7 × 17= 119

 Hitung ɸ ( ݊) = ( ݌ − 1) ( ݍ − 1) = ( 6) ( 16) = 96

 Pilih ݁ yang relatif prima dengan ɸ( ݊) = 96 dan kurang dari ɸ ( ݊) , dalam hal

ini ݁ = 5

 Tentukan dimana ݁݀ = 1 ݉݋݀ ɸ ( ݊) dan ݀ < ɸ ( ݊) , berarti 5 × ݀ =

1 ݉݋݀ 96, ݀ = 77 karena 5 × 77 = 4 × 96 + 1

 Didapat kunci publik { 5,119} dan kunci privat { 77,119}

Misalkan plaintext M = “F I T R A”, di ubah ke dalam ASCII “70 73 84 82 65”

Enkripsi dengan kunci public ݁ = 5 ݀ܽ݊ ݊ = 119, dapat digunakan rumus enkripsi

(3) sebagai berikut :

ܿ ≡ ݉௘ ( ݉݋݀ ݊)

M1 = 70 -> C1 = 70^5 mod 119 = 49 -> “1”

M2 = 73 -> C2 = 73^5 mod 119 = 82 -> “R” M3 = 84 -> C3 = 84^5 mod 119 = 28 -> “FS” M4 = 82 -> C4 = 82^5 mod 119 = 80 -> “P” M5 = 65 -> C5 = 65^5 mod 119 = 46 -> “.”

Didapat ciphertextnya = 1RFSP. , Kemudian didekripsi menggunakan rumus (4)

sebagai berikut :

݉ ≡ ܿ ௗ ( ݉݋݀ ݊)

Dekripsi dengan kunci ݀ = 77 ݀ܽ݊ ݊ = 119

C1 = 1 -> M1 = 49^77 mod 119 = 70 -> “F”

C2 = R -> M2 = 82^77 mod 119 = 73 -> “I” C3 = FS -> M3 = 28^77 mod 119 = 84 -> “T” C1 = P -> M4 = 80^77 mod 119 = 82 -> “R” C2 = . -> M5 = 46^77 mod 119 = 65 -> “A”

Didapatkan hasil plaintextnya kembali yaitu “FITRA”

***2.6. Affine Cipher***

Affine cipher adalah jenis *monoalphabetik cipher* substitusi, dimana setiap huruf dalam alfabet dipetakan ke setara *numerik*, dienkripsi menggunakan fungsi matematika sederhana, dan diubah kembali ke kata. Setiap huruf dienkripsi dengan fungsi (ax + b) mod 26 di mana b adalah besarnya *shift* atau pergeseran. Dalam affine cipher yang huruf alfabet dari ukuran yang pertama dipetakan ke bilangan bulat dalam kisaran. Ia kemudian menggunakan aritmatika modular untuk mengubah integer yang setiap huruf *plaintext* berkorespondensi untuk menjadi bilangan bulat lain yang sesuai dengan kata ciphertext. ( Shukla et al, 2014).

Dalam kriptografi klasik Modifikasi *Affine cipher* yang diperkuat dengan *Vigenere cipher* merupakan dua kali proses penyandian karena melalui dua kali proses penyandian yang *cipherteks-*nya mempunyai karakteristik dari *cipher* alfabet majemuk. Jika kriptanalisis mengetahui ukuran konversi serta alur enkripsi maupun dekripsi pada modifikasi metode tersebut. maka kriptanalisis akan bekerja keras dalam menemukankan kuncinya (Juliadi et al, 2013).

*Affine cipher* merupakan perluasan dari *caesar cipher*, yang mengalikan *plainteks* dengan sebuah nilai dan menambahkannya dengan sebuah pergeseran. Dengan menggunakan faktor pengali dan substitusi, didapatkan cipherteks hasil enkripsi (Winata, 2012). Pada awalnya pada *affine cipher* klasik menggunakan bilangan abjad a-z sehingga kunci yang digunakan ada 26, Tetap pada penelitian ini menggunakan ukuran dari bilangan ASCII pada perhitungannya agar lebih modern dan cocok digabungkan dengan algoritma RSA.

Secara matematis enkripsi *plainteks* menghasilkan *cipherteks* dinyatakan dengan fungsi sebagai berikut (Winata, 2012):

ܥ( ܲ) ≡ ݉ܲ + ܾ ( ݉݋݀ ݊) (5)

Sementara dekripsi *cipherteks* menjadi *plainteks* dinyatakan sebagai berikut:

ܲ( ܥ) ≡ ݉ିଵ ( ܥ − ܾ) ( ݉݋݀ ݊) (6)

Dimana:

*n* = ukuran bilangan ASCII,

*P* = plainteks yang dikonversi menjadi bilangan bulat sesuai dengan urutan dalam

ASCII

*C* = cipherteks yang dikonversi menjadi bilangan bulat sesuai dengan urutan dalam

ASCII

*m* = bilangan bulat yang harus relatif prima dengan *n* (jika tidak relatif prima, maka dekripsi tidak bisa dilakukan)

*b* = jumlah pergeseran urutan huruf dalam ASCII berjumlah 256

Misalkan Plaintext: K R I P T O G R A F I

 n = 256 (karena bilangan ASCII = 256)

 m = 7, harus relatif prima dengan n buktinya : 256 mod 7 = 4

7 mod 4 = 3

4 mod 3 = 1

3 mod **1** = 0

 misal b = 10

Proses Enkripsi menggunakan rumus (5) sebagai berikut :

ܥ( ܲ) ≡ ( ݉ܲ + ܾ) ( ݉݋݀ ݊)

Terlihat pada tabel 2.1 di bawah ini :

Tabel 2.1. perhitungan enkripsi metode affine cipher

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plaintext | K | R | I | P | T | O | G | R | A | F | I |
| X | 75 | 82 | 73 | 80 | 84 | 79 | 71 | 82 | 65 | 70 | 73 |
| 7X+10 | 535 | 584 | 521 | 570 | 598 | 563 | 507 | 584 | 465 | 500 | 521 |
| (7X+10) mod  256 | 23 | 72 | 9 | 58 | 86 | 51 | 251 | 72 | 209 | 244 | 9 |
| Ciphertext | ETB | H | TAB | : | V | 3 | x | H | Ň | Ô | TAB |

Proses Dekripsi menggunakan rumus (6) sebagai berikut :

ܲ( ܥ) ≡ ݉ିଵ ( ܿ − ܾ) ( ݉݋݀ ݊)

Mula-mula hitung m-1 pada tabel 2.2 berikut:

Tabel 2.2. Perhitungan m-1 affine cipher

Di dapat nilai m-1 = 183, maka proses dekripsinya terlihat pada tabel 2.3. berikut : Tabel 2.3. Perhitungan dekripsi affine cipher

|  |  |
| --- | --- |
| m-1 | m-1 m (mod n)  m-1.7 (mod 256) |
| 1  2  3  4  5  .  .  .  183 | 7  14  21  2  9  .  .  .  **1** |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ciphertext | ETB | H | TAB | : | V | 3 | x | H | Ň | Ô | TAB |
| X | 23 | 72 | 9 | 58 | 86 | 51 | 251 | 72 | 209 | 244 | 9 |
| 183(X-10) | 2379 | 11346 | -183 | 8784 | 13908 | 7503 | 44103 | 11346 | 36417 | 42822 | -183 |
| 183(X-10) mod 256 | 75 | 82 | 73 | 80 | 84 | 79 | 71 | 82 | 65 | 70 | 73 |
| Plaintext | K | R | I | P | T | O | G | R | A | F | I |

**2.7. Kode ASCII**

ASCII (*American Standard Code for Information Interchange*) digunakan dalam pertukaran informasi dan komunikasi data. ASCII merupakan kode angka yang mewakili sebuah karakter, karena komputer hanya dapat membaca angka-angka maka kode ASCII bertugas merepresentasikan karakter tersebut seperti ‘a’ atau ‘@’ begitu juga *non-printed character*.Gambar 3.2 menyajikan karakter yang memiliki kode ASCII dan juga menyajikan 32 buah *non-printed character*. Kode ASCII memiliki beberapa basis dalam merepresentasikan karakter, diantaranya dalam bilangan desimal, heksadesimal, oktal dan html.

**2.8. Penelitian Terdahulu**

Penelitian terhadap pengamanan data dan *file* sudah pernah dilakukan. Diantaranya yaitu Suryawan, S.H dan Hamdani (2013) melakukan penelitian dengan judul Pengamanan data *file* menggunakan algoritma enkripsi Rivest Code 5 (RC5), dimana pada penelitian ini enkripsi dilakukan pada struktur bit data pada file. Sehingga data pada file tersebut tidak bisa dibuka dan dibaca orang lain sebelum melakukan dekripsi terlebih dahulu, Karena struktur bit pada file tersebut telah terenkripsi. Hal ini dibuktikan dengan tampilan struktur file asli dan file terenkripsi berbeda dalam heksadesimal.

Kemudian menggunakan algoritma *Twofish* dimana *Twofish* menggunakan

128 bit setiap blok yang akan dienkripsi. Kunci yang digunakan dengan panjang maksimal 256 bit dengan menggunakan kunci simetrik. dalam penelitian ini menghasilkan bahwa aplikasi dapat mengamankan file dan folder dimana susunan dari file tersebut berbeda dengan file aslinya, ditampilkan dalam bentuk heksadesimal (Fanidzar. 2010).

Selanjutnya implementasi algoritma RSA dan Blowfish untuk enkripsi dan

dekripsi data menggunakan Delphi 7 menghasilkan bahwa algoritma RSA lebih cepat dibandingkan dengan Blowfish (Hamzah.2011).

Kemudian dengan menggunakan algoritma RSA (Rivest Shamir Adleman) dalam sistem enkripsi file dan pengamanan folder menghasilkan bahwa algoritma RSA merupakan algoritma yang cocok dalam pengamanan data, karena algoritma ini melakukan pemfaktoran bilangan yang sangat besar, RSA dianggap aman karena

susah dalam pemecahan sandinya dan lock foldernya belum mengimplementasikan

RSA sendiri karena tidak mempunyai public key dan private key (Lestari.2013).

Pada penelitian Sitinjak, 2010 yang berjudul Aplikasi Kriptografi *file* menggunakan algoritma Blowfish membuktikan bahwa aplikasi dapat mengamankan data ataupun informasi yang berupa *file (plainteks)* dengan mengacak *file* tersebut sehingga tidak dapat dibaca atau dimengerti. Aplikasi ini juga telah berhasil mengembalikan *file* yang telah diacak tersebut *(cipherteks)* seperti semula dengan menggunakan kunci yang sama sewaktu enkripsi. Kecepatan proses enkripsi/dekripsi bergantung pada besarnya ukuran *file*, semakin besar ukuran *file* semakin banyak waktu yang dibutuhkan untuk enkripsi/dekripsi. (Sitinjak, et al, 2010).

Penelitian Chauhan, et al tahun 2013 yang berjudul *Enhancing Data Security by using Hybrid Cryptographic Algorithm* menghasilkan bahwa model hibrida memberikan *linearitas* yang lebih baik seperti AES dan yang digabung dengan DES, ada *difusi* yang lebih baik. Oleh karena itu kemungkinan serangan aljabar pada model *hybrid* berkurang. Metode *hybrid* melibatkan lebih banyak perhitungan dibandingkan dengan AES atau DES saja.

**BAB 3**

**ANALISIS DAN PERANCANGAN**

Bab ini membahas tentang implementasi metode *affine cipher* dan RSA untuk melakukan enkripsi dan dekripsi pada data file. Bab ini juga membahas tentang arsitektur umum sistem yang dibangun.

**3.1 Arsitektur Umum**

 Arsitektur Umum pada Proses Enkripsi dapat di lihat pada gambar 3.1.1 berikut :



*affine*, sehingga *cipherfile* dan *cipherkey* ini menghasilkan *file\_secure* yang dikunci dengan aman.

 Arsitektur Umum pada Proses Dekripsi dapat di lihat pada gambar 3.1.2. berikut :



**3.2.1 Proses Enkripsi dan Dekripsi *Hybrid* Affine Cipher dan RSA**

Proses enkripsi dan dekripsi kombinasi affine cipher dan RSA secara manual adalah sebagai berikut :

**a. Proses Enkripsi**

Contoh :

1. Diberikan sebuah *plaintext* “AB” kemudian *plaintext* diubah dalam bentuk

ASCII yaitu 65 dan 66.

2. *Plaintext* “AB” di enkripsi untuk mendapatkan *ciphertext*-nya yaitu dengan cara sbb:

Menentukan kunci *affine* (m,b) Ditetapkan m = 7 , b = 10, dan N =256

Enkripsi : ܥ( ܲ) ≡ ( ݉ܲ + ܾ) ( ݉݋݀ ݊) (5)

⟺ ܥ( ܲ1) ≡ ( 7.65 + 10) ( ݉݋݀ 256)

⟺ ܥ( ܲ1) ≡ ( 455 + 10) ( ݉݋݀ 256)

⟺ ܥ( ܲ1) ≡ 465 ( ݉݋݀ 256)

⟺ ܥ( ܲ1) ≡ 209 ( ݉݋݀ 256)

Enkripsi : ܥ( ܲ) ≡ ( ݉ܲ + ܾ) ( ݉݋݀ ݊) (5)

⟺ ܥ( ܲ2) ≡ ( 7.66 + 10) ( ݉݋݀ 256)

⟺ ܥ( ܲ2) ≡ ( 462 + 10) ( ݉݋݀ 256)

⟺ ܥ( ܲ2) ≡ 472 ݉݋݀ 256

⟺ ܥ( ܲ2) ≡ 216 ( ݉݋݀ 256)

Didapat *ciphertext* enkripsi *affine* adalah **(209,216)**

3. Enkripsi kunci affine (7,10) menggunakan kunci publik RSA

- ditetapkan p dan q adalah 3 dan 11 (11 mod 3=1)

- ɸ ( ݊) = ( ݌ − 1) ( ݍ − 1) = ( 2) ( 10) = 20

- ݊ = ݌.q = 3 . 11 = 33

|  |  |  |
| --- | --- | --- |
|  | - Ambil secara acak kunci e dengan syarat 1 < ݁ < ɸ ( ݊)  - Sehinga didapat kunci publik RSA (e,n) = (3,33) |  |
| Proses enkripsinya : |
| ܥ݅ = ܲ݅ ௘ ݉݋݀ ݊  C1 = 73 mod 33 = 343 mod 33 = 13 | (3) |
| C2 = 103 mod 33 = 1000 mod 33 = 10 |  |
| Sehingga menghasilkan *cipherkey affine* yaitu (13,10) |  |
| **b.** | **Proses Dekripsi** |  |

1. *Cipherkey affine* yang diketahui (13,10)

2. *Cipherkey affine* ini di dekripsi menggunkan kunci privat RSA :

- Mencari kunci privat RSA

- Kunci d

݀ ݁ = 1 ( ݉݋݀ ɸ( ݊) )

|  |  |
| --- | --- |
| - Didapat kunci privatnya (d,n) = (7,33) |  |
| - Dekripsi *cipherkey affine* |
| - ܲ݅ = ܥ݅ ௗ ݉݋݀ ݊  P1 = 137 mod 33 = (134 mod 33) (133 mod 33) mod 33 | (4) |

Tabel 3.1. Perhitungan kunci privat “d” RSA

|  |  |
| --- | --- |
| d | d e (݉݋݀ ɸ ( ݊) ) |
| 1 | 3 |
| 2 | 6 |
| 3 | 9 |
| 4 | 12 |
| 5 | 15 |
| 6 | 18 |
| 7 | 1 |

= (16)(19) mod 33=304 mod 33

= 7

P2 = 107 mod 33 = (104 mod 33 ) (103 mod 33 ) mod 33

= (1)(10)mod 33

= 10

Didapat kunci *affine* (7,10)

- kunci *affine* tersebut di dekripsi dengan *affine cipher*

- ܲ( ܥ) ≡ ݉ିଵ ( ܥ − ܾ) ( ݉݋݀ ݊) (6)

Tabel 3.2. Perhitungan m-1 affine cipher pada contoh *hybrid*

|  |  |
| --- | --- |
| m-1 | m-1 m (mod n) =  m-1 7 (mod 256) |
| 1 | 7 |
| 2 | 14 |
| 3 | 21 |
| 4 | 28 |
| 5 | 35 |
| .  .  . | .  .  . |
| 183 | 1 |

Sehingga didapat m-1 = 183

P1= 183 (c-10) mod 256

*Ciphertext* **(209,216)**

C1=209,

P1 = 183(209-10) mod 256

P1 = 183(199) mod 256

P1 = 36417 mod 256

P1 = 65  A

C2=10

P2 = 183(216-10) mod 256

P2 = 183(206) mod 256

P2 = 37698 mod256

P2 = 66  B

Sehingga didapat *plaintext*-nya : A B

**3.2.2 Implementasi Algoritma Terhadap *File***

Pengamanan data informasi yang terdapat dalam suatu *file* sangat dibutuhkan dalam hal penyimpanan maupun dalam proses pengiriman. Karena diantaranya merupakan data-data informasi penting dan rahasia. Kerena adanya kemungkinan-kemungkinan dalam hal pencurian data, penyadapan, dan lain sebagainya, maka *file* tersebut harusnya diamankan dengan cara enkripsi data agar data dalam suatu *file* tersebut tidak dapat dibaca maupun dikenali tanpa proses dekripsi terlebih dahulu. Proses pengamanan data pada *file* dapat digambarkan pada gambar berikut:



Pada dasarnya data berupa *file* terbentuk dari bit-bit data *biner* yang nantinya akan diproses oleh komputer sebagai perintah-perintah dan akan menghasilkan tampilan maupun data pada akhir proses tersebut. Bit data tersebut memiliki informasi berupa struktur *file* dan termasuk data yang terdapat dalam *file* itu sendiri.

**3.2.3 *Flowchart* Sistem**

*Flowchart* sistem merupakan suatu bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada didalam sistem. Dengan kata lain, *flowchart* merupakan deskripsi secara grafik dari urutan prosedur-prosedur yang terkombinasi membentuk suatu sistem.

Adapun fungsi lainnya dari *flowchart* ini dibuat adalah agar *developer* atau pengembang aplikasi perorangan dapat melihat kembali tahapan-tahapan yang akan dikerjakan jika suatu saat terjadi kesalahan dalam perancangan aplikasi (Prayudi,

2013).

Berikut ini adalah bentuk-bentuk *flowchart* atau diagram alur yang menjelaskan alur proses aplikasi pengamanan data dari awal proses hingga akhir, termasuk proses pengenkripsian data dan mendekripsikan data itu kembali.

Pada gambar 3.2.1 berikut, proses enkripsi dapat dijelaskan bahwa aplikasi memulai dengan menampilkan menu *login user*, dan apabila belum mempunyai *data user* maka terlebih dahulu melakukan registrasi. Setelah memiliki *akun user* maka dapat masuk melalui login. Setelah berhasil login dengan memasukkan data nama dan password yang benar, maka akan masuk pada menu enkripsi dimana *user* dapat menginput *file* yang ingin di enkripsi. Setelah itu *file* akan dienkripsi sehingga tidak dapat dibaca oleh orang-orang yang tidak berhak. Apabila *file* tersebut memang sudah terenkripsi sebelumnya, maka *file* tersebut tidak dapat di enkripsi lagi.

Mulai

Punya Akun

*User*?

Tidak

Registrasi

*User*

Ya

Login

Nama dan

*Password* Benar Tidak

Ya

Input *File* yang akan di enkripsi

*Encrypt*

*File* Telah di Ya enkripsi?

*File* Tidak

Terenkripsi lagi

Tidak

*File* Berhasil dienkripsi

Selesai

Gambar 3.2.1 Proses enkripsi pada sistem



Pada Gambar 3.2.2 dapat di lihat bahwa pada proses dekripsi dimulai dari proses login, Setelah berhasil login dengan data yang benar, maka *user* dapat menginput *file* yang diinginkan untuk dibuka atau di dekripsi kembali. Setelah didekripsi, *file* akan dapat dibuka kembali seperti data aslinya. Disini perlu diperhatikan bahwa untuk mendekripsi suatu *file* yang telah di enkripsi haruslah *user* yang telah mengenkripsi *file* tersebut sebelumnya. Apabila *user* tidak sesuai maka *file* tidak dapat diproses untuk didekripsi.

**3.3. Perancangan Sistem**

Pada Penelitian ini perancangan sistem terbagi dua yakni menjelaskan tentang kebutuhan perangkat lunak yang digunakan dalam perancangan aplikasi dan perancangan antarmuka sistem yang nantinya akan digunakan pada setiap halaman aplikasi.

**3.3.1. Kebutuhan Perangkat Lunak**

Adapun aplikasi ini dibangun dengan menggunakan bahasa pemograman java. Untuk aplikasi pada sisi pengguna berjalan pada sistem operasi windows.

**3.3.2. Perancangan antarmuka sistem**

Perancangan sistem diperlukan agar sistem yang dibangun sesuai dengan yang diharapkan. Dengan adanya rancangan antarmuka bertujuan untuk memudahkan pengguna dalam pemakaian sistem. Berikut adalah beberapa rancangan antar muka sistem pada aplikasi :

1. Tampilan Menu Login

Berikut ini adalah rancangan menu utama login :

L O G I N

U s e r n a m e : A

P a s s w o r d : B

C L O G I N

D R e g i s t e r

Gambar 3.3.2.1 Rancangan Tampilan Menu Login

Pada Gambar 3.3.2.1, dapat dilihat bahwa terdapat 4 menu pilihan, yaitu form isian

*username*, *password*, *login*, dan *register.*

Keterangan :

A. Form isian *username* yang telah terdaftar

B. Form isian *password* yang telah terdaftar

C. *Login*, Tombol untuk masuk ke menu enkripsi dan dekripsi

D. *Register*, Tombol yang berisi form untuk mendaftar *user* yang belum didata.

2. Tampilan Menu *Register*

Register Form

Username : A Password : B

Re-type Password : C

REGISTRASI D

Gambar 3.3.2.2 Rancangan Tampilan Menu *Register*

Pada Gambar 3.3.2.2, dapat dilihat terdapat form untuk *user* yang baru. Terdapat 3 form yaitu *username*, *password*, dan *re-type password*. Dan terdapat tombol registrasi.

Keterangan :

A. *Username*, nama *user* yang ingin di simpan. B. *Password*, *Password* yang ingin di simpan.

C. *Re-type Password*, tulis ulang *password* yang diinginkan.

D. *Registrasi*, Tombol untuk menyimpan registrasi *user* yang baru.

3. Tampilan Menu Enkripsi

Berikut ini adalah tampilan dari menu enkripsi :

ENCRYPT FILE

Input File :

A Browse B

Encrypt File C

Gambar 3.3.2.3 Rancangan Tampilan Menu Enkripsi

Pada gambar 3.3.2.3, dapat dilihat bahwa terdapat form *input file*, tombol *Browse file*, dan tombol *encrypt file*.

Keterangan :

A. *Form input file*, tempat menyimpan nama *file* yang akan di enkripsi.

*B. Browse*, Pencarian *file* yang akan di enkripsi.

*C. Encrypt File,* Tombol untuk mengenkripsi *file* yang diinginkan.

4. Tampilan menu Dekripsi

Berikut ini adalah tampilan dari menu Dekripsi :

DECRYPT FILE

Input File :

A Browse B

Decrypt F ile C

Gambar 3.3.2.4 Rancangan Tampilan Menu Dekripsi

Pada gambar 3.3.2.4, dapat dilihat bahwa terdapat *form input file*, tombol *browse file*, dan tombol *decrypt file*.

Keterangan :

A. *Form input file*, tempat menyimpan nama *file* yang akan di dekripsi.

*B. Browse*, Pencarian *file* yang akan di dekripsi.

*C. Decrypt File,* Tombol untuk mendekripsi *file* yang diinginkan.

**3.4. Perancangan Database**

**3.4.1.Perancangan Tabel**

1. Tabel *data\_user*

Berikut ini adalah perancangan dari tabel *data\_user:*

Name Type

A ID

B Name

C Password

D n E d F e

INTEGER VARCHAR VARCHAR VARCHAR VARCHAR VARCHAR

Keterangan Gambar :

Gambar 3.4.1. Tabel *data\_user*

A. *ID*, atribut untuk menyimpan id dari tabel *data\_user* dengan tipe data *integer*

B. *Name*, atribut untuk menyimpan nama dari tabel *data\_user* dengan tipe data

*varchar*

C. *Password*, atribut untuk menyimpan *password* dari tabel *data\_user* dengan tipe data *varchar*

D. n, atribut untuk menyimpan data nilai n dari tabel *data\_user* dengan tipe data

*varchar*

E. d, atribut untuk menyimpan data nilai d dari tabel *data\_user* dengan tipe data

*varchar*

F. e, atribut untuk menyimpan data nilai i dari tabel *data\_user* dengan tipe data

*varchar*

2. Tabel *data\_file*

Berikut ini adalah perancangan dari tabel *data\_file*:

Name Type

A ID\_File

B ID

C Filename

INTEGER INTEGER

VARCHAR

Gambar 3.4.2. Tabel *data\_file*

Keterangan Gambar :

A. *ID\_File*, atribut untuk menyimpan id *file* dari tabel *data\_file* dengan tipe data

*integer*

B. *ID*, atribut untuk menyimpan id dari tabel *data\_user* dengan tipe data *integer*

C. *Filename*, atribut untuk menyimpan nama *file* dengan tipe data *varchar*

**3.4.2. Relasi Antar Tabel**

Relasi antar tabel dalam *database* berguna untuk keterkaitan identitas dari suatu tabel dengan tabel yang lainnya, agar identitas yang ada tidak *redudancy*. Berikut adalah gambar relasi antar tabel :

Tabel Data\_User

Name Type

Tabel Data\_File

Name Type

ID\_File INTEGER

ID Name Password

n

d e

INTEGER VARCHAR VARCHAR VARCHAR VARCHAR

VARCHAR

ID

Filename

INTEGER

VARCHAR

Gambar 3.4.3 Database Relationship

Pada gambar 3.4.3 terihat relasi antar tabel *data\_file* dan *data\_user* yang dihubungkan oleh *field ID*. *Field ID* merupakan id dari *tabel user* yang digunakan untuk *mengetahui id use*r dalam melakukan enkripsi dan dekripsi.

**BAB 4**

**IMPLEMENTASI DAN PENGUJIAN**

Bab ini akan membahas hasil yang didapatkan dari implementasi *Affine cipher* dan RSA dalam pengamanan data *file* sesuai dengan spesifikasi penerapan yang dibahas pada bab 3. Bab ini akan menjabarkan hasil dan proses enkripsi dan dekripsi, serta hasil perhitungan waktu metode *affine cipher* dan RSA untuk hasil enkripsi dekripsi.

**4.1 Spesifikasi Perangkat Lunak dan Perangkat Keras**

Pada proses pengimplementasian,lingkungan dimana proses pengoperasian aplikasi dan pengujian sistem berlangsung harus jelas, yang dimaksudkan untuk mengetahui hasil dari pengujian secara *significant*. Dalam hal ini spesifikasi dari perangkat keras dari perangkat lunak yang digunakan.

Spesifikasi perangkat keras yang digunakan dalam pembangunan aplikasi

1. *Processor* intel® Pentium®Processor P600

2. *Memory* RAM yang digunakan 2 GB

Spesifikasi perangkat lunak yang digunakan dalam pembangunan aplikasi

1. Windows 7 Ultimate

2. Eclipse Galileo

3. Java JRE 7

**4.2 Tampilan Sistem Secara menyeluruh**

Berikut tampilan yang terdapat pada aplikasi:

*1. Menu Login*



Gambar 4.2.1 Tampilan *Menu Login*

Pada tampilan awal aplikasi, *user* diwajibkan untuk login terlebih dahulu sebelum memasuki menu enkripsi dan dekripsi. Apabila belum mempunyai *akun* maka harus melakukan registrasi *user*. Sehingga akan mendapatkan *username* dan *password* sebagai user yang diinginkan.

2. Tampilan *Menu Register*



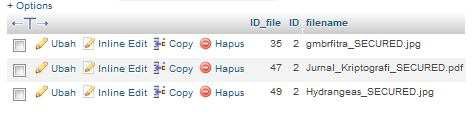
Gambar 4.2.2 Tampilan *Form Register Form*

Pada Tampilan *form* aplikasi, calon user harus mengisi data-data berupa *username* dan *password* yang di inginkan, sehingga data-data tersebut akan tersimpan di *database* dan kemudian *user* dapat login menggunakan *username* dan *password* yang telah diberikannya.

Setiap *user* yang telah terdaftar maka kunci yang didapat pada masing-masing *user* berbeda, jadi setiap *user* yang mengenkripsi suatu *file* maka harus *user* tersebut juga yang harus login untuk mengdekripsinya menjadi bentuk semula. Jika mendekripsi sebagai *user* lain maka struktur *file* tidak dapat diproses karena kunci d, n, dan e nya berbeda. Gambar berikut adalah *user* yang telah terdaftar beserta kunci yang telah dirandom terdata pada *database* :



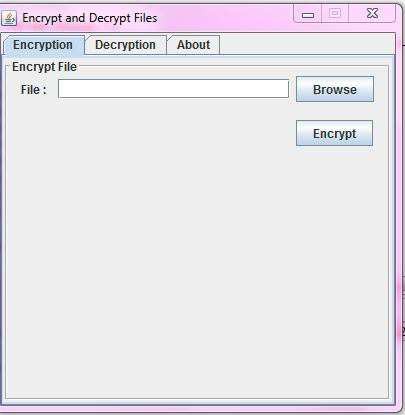
Gambar 4.2.3 Tampilan *Database data\_user*



Gambar 4.2.4 Tampilan *Database data\_file*

3. Tampilan Menu Enkripsi

Gambar 4.2.5 Tampilan Menu Enkripsi



Pada tampilan menu enkripsi, terdapat tampilan untuk menginput *file* yang ingin di enkripsi, setelah itu *file* akan terenkripsi dengan menggunakan *button encrypt*.

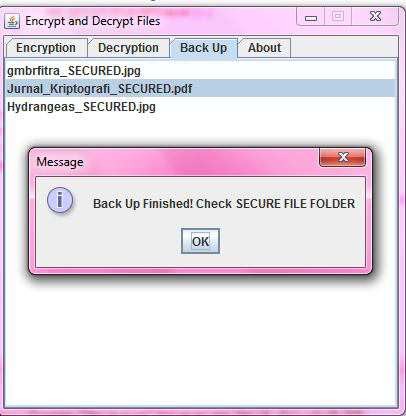
4. Tampilan Menu Dekripsi



Gambar 4.2.6 Tampilan Menu Dekripsi

Pada tampilan menu dekripsi, terdapat tampilan untuk menginput *file* yang ingin di dekripsi, setelah itu *file* akan terdekripsi dengan menggunakan button dekripsi.

5. Tampilan menu *backup*



Gambar 4.2.7 Tampilan Menu *backup*

Pada tampilan menu *backup* ini berfungsi untuk mem*backup* data apabila data di hapus atau di salahgunakan oleh orang-orang yang tidak berhak.

**4.2 Pengujian Sistem Secara menyeluruh**

Pengujian pada aplikasi pengamanan data berupa *file* dengan metode *affine cipher* dan RSA dititikberatkan pada kemampuan aplikasi dalam melakukan pengamanan data. Hal ini difokuskan pada *file* yang dienkripsi dapat mengamankan data dengan baik dan dekripsi untuk mengembalikan *file* kebentuk aslinya. Pengujian dilakukan pada beberapa jenis file yaitu dokument word, pdf, txt, jpeg, dan ppt, mp3, video, dsb

Untuk mengetahui apakah penelitian ini telah memenuhi tujuan penelitian yang telah disampaikan pada bab 1, maka diperlukanlah pengujian terhadap *software* yang dibangun dalam penelitian ini. Pengujian sistem menyeluruh yang dilakukan yaitu pengujian sistem pada aplikasi yang akan digunakan oleh *user*, dari segi tampilan dan segi proses yang terjadi pada setiap halaman dan melakukan analisis terhadap performa aplikasi pada saat proses enkripsi dan dekripsi data dengan menerapkan kriptografi. Pengujian pada sisi *user*, dilakukan pengujian pada semua proses yang terjadi pada aplikasi, seperti menu utama, enkripsi dan dekripsi.



Gambar 4.3.1.1 *Login User*

Pada gambar 4.3.1.1 terlihat bahwa *user* masuk dengan melakukan *login* terlebih dahulu, Setelah itu maka *user* dapat masuk ke menu enkripsi dan dekripsi *file* sbb:

Gambar 4.3.1.2 Melakukan enkripsi terhadap sebuah *file*



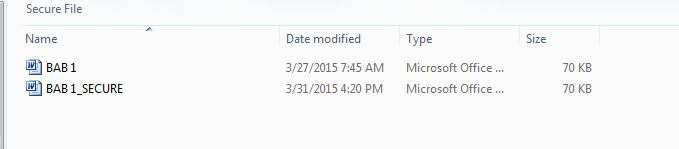
Pada gambar 4.3.1.2 terlihat bahwa *user* melakukan enkripsi terhadap sebuah *file* dengan cara memilih *file* yang ingin di enkripsi kemudian klik tombol *encypt*. Setelah itu proses akan berjalan dan jika enkripsi telah selesai dapat dilihat pada gambar

4.3.1.3 berikut :

Gambar 4.3.1.3 Proses enkripsi telah selesai



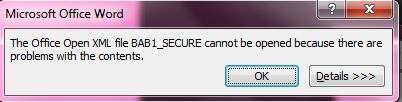
Pada gambar 4.3.1.3 Terlihat bahwa proses enkripsi telah selesai dilakukan. *File* yang telah di enkripsi akan mendapatkan penambahan nama "SECURE" dibelakangnya. misal *file* asli adalah BAB1.docx maka *file* terenkripsi menjadi BAB1\_SECURE.docx. *File* terenkripsi tersebut akan disimpan di dalam folder “*secure file”*.



|  |  |  |
| --- | --- | --- |
|  | |  |
| *File Asli* | *F* | *ile Enkripsi* |
|  |  | |

Gambar 4.3.1.4 f*ile* asli dan *file* enkripsi

Selanjutnya kita akan mencek apakah *file* yang telah di enkripsi tidak dapat diketahui informasinya terhadap orang lain yang tidak berhak.



Gambar 4.3.1.5 *File* yang telah di enkripsi dan tidak dapat dibuka/dibaca

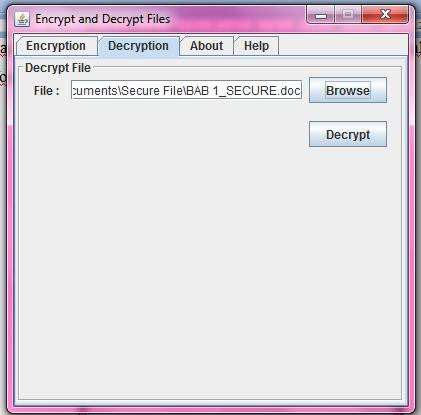
Pada gambar 4.3.1.5 terlihat *file* yang telah di enkripsi oleh *user* tadi tidak dapat dibuka dan dibaca informasinya oleh orang-orang yang tidak berkepentingan. Karena setelah di enkripsi, struktur data tersebut berubah dan teracak.

Gambar 4.3.1.6 *backup file* enkripsi



Pada gambar 4.3.1.6 Pada menu *backup* Terdapat *file-file* hasil enkripsi sebagai

*backup* buat *user*.



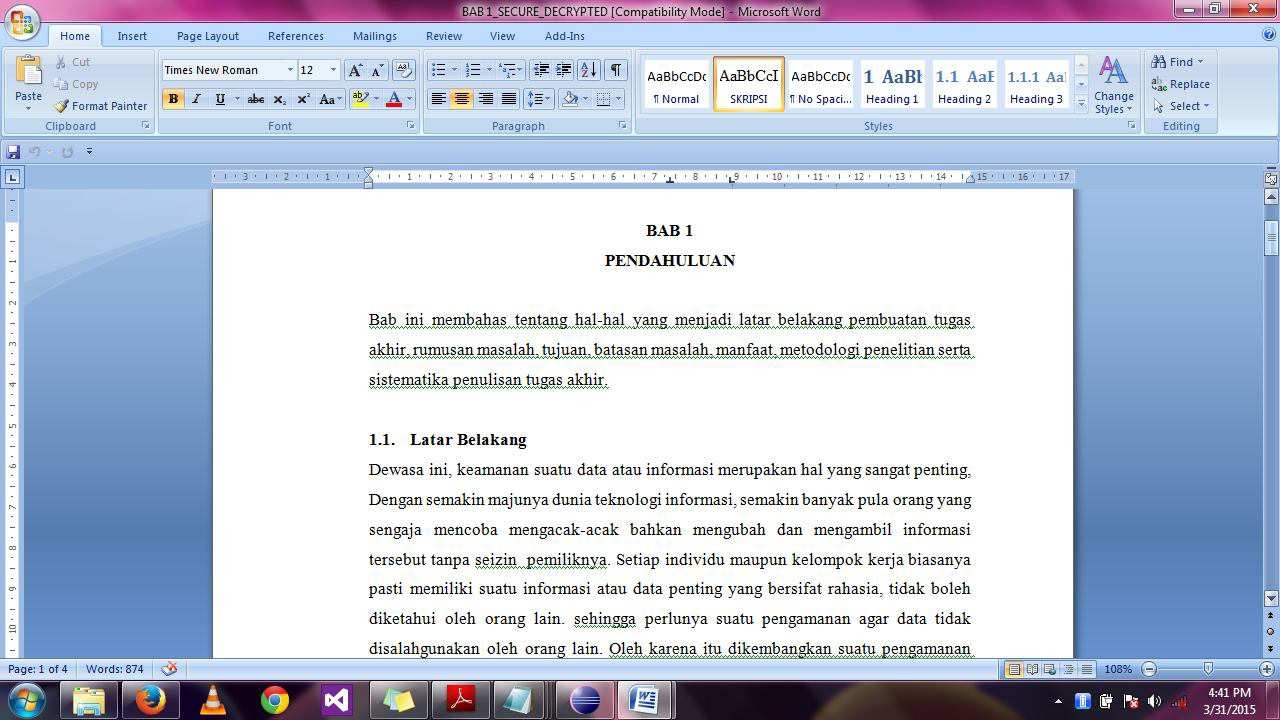
Gambar 4.3.1.7 Melakukan dekripsi



Pada gambar 4.3.1.7 terlihat bahwa *user* melakukan dekripsi terhadap *file* yang telah di enkripsi sebelumnya.

Gambar 4.3.1.8 Proses akhir dekripsi

Pada gambar 4.3.1.8 terlihat proses akhir dekripsi, dan *file* berhasil di dekripsi. Setelah di dekripsi kembali data tersebut, maka data kembali dapat dibaca dan diketahui informasinya seperti gambar 4.3.1.9 berikut :



Gambar 4.3.1.9 *File* yang telah di dekripsi kembali

Pada gambar 4.3.1.9 terlihat *file* yang telah di dekripsi kembali dapat dibuka dan dibaca kembali informasi yang terkandung di dalamnya.

Proses pengamanan data melakukan enkripsi pada bit data pada *file* tersebut sehingga struktur daripada *file* tersebut akan berubah dan tidak akan dapat dibuka sebelum melakukan proses dekripsi kembali. Pada proses dekripsi akan mengembalikan nilai dan struktur bit data pada *file* tersebut ke dalam bentuk semula. Proses pengembalian data pada aplikasi pengamanan data ini, memerlukan *user* yang sama, karena setiap *user* yang telah didaftar mempunyai kuncinya masing masing. Jika *user* berbeda maka *file* yang telah di enkripsi tadi tidak dapat diproses.

**4.3 Analisis Kriptografi Terhadap Sistem**

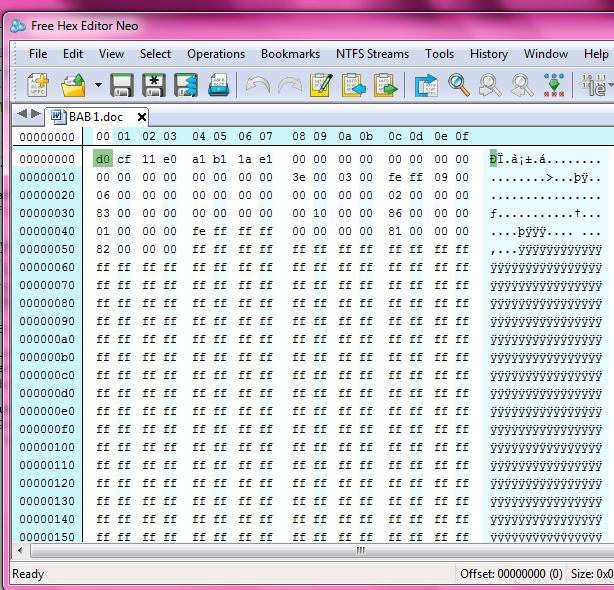
4.3.1 Analisis Keamanan Data *File* Terhadap Pengguna

Untuk menganalisis keamanan data *file* pengguna, maka perlu dicek apakah *file* yang diinginkan sudah terenkripsi atau tidak dapat diketahui informasinya terhadap orang lain yang tidak berhak. Hal ini dapat dilakukan dengan mengecek *file* tersebut apakah telah terenkripsi berbentuk *chipherfile* dan tidak dapat di mengerti orang lain. Setiap *user* yang telah terdata masing-masing dapat mengenkripsi *file* yang di inginkan. Dan kembali dapat mendekripsi *file* tersebut. Setiap *user* yang berbeda tidak dapat mendekripsi *file* yang telah di enkripsi oleh *user* yang lain. Jadi setiap *user* yang telah

mengenkripsi *file* maka harus *user* itu sendiri yang harus mendekripsinya. Karena setiap *user* memiliki kunci yang berbeda-beda.

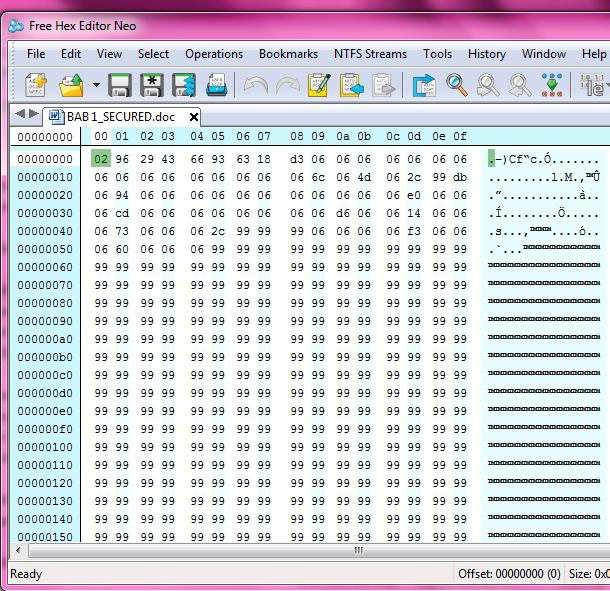
a. Pada Pengujian yang telah dilakukan pada *file* “BAB 1.docx”, Berikut adalah struktur *file* asli, file setelah di enkripsi, dan file setelah didekripsi dalam bentuk heksadesimal.

Gambar 4.4.1 berikut merupakan file asli “BAB1.docx” dalam heksadesimal:



Gambar 4.4.1. file “BAB 1.docx” dalam heksadesimal

File “BAB 1.docx” setelah di enkripsi di beri nama dengan “BAB 1\_SECURE.docx” Pada gambar 4.4.2 berikut struktur *file* hasil enkripsinya dalam heksadesimal.



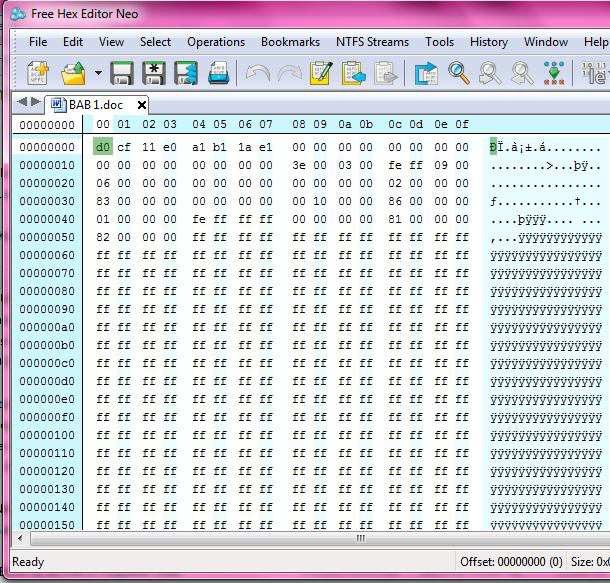
Gambar 4.4.2. File “BAB 1\_SECURE” dalam heksadesimal.

Pada gambar 4.4.2. bagian kiri menunjukkan struktur *file* dalam bentuk heksadesimal, sedangkan pada bagian kanan merupakan perwujudan bentuk karakter (ASCII) dari heksadesimal. Terlihat perbedaan antara *file* asli dan *file* hasil enkripsinya. Dengan begitu *file* tersebut sudah tidak bisa dimengerti.

Untuk menguji keberhasilan dan mengembalikan *file* menjadi bentuk semula maka dilakukan pendekripsian *file*. Setelah di dekripsi *file* akan kembali seperti semula seperti *file* aslinya.

Pada gambar 4.4.3 berikut merupakan file hasil dekripsi “BAB

1\_SECURE.docx” menjadi ke bentuk file aslinya “BAB 1.docx”.



Gambar 4.4.3. File “BAB 1.docx” hasil dekripsi dalam heksadesimal.

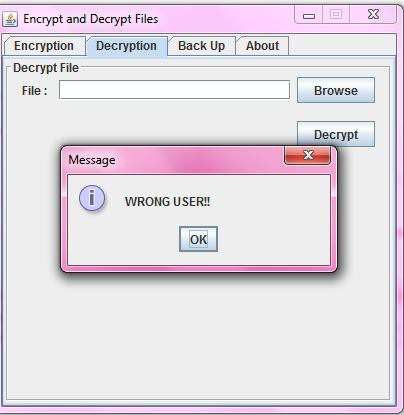
Hasil dekripsi menunjukkan bahwa susunan heksadesimalnya sesuai dengan *file* asli sebelum enkripsi.

b. *File* tidak bisa di enkripsi beberapa kali, jika *file* telah di enkripsi satu kali maka *file* tersebut tidak bisa di enkripsi lagi. Jika *file* di enkripsi lagi maka akan mincul pesan seperti gambar 4.4.4. berikut:



Gambar 4.4.4. *File* tidak dapat di enkripsi lebih dari sekali

c. Setiap *user* memiliki kunci yang berbeda-beda dan jika *user* yang berbeda mengenkripsi *file* yang sama maka *file* tersebut tidak dapat terdekripsi. Maka *user* yang mendekripsi haruslah *user* yang telah menenkripsi *file* tersebut sebelumya. Jika *user* yang mendekripsi salah, maka proses tidak dapat berjalan, dan akan muncul pesan seperti pada gambar 4.4.5. berikut :



Gambar 4.4.5. Menggunakan *user* yang salah

d. Jika *file* dihapus oleh orang yang lain maka *user* tetap memiliki *backup file*nya di dalam aplikasi, yang tersimpan di dalam *database* “*data\_file”*. Kita dapat mengembalikan *file backup* yang telah di hapus. Kemudian Jika file telah di dekripsi kembali maka *file backup-*nya akan terhapus juga.



Gambar 4.4.6 Menampilkan *backup file*

Pada gambar 4.4.6 *backup* data yang tesimpan dalam aplikasi dapat ditampilkan kembali.

e. Jika *file* di timpa dan diubah oleh orang-orang yang tidak berhak maka *user* akan tetap mimiliki *backup* dari *file*nya.

**4.4 Analisis Waktu Proses Enkripsi dan Dekripsi *File***

Tabel 4.1. Pengujian waktu proses enkripsi dan dekripsi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Nama *File* | Ukuran *File*  Asli(*Byte*) | Waktu Enkripsi (*millisecond*) | Waktu Dekripsi (*millisecond*) |
| 1 | Desert.jpg | 845,941 | 90 | 48 |
| 2 | TugasMakalahEP.docx | 31.070 | 44 | 40 |
| 3 | Analisis Sistem.txt | 997 | 52 | 42 |
| 4 | JurnalKriptografi.pdf | 318.539 | 44 | 44 |
| 5 | EtikaProfesi01.ppt | 174.080 | 42 | 40 |
| 6 | Coba.doc | 26.624 | 76 | 87 |
| 7 | PengamananData.pdf | 169.837 | 38 | 37 |
| 8 | BAB1.docx | 29.653 | 35 | 33 |
| 9 | Abstrack.txt | 469 | 40 | 52 |
| 10 | Standarisasi.ppt | 714.240 | 93 | 62 |
| 11 | Ocean.jpg | 175.805 | 44 | 44 |
| 12 | Bahan2.rar | 66.379.613 | 613 | 975 |
| 13 | Blue-all rise.mp3 | 3.571.712 | 51 | 91 |
| 14 | WebminSlackware.mp4 | 10.846.208 | 102 | 161 |
| 15 | Wildlife.wmv | 26.246.026 | 295 | 493 |
| Jumlah : | | | 1695 | 2249 |

Dari pengujian yang telah dilakukan pada tabel 4.1 di peroleh hasil rata-rata waktu enkripsi adalah 1695/15 = 113 *millisecond*, Dan rata-rata waktu dekripsi adalah

2249/15 = 150 *millisecond.* Ukuran *file* mempengaruhi lamanya proses enkripsi. Semakin besar ukuran *file*, maka akan semakin lama pula proses enkripsinya.

**BAB 5**

**KESIMPULAN DAN SARAN**

Bab ini membahas tentang kesimpulan yang didapat berdasarkan hasil penelitian mengenai pengamanan data *file* menggunakan metode *affine cipher* dan RSA. serta saran-saran untuk pengembangan penelitian selanjutnya.

**5.1 Kesimpulan**

Berdasarkan analisis dari sistem dan pengujian sistem secara menyeluruh yang dilakukan pada bab-bab sebelumnya, maka ada beberapa hal yang dapat dijadikan kesimpulan pada penelitian ini antara lain :

1. Lamanya waktu rata-rata enkripsi adalah 113 *millisecond*, Sedangkan rata-rata waktu dekripsi adalah 150 *millisecond.*

2. Ukuran *file* mempengaruhi lamanya proses enkripsi. Semakin besar ukuran *file*, maka akan semakin lama pula proses enkripsinya.

3. Enkripsi dapat dilakukan pada semua jenis file.

55

**5.2 Saran**

Penulis menyarankan dalam pengembangan penelitian lebih lanjut sebagai berikut:

1. Pengembang dapat menambahkan pengamanan data *file* dalam pengiriman dengan menambahkan fitur enkripsi dan dekripsi *drive* dan dijalankan pada *Local Area Network* (LAN) atau jaringan internet agar data yang dikirim juga dapat terjaga keamanannya.

2. Pengembang dapat menambahkan prinsip *steganografi* agar *file* dapat disembunyikan dan menjadi lebih aman.

3. Pengembang dapat menggunakan algoritma kriptografi lainnya dalam pengamanan data.

**DAFTAR PUSTAKA**

Basharat, I., Azam, F., & Muzaffar, A.W. 2012. Database Security and Encryption: A Survey Study*. International Journal of Computer Applications* (0975 – 888) Volume 47– No.12, June 2012. National Universit y of Sciences and Technology (NUST), H-12.

Benz, Jessica J. 2001. PGP : *A Hybrid Solution*. Sains Institude.

Chauhan, J., Dedhia, N., & Kulkarni, B. 2013. Enhancing Data Securit y by using Hybrid Cryptographic Algorithm*. International Journal of Engineering Science and Innovative Technology (IJESIT)* Volume 2, Issue 3, May 2013. University of Mumbai.

Fanidzar, Montaseri. 2010.Perancangan dan Pembuatan Aplikasi Enkripsi dan

Dekripsi *file* dan *folder* menggunakan metode *Twofish*. Skripsi. UIN Malang.

Hamzah, Rio. 2011. Implementasi algoritma RSA dan *Blowfish* untuk enkripsi dan dekripsi data menggunakan Delphi 7. Skripsi. UIN Jakarta.

Juliadi, dkk. 2013. Kriptografi Klasik dengan Metode Modifikasi Affine Cipher yang diperkuat dengan Vigenere Cipher. *Buletin Ilmiah Mat.Stat. dan terapannya.*

Kromodimoeljo, Sentot. 2010. *Teori dan Aplikasi Kriptografi*, SPK IT Consulting. Lestari, Puji. 2013. Implementasi Algoritma RSA (Rivest Shamir Adleman) Dalam

Sistem Enkripsi *File* dan Pengamanan *Folder*. Skripsi.UIN Yogyakarta.

Mollin, Richard A. 2007. *An Introduction to Cryptography 2nd Ed*. CRC Press LLC Munir, Rinaldi. 2006. *Kriptografi*. Bandung. Informatika.

Praphul, M.N., & Nataraj,K.R., 2013. FPGA Implementation of Hybrid Cryptosystem.

*International Journal of Emerging Science and Engineering (IJESE*) ISSN:

2319–6378, Volume-1, Issue-8, June 2013.

Prayudi , Satria. 2013. Kombinasi Algoritma *RSA 512-Bit Dan One Time Pad* Untuk

Pengamanan Pesan Pada *Nic Messenger*. Universitas Sumatera Utara

Rudianto. 2007. Analisis Keamanan Algoritma Kriptografi *RC6*. Skripsi. Teknik

Informatika ITB, Bandung.

57

Sadikin, R.2012. *Kriptografi Untuk Keamanan Jaringan*. Yogyakarta. Penerbit Andi. Shukla, S., & Verma, P.K. 2014. Implementation of Affine Substitution Cipher with

Keyed Transposition Cipher for Enhancing Data Security*. International Journal of Advanced Research in Computer Science and Software Engineering.*

Sitinjak, S., & Fauziah, Y. 2010. Aplikasi Kriptografi File Menggunakan Algoritma Blowfish. *Seminar Nasional Informatika 2010 (semnasIF 2010)* UPN ”Veteran” Yogyakarta, 22 Mei 2010 ISSN: 1979-2328. Teknik Informatika UPN "Veteran" Yogyakarta.

Suryawan, S.H. & Hamdani, 2013. Pengamanan Data File Menggunakan Algoritma Enkripsi Rivest Code 5. *Jurnal Informatika Mulawarman Vol. 8 No. 2*. Universitas Mulawarman

Syaputra, H., Herdiyatmoko, & Hendrik, F. 2012. Aplikasi Enkripsi Data pada File Teks dengan Algoritma RSA( Rivest Shamir Adleman). *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2012 (Semantik 2012)* ISBN 979

- 26 - 0255 – 0 Semarang, 23 Juni 2012. Teknik Informatika, Sekolah Tinggi

Teknik Musi, Palembang 30113.

Wahyuni, A. 2011. Aplikasi Kriptografi Untuk Pengamanan *E-Dokument* Dengan Metode *Hybrid* : *Biometrik* Tanda Tangan dan DSA*(Digital Signature Algoritm)*. Tesis. Universitas Diponogoro, Semarang.

Winata, G.R. 2012. Eksplorasi Kombinasi Dua buah teknik cipher. *makalah kriptografi Institute Teknologi Bandung.*